OULU 2006
C 237

ACTA UNIVERSITATIS OULUENSIS

Olga Kayo

LOCALLY LINEAR EMBEDDING ALGORITHM

EXTENSIONS AND APPLICATIONS

FACULTY OF TECHNOLOGY,
DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING,
UNIVERSITY OF OULU

ACTA UNIVERSITATIS OULUENSIS
C Technica 237

OLGA KAYO

LOCALLY LINEAR EMBEDDING ALGORITHM

Extensions and applications

Academic Dissertation to be presented with the assent of the Faculty of Technology, University of Oulu, for public discussion in the Auditorium TS101, Linnanmaa, on April 21st, 2006, at 12 noon

OULUN YLIOPISTO, OULU 2006

Acta Univ. Oul. C 237, 2006

Supervised by
Professor Matti Pietikäinen

Reviewed by
Doctor Pasi Koikkalainen
Doctor Jaakko Peltonen

ISBN 951-42-8040-7 (Paperback)
ISBN 951-42-8041-5 (PDF) http://herkules.oulu.fi/isbn9514280415/
ISSN 0355-3213 (Printed)
ISSN 1796-2226 (Online) http://herkules.oulu.fi/issn03553213/

Cover design
Raimo Ahonen

OULU UNIVERSITY PRESS
OULU 2006

Kayo, Olga (neè Kouropteva), Locally linear embedding algorithm. Extensions and applications
Faculty of Technology, University of Oulu, P.O.Box 4000, FI-90014 University of Oulu, Finland,
Department of Electrical and Information Engineering, University of Oulu, P.O.Box 4500, FI-90014 University of Oulu, Finland
Acta Univ. Oul. C 237, 2006
Oulu, Finland

Abstract

Raw data sets taken with various capturing devices are usually multidimensional and need to be preprocessed before applying subsequent operations, such as clustering, classification, outlier detection, noise filtering etc. One of the steps of data preprocessing is dimensionality reduction. It has been developed with an aim to reduce or eliminate information bearing secondary importance, and retain or highlight meaningful information while reducing the dimensionality of data.

Since the nature of real-world data is often nonlinear, linear dimensionality reduction techniques, such as principal component analysis (PCA), fail to preserve a structure and relationships in a highdimensional space when data are mapped into a low-dimensional space. This means that nonlinear dimensionality reduction methods are in demand in this case. Among them is a method called locally linear embedding (LLE), which is the focus of this thesis. Its main attractive characteristics are few free parameters to be set and a non-iterative solution avoiding the convergence to a local minimum. In this thesis, several extensions to the conventional LLE are proposed, which aid us to overcome some limitations of the algorithm. The study presents a comparison between LLE and three nonlinear dimensionality reduction techniques (isometric feature mapping (Isomap), self-organizing map (SOM) and fast manifold learning based on Riemannian normal coordinates (S-LogMap) applied to manifold learning. This comparison is of interest, since all of the listed methods reduce high-dimensional data in different ways, and it is worth knowing for which case a particular method outperforms others.

A number of applications of dimensionality reduction techniques exist in data mining. One of them is visualization of high-dimensional data sets. The main goal of data visualization is to find a one, two or three-dimensional descriptive data projection, which captures and highlights important knowledge about data while eliminating the information loss. This process helps people to explore and understand the data structure that facilitates the choice of a proper method for the data analysis, e.g., selecting simple or complex classifier etc. The application of LLE for visualization is described in this research.

The benefits of dimensionality reduction are commonly used in obtaining compact data representation before applying a classifier. In this case, the main goal is to obtain a low-dimensional data representation, which possesses good class separability. For this purpose, a supervised variant of LLE (SLLE) is proposed in this thesis.

Keywords: classification, clustering, dimensionality reduction, locally linear embedding, visualization

To the memory of my grandmother

Acknowledgments

The work reported in this thesis was carried out in the Machine Vision Group of the Department of Electrical and Information Engineering at the University of Oulu, Finland during the years 2002-2005.

I would like to express my deep gratitude to Professor Matti Pietikäinen for allowing me to work in his research group and providing me with excellent facilities for completing this thesis. I have received invaluable benefits from his supervision and guidance of my thesis. I also wish to give my special thanks to Docent Oleg Okun, who has been an adviser of my PhD study, for his enormous support during my work in the laboratory. I am very thankful for their constructive criticism and useful advice, which improved the quality of my skills in doing research and writing scientific publications.

My best thanks to all the members of the Machine Vision Group for providing a friendly atmosphere. Especially, I would like to thank Abdenour Hadid for his interest in my research, Jukka Kontinen, Ilkka Mattila, Hannu Rautio for technical support, and Anu Angeria for her advise in social life.

Some results presented in the thesis were obtained during joint research with Dutch colleagues. I am very grateful to Dick De Ridder and Robert P.W. Duin for allowing me to include our joint work.

The generous financial support for this thesis provided by Infotech Oulu Graduate School, the Nokia Foundation and Oulu University Scholarship Foundation is gratefully acknowledged.

I would like to give my sincere and warm thanks to Professor Jussi Parkkinen, the head of the Department of Computer Science at the University of Joensuu for providing an opportunity to get Master's degree in computer science (International Master Program in Information Technology (IMPIT)) and his assistance in finding the position at the University of Oulu.

Finally, I would like to give many thanks to my parents, husband, daughter, and my friends for supporting me during my study.

Oulu, October 2005 Olga Kayo (Kouropteva)

Symbols and abbreviations

1D, 2D, 3D	One-dimensional, two-dimensional, three-dimensional
1-SLLE	Fully supervised locally linear embedding
α-SLLE	Partially supervised locally linear embedding
BMU	Best matching unit
CDA	Curvilinear distance analysis
EM	Expectation maximization
HLLE	Hessian locally linear embedding
ICA	Independent component analysis
ID	Intrinsic dimensionality
ILLE	Incremental locally linear embedding
Isomap	Isometric feature mapping
KLLE	Kernelized locally linear embedding
KNN	K nearest neighbor classifier
LDA	Linear discriminant analysis
LEM	Laplacian Eigenmap
LG	Linear generalization
LLE	Locally linear embedding
S-LogMap	Fast manifold learning based on Riemannian normal coordinates
LPP	Locality preserving projection
LSI	Latent semantic indexing
LVQ	Learning vector quantization
MDS	Multidimensional scaling
NNs	Neural networks
OCM	Orthogonal centroid method
PCA	Principal component analysis
PP	Projection pursuit
RGB	Red-green-blue
SOM	Self-organizing map

SVD	Singular value decomposition
VF1	The first variant of the visualization framework
VF2	The second variant of the visualization framework
$\boldsymbol{0}_{d\times 1}$	Zero-vector of length d
$\mathbf{B}$	Inner product matrix
$B(\boldsymbol{x}_p)$	A ball of the K closest points around $\boldsymbol{x}_p$
b	Winner unit at SOM
$\mathbf{C}$	Matrix containing translation component used to calculate procrustes measure
C	Clustering performance coefficient
C_x	Clustering performance of the original data
C_y	Clustering performance of the projected data
$\boldsymbol{c}_i$	i^{th} column of the matrix $\mathbf{C}$
c	The number of nonzero entries per column of data matrix
$\mathbf{D}_x$	Matrix of pairwise Euclidean distances of original data points
$\mathbf{D}_y$	Matrix of pairwise Euclidean distances of projected data points
$\mathbf{D}_{\mathscr{G}}$	Matrix of geodesic distances
$d_x(\boldsymbol{x}_i,\boldsymbol{x}_j)$	Euclidean distance between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
$d_{\mathscr{G}}(i,j)$	Geodesic distance estimated in the graph $\mathscr{G}$ between nodes i and j
D	Dimensionality of original data
d	Dimensionality of projected data or intrinsic dimensionality of data (depending on context)
d_G	Global intrinsic dimensionality for a data set
d_L	Local intrinsic dimensionality for a data set
$\check{d}_i$	Difference between the two ranks squared
e	Euler's number, $e = 2.7182818285$
$\{\hat{\boldsymbol{e}}_i\}$	An orthonormal basis
$\mathbf{F}$	Orthogonal matrix containing the left singular vectors of $\mathbf{X}$
$\mathbf{F}_d$	Matrix containing left singular vectors corresponding to the d largest singular values of $\mathbf{X}$
f, $f(x)$	Smooth function on a manifold
$\mathbf{G}$	Gram matrix
g_{ij}	ij^{th} element of Gram matrix $\mathbf{G}$
g_{ij}^{-1}	ij^{th} element of inverse the Gram matrix $\mathbf{G}$
$\check{\boldsymbol{g}}, g^i$	Gradients
$\mathbf{H}$	Matrix used in Isomap
h_{ij}	ij^{th} element of matrix $\mathbf{H}$
H_f	Hessian of f
$h_{bi}(t)$	Neighborhood function around the winner unit b at time t
$\mathbf{I}_{d\times d}$	$d \times d$ identity matrix

i, j, k, l, m	Indices
K, L	The number of nearest neighbors of particular point
K_i^*	i^{th} potential candidate for K_{opt}
K_{opt}	Optimal value for K used in LLE
K_{opt}	Optimal value for K in terms of residual variance
M	The number of nodes in SOM
$\mathbf{M}$	Cost matrix
$\mathbf{M}_{new}$	Cost matrix calculated by ILLE
$\boldsymbol{m}_i$	Weight vector of unit i in SOM
N	The number of data points
N_i	The number of points in class ω_i
N_S	The number of potential candidates for K_{opt}
$N_{classes}$	The number of classes in data
$N_x^{correct}$	The number of correctly classified data samples in original space
$N_y^{correct}$	The number of correctly classified data samples in projected space
n	Parameter
$\mathbf{P}$	Estimation of mixing matrix in ICA
$Procr$	Procrustes measure
p	Parameter or index (depending on context)
$\mathbf{Q}$	Matrix containing scale, orthogonal rotation and reflection components used to calculate procrustes measure
R	Classification rate reduction
$\mathbb{R}^D$, $\mathbb{R}^d$	D- and d-dimensional space
$\mathbf{S}$	Diagonal matrix containing the singular values of $\mathbf{X}$
$\mathbf{S}_B$	Between-cluster scatter matrices
$\mathbf{S}_W$	Within-cluster scatter matrices
S	Set of potential candidates for K_{opt}
$\tilde{\mathbf{S}}$	Set of original sources in ICA
$\tilde{\boldsymbol{s}}_i$	i^{th} source in ICA
$\mathbf{T}$	Estimation of source matrix in ICA
t	Parameter
$\mathbf{U}$	Matrix of linear transformation for PCA
$\boldsymbol{u}_i$	i^{th} tangent coordinate of a manifold
$\mathbf{V}$	Orthogonal matrix containing the right singular vectors of $\mathbf{X}$
v	Parameter
$\boldsymbol{v}_i$	Eigenvector corresponding to the i^{th} eigenvalue of a matrix
$\mathbf{W}$	Weight matrix
$\boldsymbol{w}_i$	i^{th} column of weight matrix $\mathbf{W}$
w_{ij}	ij^{th} element of weight matrix $\mathbf{W}$
$\mathbf{X}$	Matrix of original data points

$\mathbf{X}^i$	K nearest neighbors of $\boldsymbol{x}_i$
$\boldsymbol{x}$;$\boldsymbol{x}_i$	An original data point; the i^{th} original data point
$\boldsymbol{x}_{N+1}$	An unseen original data point
$\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^K$	K nearest neighbors of $\boldsymbol{x}_i$
$\boldsymbol{x}^i$	i^{th} Riemannian normal coordinate
$\mathbf{Y}$	Matrix of projected data points
$\mathbf{Y}^i$	K nearest neighbors of $\boldsymbol{y}_i$
$\mathbf{Y}_{new}$	New embedded coordinates obtained for $\mathbf{X}$ augmented by new data point(s)
$\boldsymbol{y}$;$\boldsymbol{y}_i$	A projected data point; the i^{th} projected data point
$\boldsymbol{y}_{N+1}$	A projection of an unseen original data point $\boldsymbol{x}_{N+1}$
$\tilde{\boldsymbol{y}}_i$	The $(i+1)^{th}$ eigenvector corresponding to the $(i+1)^{th}$ eigenvalue of the cost matrix $\mathbf{M}$
$\mathbf{Z}$	Linear transformation matrix
α	Parameter of α-SLLE
$\alpha(t)$	Learning rate of SOM at time t
Δ	Distances between pairs of points
ε	Parameter or radius (depending on context)
θ	Infinitesimal quantity
$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$	Kernel function of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
Λ	Matrix used in SLLE
λ_i	i^{th} eigenvalue of a matrix
μ	Mean of data
μ_i	Mean of class ω_i
$\rho_{\mathbf{D}_x\mathbf{D}_y}$	Standard linear correlation coefficient taken over $\mathbf{D}_x$ and $\mathbf{D}_y$
ρ_{Sp}	Spearman's rho
ω_i	i^{th} class of data
$\Delta^{(tan)}(f)$	Laplacian operator in tangent coordinates
δ_{ij}	Delta function: $\delta_{ij} = 1$ if $i = j$ and 0 otherwise
$J(\cdot)$	Feature selection criterion
$O(\cdot)$	Complexity function
$Tr(\cdot)$	Trace of a matrix
$\Phi(\mathbf{Y})$	Embedding cost function
$\phi(\boldsymbol{x})$	Mapping function from original data space into a feature space
$\varepsilon(\mathbf{W})$	Weight cost function
$\mathscr{G}, \mathscr{G}_L$	Graphs
$\mathscr{L}, \mathscr{H}$	Functionals
$\mathscr{M}$	Manifold
$T_p\mathscr{M}$	Orthonormal basis of the tangent space at point $\boldsymbol{x}_p$ to the manifold $\mathscr{M}$
$\mathscr{X}$	Feature subset
$\mathscr{Y}, \mathscr{Y}_1, \mathscr{Y}_2$	Feature subsets

$\mathscr{L}$	Feature subset
$\|\| \cdot \|\|$	Euclidean norm (distance measure)
$\|\| \cdot \|\|_F$	Frobenius norm
$.^T$	Vector or matrix transposition

Contents

1 Introduction

1.1 Dimensionality reduction

Many problems in machine learning begin with the preprocessing of raw multidimensional data sets, such as images of any objects, speech signals, spectral histograms etc. The goal of data preprocessing is to obtain more informative, descriptive and useful data representations for subsequent operations, e.g., classification, visualization, clustering, outlier detection etc.

One of the operations of data preprocessing is dimensionality reduction. Because high-dimensional data can bear a lot of redundancies and correlations hiding important relationships, the purpose of this operation is to eliminate the redundancies of the data to be processed. Dimensionality reduction can be done either by feature selection or by feature extraction. Feature selection methods choose the most informative features among those given; therefore low-dimensional data representation possesses a physical meaning. In their turn, feature extraction methods obtain informative projections by applying certain operations to the original features. The advantage of feature extraction over feature selection methods is that if one is given the same dimensionality of reduced data representation, the transformed features might provide better results in further data analysis.

There are two possibilities to reduce dimensionality of data: supervised, when data labels are provided, and unsupervised, when no data labels are given. In most cases in practice, no prior knowledge about data is available, since it is very expensive and time consuming to assign labels to the data samples. Moreover, human experts might assign different labels to the same data sample, which may lead to incorrectly discovered relationships between the data samples, and can affect the results of subsequent operations. Therefore, nowadays, unsupervised methods discovering the hidden structure of the data are of prime interest.

To obtain a relevant low dimensional representation of high dimensional data, several manifold learning algorithms (Roweis & Saul 2000, Tenenbaum *et al.* 2000, DeCoste 2001, Belkin & Niyogi 2003, Donoho & Grimes 2003) have been recently proposed. The main assumption used in manifold learning algorithms is that high-dimensional data points form a low-dimensional manifold embedded into the high-dimensional space. Many types of high-dimensional data can be characterized in this way, e.g., images generated by a fixed object under different illumination conditions have high extrinsic dimen-

sions (number of pixels), whereas the number of natural characteristics is lower and corresponds to the number of illumination sources and their intensities. In a manifold learning problem one seeks to discover low-dimensional representations of large high-dimensional data sets, which provide better understanding and preserve the general properties of the data sets.

There are a number of applications of dimensionality reduction techniques. Thus, they are widely used in visualizing high-dimensional data sets. Since a human observer cannot visually perceive a high-dimensional representation of data, its dimensionality is reduced to one, two or three by applying a dimensionality reduction technique. The main goal of data visualization is to find a one, two or three-dimensional descriptive data projection which captures important knowledge about data, loses the least amount of information and helps people to understand and explore the data structure.

Another good example where the benefits of dimensionality reduction techniques can be used is a classification where the goal is to achieve a low-dimensional projection (may be larger than two or three) in which the data classes are clearly separated. Here, dimensionality reduction plays an important role, since classification of data in a high-dimensional space requires much more time than classification of reduced data. Moreover, the curse of dimensionality may affect the result of applying a classifier to high-dimensional data, thereby reducing the accuracy of label prediction. Hence, in order to avoid the curse of dimensionality it is suggested that one uses a dimensionality reduction method before applying a classifier. Since labels are provided in this task, it would be better to use them while reducing data dimensionality in order to achieve better class separability in a low-dimensional space, i.e., to apply a supervised dimensionality reduction method to the high-dimensional data.

1.2 Locally linear embedding algorithm

In this thesis a manifold learning algorithm called locally linear embedding (LLE) (Roweis & Saul 2000) is considered. This is an unsupervised non-linear technique that analyses high-dimensional data sets and reduces their dimensionalities while preserving local topology, i.e. the data points that are close in the high-dimensional space remain close in the low-dimensional space. The advantages of LLE are 1) only two parameters to be set; 2) a single global coordinate system of the embedded space; 3) good preservation of local geometry of high-dimensional data in the embedded space and 4) a non-iterative way of solving well scaling to large, high-dimensional data sets (due to a sparse eigenvector problem) and avoiding the problem with local minima plaguing many iterative techniques.

LLE obtains a low-dimensional data representation by assuming that even if the high-dimensional data forms a nonlinear manifold, it still can be considered locally linear if each data point and its neighbors lie on or close to a locally linear patch of the manifold. The simplest example of such cases is the Earth - its global manifold is a sphere, which is described by a nonlinear equation $x^2 + y^2 + z^2 = r^2$, where (x, y, z) are coordinates of three-dimensional space and r is the radius of the sphere, while locally it can be considered as a linear two-dimensional plane $ax + by = 0$, where (x, y) are coordinates and a, b are

coefficients. Hence, the sphere can be locally approximated by linear planes instead of convex ones. Unfortunately, LLE cannot deal with closed data manifolds such as sphere or torus. In this case, one should manually cut the manifold before processing it, e.g., to delete a pole from a sphere.

Because of the assumption that local patches are linear, i.e., each of them can be approximated by a linear hyperplane, and each data point can be represented by a weighted linear combination of its neighborhood: either K nearest neighbors or points belonging to a circle of radius ε, coefficients of this approximation characterize local geometries in a high-dimensional space and they are then used to find low-dimensional embeddings preserving the geometries in a low-dimensional space. The main point in replacing the nonlinear manifold locally with the linear hyperplanes is that this operation does not bring significant error, because, when locally analyzed, the curvature of the manifold is not large.

1.3 The scope and contribution of the thesis

This thesis presents a case study of the LLE algorithm, its extensions and applications to data mining problems, such as visualization and classification. The original LLE algorithm possesses a number of limitations that make it to be less attractive for scientist. First of all, a natural question is how does one choose the number of nearest neighbors K to be considered in LLE, since this parameter dramatically affects the resulting projection? A large K causes smoothing or eliminating of small-scale structures in the data manifold, whereas a small amount of nearest neighbors can falsely divide the continuous data manifold into several disjointed components. To answer the question, a procedure for automatic selection of the optimal value for the parameter K is proposed in the thesis.

The second LLE parameter, which has to be defined by a user, is a dimensionality of the projected space d. It is natural that for visualization purposes $d = 1, 2$ or 3. But what about other cases when one needs to preprocess data before applying subsequent operations, e.g., a classifier? In the thesis, the LLE approach for calculating an approximate intrinsic dimensionality (Polito & Perona 2002) is discussed and compared with one of the classical methods of estimating intrinsic dimensionality (Kégl 2003).

The conventional LLE algorithm 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 LLE for the previously seen data set augmented by the new data points. Because of this fact, LLE cannot be used for large data sets in a dynamic environment where a complete rerun of the algorithm becomes prohibitively expensive. An extension overcoming this problem is proposed in the thesis. This is an incremental version of LLE which allows dealing with sequentially incoming data.

LLE is designed for unsupervising learning. In this thesis its comparison to other unsupervised techniques (isometric feature mapping (Isomap), self-organizing map (SOM) and S-LogMap) is considered:

– Isomap (Tenenbaum *et al.* 2000) is closely related to LLE, since both of them try to preserve local properties of data and, based on them, to obtain a global data repre-

sentation in a low-dimensional space.
- SOM (Kohonen 1995, 1997) obtains a low-dimensional data representation structured in a two or higher dimensional grid of nodes. It belongs to the state-of-the-art techniques and is widely used in practice (Kaski *et al.* 1998, Oja *et al.* 2003).
- S-LogMap (Brun *et al.* 2005) is a novel, fast and promising technique for manifold learning. Its performance has been explored only on smooth low-dimensional data manifolds with uniform distribution.

Because of these facts, it is very interesting to compare LLE with these algorithms and to figure out in which cases LLE can outperform them.

Any dimensionality reduction technique can be used for visualization by merely reducing the original data dimensionality to two or three: linear dimensionality reduction methods might obtain resulting axes that are easier to interpret (axes have intuitive meanings) than those produced with nonlinear methods. However, nonlinear dimensionality reduction methods are mostly used for visualization in practice, since they outperform linear ones in preserving data topology, discovering intrinsic structures of nonlinear data manifolds etc. Nevertheless, even these nonlinear techniques can suffer from two problems. First, dimensionality reduction depends on a metric that captures relationships between data points, and if the metric function is not properly chosen for particular data, then the visualization results are poor. Second, since the dimensionality of the original data is typically large, determining correct similarity relationships in high-dimensional space and preserving them in visualization space can be difficult because of dramatic dimensionality reduction. To overcome these two problems a new visualization framework is proposed in the thesis. Its features are:

- Metric learning: employment of metric learning in order to capture specific (non-Euclidean) similarity and dissimilarity data relationships by using label information of a fraction of the data.
- A semi-supervised scheme of visualization: since the metric learning procedure does not need all data samples to be labeled (the precise amount is data-driven), the framework occupies an intermediate place between unsupervised and supervised visualization.
- A two-level dimensionality reduction scheme: the original data dimensionality is first reduced to the local/global intrinsic data dimensionality, which afterwards is reduced to one, two or three. This approach softens changes in the data dimensionality, thus preventing severe topology distortion during visualization.

The original LLE belongs to the class of unsupervised methods that are mostly designed for data mining when the number of classes and relationships between elements of different classes are unknown. To complement the original LLE, a supervised LLE (SLLE) extending the concept of LLE to multiple manifolds is proposed in the thesis. It uses membership information of data in order to decide which points should be included in the neighborhood of each data point. Hence the objectives of LLE and SLLE are different: LLE attempts to represent the data manifold as well as possible, whereas SLLE retains class separability as well as possible.

Publications (Okun *et al.* 2002, De Ridder *et al.* 2003a,b, Kouropteva *et al.* 2002a,b, 2003a,b, 2004a,b, 2005a,b,c) reflect the author's contribution to the existing theory. All of

the work has been carried out in compliance with the guidelines of the author's thesis supervisor, Prof. Matti Pietikäinen. The co-authors have also added their important merits. None of the publications has been used as a part of any other person's work.

The author's own contributions are generalization algorithms for LLE: linear generalization 1 (LG1) and incremental LLE (ILLE); a visualization framework, and supervised variant of LLE (1-SLLE). The method for defining the value of the optimal number of nearest neighbors used in LLE was proposed by the author and Dr. Oleg Okun. The author did all the experiments presented in Chapters 3, 4, 5 and partially those in Chapter 6. The material in Chapter 6 is mainly based on the joint publication (De Ridder *et al.* 2003a), where the author contribution is the 1-SLLE algorithm and partially the experimental part.

1.4 The outline of the thesis

The thesis starts with an introduction to the dimensionality reduction methods and a review of the related work. The detailed description of the LLE algorithm and its several extensions are presented. Next, LLE is applied to data mining tasks, such as unsupervised learning, visualization and classification, and compared with several known unsupervised techniques for dimensionality reduction. The results obtained demonstrate the strong and weak points of LLE, which are discussed at the end of the thesis. A more detailed description of the chapters is given in the following.

Chapter 2 provides an introduction to the problem of dimensionality reduction. First, the effect known as the curse of dimensionality is described and possibilities to overcome this problem are suggested. One of them is to reduce the dimensionality of the original data that can be done either by feature extraction or by feature selection. Methods belonging to the both variants are described in brief, and are summarized at the end of the chapter in two tables.

Chapter 3 describes the original LLE algorithm and its extensions allowing us to automatically define the number of nearest neighbors K and deal with sequentially incoming data points. Moreover, an approach to estimating the data intrinsic dimensionality by using the LLE algorithm proposed in (Polito & Perona 2002) is discussed in this chapter. Experiments demonstrate that the proposed extensions are applicable to real-world data sets.

Chapter 4 starts with a description of unsupervised learning and its application in data mining. Then three unsupervised learning methods (Isomap, SOM and S-LogMap) are described and compared to LLE. This is done by calculating evaluation criteria estimating the performance of the mapping methods.

Chapter 5 presents a new semi-supervised visualization framework that uses LLE as an intermediate step. This framework achieves a one, two or three-dimensional representation of high-dimensional data by utilizing a two-level dimensionality reduction scheme that allows eliminating an enormous loss of information content of the data while obtaining the most descriptive visualization components.

Chapter 6 extends the concept of LLE to multiple manifolds by providing two supervised variants of LLE that were independently proposed (Kouropteva *et al.* 2002a,

De Ridder & Duin 2002). In the experiment part, both variants of SLLE, principal component analysis, linear discriminant analysis, multidimensional scaling, and locally linear embedding are applied as feature extractors to the number of data sets before applying a classifier. Then, the classification is done in the reduced spaces by nearest mean, K nearest neighbor and Bayes plug-in (linear and quadratic) classifiers. The results demonstrate for which kind of data sets the proposed SLLE works better than the other feature extraction techniques.

Chapter 7 discusses the material described in the previous chapters depicting the weak and strong points of LLE and bringing up new questions for further research on the algorithm. Finally, Chapter 8 concludes the thesis.

Appendix 1 gives a detailed description of the data sets used in the experiments throughout the thesis. Appendix 2 describes the meaning of ill-conditioning of eigenvectors and eigenvalues of the Hermitian matrix. Appendix 3 introduces an algorithm of data intrinsic dimensionality estimation by packing numbers proposed by Kégl (Kégl 2003). Appendix 4 gives a definition of the intrinsic mean. Appendix 5 overviews global and local data intrinsic dimensionality estimation by PCA.

2 Overview of dimensionality reduction methods

Nowadays, scientists are faced with the necessity of exploring high-dimensional data more often than ever since information impacting on life and the evolution of mankind is growing extremely quickly. Dimensionality reduction is an important operation for dealing with multidimensional data. Its primary goal is to obtain compact representations of the original data that are essential for higher-level analysis, while eliminating unimportant or noisy factors otherwise hiding meaningful relationships and correlations. As a result of dimensionality reduction, the subsequent operations on the data will be faster and will use less memory. Moreover, dimensionality reduction is necessary for successful pattern recognition in order to curb the effect known as the curse of dimensionality when the dimension of the data is much larger than the limited number of patterns in a training set. This effect manifests itself in decreasing the recognition accuracy as the dimensionality grows (for details see Section 2.1). On the other hand, a part of the information is lost during dimensionality reduction, hence it is important for the resulting low-dimensional data to preserve a meaningful structure and the relationships of the original high-dimensional space.

For illustration of situations where dimensionality reduction is useful, imagine an experiment where a camera is fixed while a 3D object is rotated in one plane, that is, there is exactly one degree of freedom. When each image is of $n \times n$ pixels, it means n^2 different dimensions if pixel values are chosen as features. On the other hand, the images can be considered as points forming a one-dimensional manifold embedded in an n^2-dimensional space. The only dimension of this manifold is an angle of object rotation, and this dimension, being intrinsic, is sufficient for object characterization, whereas other dimensions might be safely ignored.

From a geometrical point of view, the dimensionality reduction can be formulated as discovering a low-dimensional embedding of high-dimensional data assumed to lie on a linear or nonlinear manifold.

2.1 The curse of dimensionality

The curse of dimensionality was introduced by Richard Bellman in his studies of adaptive control processes (Bellman 1961). This term refers to the problems associated with fitting models, estimating parameters, optimizing a multidimensional function, or multivariate data analysis as the data dimensionality increases.

To interpret the curse of dimensionality from a geometrical point of view, let $\{\boldsymbol{x}_i\}, i = 1,\ldots,N$ denote a set of D-dimensional data points that are given as an input for data analysis. The sampling density is proportional to $N^{1/D}$. Let a function $f(\boldsymbol{x})$ represent a surface embedded into the D-dimensional input space. This surface passes near the data points $\{\boldsymbol{x}_i\}, i = 1,\ldots,N$. If the function $f(\boldsymbol{x})$ is arbitrarily complex and completely unknown (for the most part), one needs dense data points to learn it well. Unfortunately, it is very difficult, almost impossible, to find dense high-dimensional data points, hence the curse of dimensionality. In particular, there is an exponential growth in complexity as a result of an increase in dimensionality, which in turn leads to the deterioration of the space-filling properties for uniformly randomly distributed points in higher-dimensional space (Haykin 1999). In Friedman (1995) the reason for the curse of dimensionality is formulated as follows: "A function defined in high-dimensional space is likely to be much more complex than a function defined in a lower-dimensional space, and those complications are harder to discern."

In practice, the curse of dimensionality means that:

- For a given number of data points, there is a maximum number of features above which the performance of a data analysis will degrade rather than improve (Figure 1).
- For a particular number of features there should be enough data points to make a good data analysis. In Jain & Chandrasekaran (1982) it was suggested that using at least ten times as many points per a data class as the number of features: $N/D > 10$ is a good practice to follow in classification design.

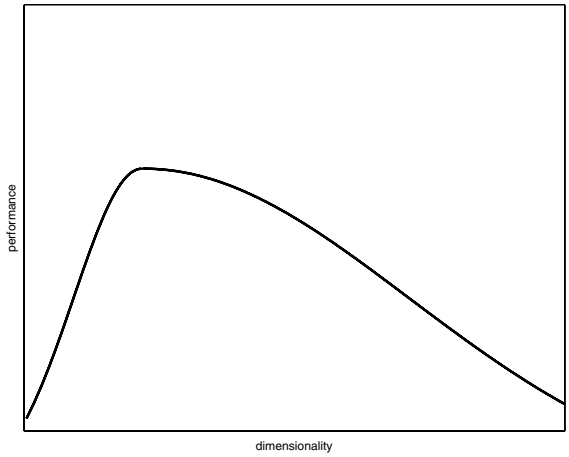


Fig. 1. **Dimensionality versus performance of a data analysis.**

In order to overcome the curse of dimensionality one of the following possibilities can be used: 1) incorporating prior knowledge about the function over and above the data points, 2) providing increasing smoothness of the target function, and 3) reducing the dimensionality of the data. The last approach is of prime interest in this thesis.

2.2 Dimensionality reduction

As was mentioned at the beginning of this chapter, there are two main reasons for obtaining a low-dimensional representation of high-dimensional data: reducing measurement cost of further data analysis and beating the curse of dimensionality. The dimensionality reduction can be achieved either by feature extraction or feature selection. It is important to discriminate these terms: feature extraction refers to algorithms that create a set of new features based on transformations and/or combinations of the original features, while feature selection methods select the most representative and relevant subset from the original feature set (Jain *et al.* 2000). The latter method obtains features possessing a physical meaning, whereas, if one is given the same number of features, the former method can provide better results in further data analysis.

2.2.1 Feature extraction

Feature extraction methods obtain a low-dimensional representation (either in a linear or nonlinear way) of high-dimensional data, i.e., $\mathbb{R}^D \rightarrow \mathbb{R}^d, d << D$. There are many linear algorithms for feature extraction: principal component analysis, singular value decomposition, latent semantic indexing, projection pursuit, independent component analysis, locality preserving projection, linear discriminant analysis etc.

A well-known linear feature extractor is the principal component analysis (PCA) algorithm, which is also known as Hotelling or the Karhunen-Loéve transform (Hotelling 1933, Jolliffe 1989). First, for N D-dimensional data points, PCA computes the d largest eigenvectors of the $D \times D$ data covariance matrix ($d << D$). Then, the corresponding low-dimensional space of dimensionality d is found by linear transformation: $\mathbf{Y}=\mathbf{UX}$, where $\mathbf{X}$ is the given $D \times N$ data matrix, $\mathbf{U}$ is the $d \times D$ matrix of linear transformation composed of the d largest eigenvectors of the covariance matrix, and $\mathbf{Y}$ is the $d \times N$ matrix of the projected data. PCA is an optimal linear dimension reduction technique in the mean-square sense, i.e., it minimizes the errors in reconstruction of the original data from its low-dimensional representation (Jolliffe 1989). The computational complexity of PCA is high, $(O(D^2N) + O(D^3))$, since it uses the most expensive features - eigenvectors associated with the largest eigenvalues. There are few computationally less expensive methods for solving the eigenvector problem in the literature, e.g., $(O(D^2N) + O(D^2))$ in (Wilkinson 1965) and $O(dDN)$ in (Roweis 1997). Moreover, the computation complexity of PCA is low in the case of $D > N$ and also for some situations where the solution is known in a closed form (e.g., for a general circulant matrix).

Singular value decomposition (SVD) (Golub & Van Loan 1989) is a dimensionality

reduction method that is closely related to PCA. First, it searches for the decomposition of the data matrix: $\mathbf{X} = \mathbf{FSV}^T$, where the orthogonal matrices $\mathbf{F}$ and $\mathbf{V}$ contain the left and right singular vectors of $\mathbf{X}$, respectively, and the diagonal of $\mathbf{S}$ contains the singular values of $\mathbf{X}$. Then, the low-dimensional data representation is found by projecting the original data onto the space spanned by the left singular vectors corresponding to the d largest singular values $\mathbf{F}_d$: $\mathbf{Y} = \mathbf{F}_d^T\mathbf{X}$. Like PCA, SVD is also computationally expensive. In practice, it is better to use SVD than PCA for sparse data matrices, since computational complexity of SVD is of the order $O(DcN)$, where c is the number of nonzero entries per column of data matrix (Berry 1992).

A technique known as latent semantic indexing (LSI) (Berry *et al.* 1995) addresses a dimensionality reduction problem for text document data. Using LSI, the document data is represented in a lower-dimensional topic space, i.e., the documents are characterized by some hidden concepts referred to by the terms. LSI can be computed either by PCA or by SVD of the data matrix of N D-dimensional document vectors (Bingham & Mannila 2001).

Contrary to PCA that captures only the second-order statistics (covariance) of the data, methods like projection pursuit (PP) (Friedman & Tukey 1974, Friedman 1987) and independent component analysis (ICA) (Comon 1994) can estimate higher-order statistics; therefore, they are more appropriate for non-Gaussian data distribution (Jain *et al.* 2000). PP uses interpoint distances in addition to the variance of the data, and optimizes a certain objective function called the projection index, in order to pursue optimal projections. Moreover, the PP procedure tends to identify outliers of the data because they exert influence on the sample covariance matrix. The complexity of this algorithm is $O(N \log N)$. In the general formulation, ICA can be considered a variant of projection pursuit (Hyvärinen & Oja 2000). The goal of ICA is to reconstruct independent components, given only linear mixtures of these components and the number of original sources (Bell & Sejnowski 1995). In other words, ICA describes how the observed data is generated by a process of mixing independent components (original sources) $\tilde{\mathbf{S}} = \{\tilde{s}_i\}, i = 1, \ldots, d$, which cannot be directly observed: $\mathbf{X} = \mathbf{PT}$, where $\mathbf{X}$ is a $D \times N$ data matrix, estimates of mixing matrix $\mathbf{P}$ and source matrix $\mathbf{T}$ are unknown matrices of size $D \times d$ and $d \times N$, correspondingly. The time complexity of ICA is $O(d^4 + d^2 N \log_2 N)$ (Boscolo *et al.* 2004).

Recently locality preserving projection (LPP) was proposed in (He & Niyogi 2004). This builds a graph incorporating neighborhood information of the data set and computes a transformation matrix by using a notion of the Laplacian of the graph. The transformation matrix is then used to map the data points into the corresponding low-dimensional space. The obtained linear transformation preserves local neighborhood information of the high-dimensional data in the low-dimensional space. The locality preserving property of LPP is of particular use in information retrieval applications where a nearest neighbor search is faster in a low-dimensional space of the data. The complexity of LPP is $(2O(DN^2) + O(dN^2))$.

All methods described above belong to unsupervised feature extraction algorithms, which do not require data label information to be given as an input. In its turn, the supervised approach is mainly built on using the labels of the data while reducing its dimensionality. Linear discriminant analysis (LDA) is a supervised feature extraction algorithm (Belhumeur *et al.* 1997, He *et al.* 2005), which assumes that data classes are Gaussian with equal covariance structure. LDA searches for the projections for which the data

points belonging to different classes are as far from each other as possible, while requiring the data points of the same class to be close to each other. Unlike PCA, which encodes information in an orthogonal linear space, LDA finds discriminating information in a linearly separable space using bases that are not necessarily orthogonal. Since LDA uses label information, one would expect that it works better than unsupervised techniques. However, in (Martinez & Kak 2001) it has been shown that PCA can outperform LDA when the training data contains a small number of points. The complexity of LDA is the same as for PCA (Kim & Kittler 2005).

Another linear supervised feature extraction algorithm is called the orthogonal centroid method (OCM) (Park *et al.* 2003), which is based on QR decomposition[1]. OCM aims to maximize the between-class distances by performing orthogonalization of the class centroids. The complexity of this method is $O(DNd)$.

Even PCA can perform a nonlinear feature extraction using nonlinearity introduced with kernel functions, hence the name of the algorithm - kernel PCA (Haykin 1999, Jain *et al.* 2000). The main principle of this algorithm (and other kernelized methods) is based on mapping data into a new feature space of higher dimensionality than the original one via a nonlinear function $\phi(\cdot)$, where the PCA algorithm is performed. Usually, Mercer kernels are used in the kernel PCA, since they facilitate a computation of the mapping function, i.e., it is described implicitly via function $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$, which can be represented as a dot product: $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$. The time complexity of the computation increases only by a multiplicative factor corresponding to the computation of the kernel.

Multidimensional scaling (MDS) (Kruskal & Wish 1978) is a feature extraction technique, which discovers the underlying structure of a data set by preserving similarity information among them in the projected space. Different kinds of stress functions may be used to measure the performance of the MDS mapping (Cox & Cox 1994). In practice, the popular criteria are the stress functions proposed by Sammon (Sammon 1969) and Niemann (Niemann 1980). A disadvantage of MDS is that it is impossible to obtain an explicit mapping function; therefore, the MDS mapping should be recalculated each time new data points arrive. To overcome this problem, the use of techniques ranging from linear interpolation to neural networks have been proposed in the literature (Gower 1968, Webb 1995, De Ridder & Duin 1997). The time complexity of MDS is $O(N^2)$. In Morrison *et al.* (2003) the time complexity was reduced to $O(N\sqrt{N})$.

Isometric feature mapping (Isomap) (Tenenbaum *et al.* 2000) and curvilinear distance analysis (CDA) (Lee *et al.* 2000) are two methods that are closely related to linear and nonlinear variants of MDS, respectively. Isomap finds a low-dimensional representation of the input data by calculating geodesic distances and applying the classical MDS to them. As a result of a non-Euclidean metric use, a nonlinear projection is obtained. The CDA algorithm shares the same metric as Isomap, but uses the distances found in different way: it preserves distances by using an energy function, which is minimized by using gradient descent or stochastic gradient descent methods. Isomap and CDA help to avoid noise influence in the low-dimensional representation of large data, while keeping reasonable computation complexities that are about $O(N^3)$.

Locally linear embedding (LLE) (Roweis & Saul 2000) is another nonlinear feature extraction algorithm. The main assumption behind LLE is that the data set is sampled from a

[1]Given a matrix $\mathbf{A}$, its QR decomposition is a matrix decomposition of the form $\mathbf{A} = \mathbf{QR}$, where $\mathbf{R}$ is an upper triangular matrix and $\mathbf{Q}$ is an orthogonal matrix.

manifold embedded in the high-dimensional space. LLE is an unsupervised non-iterative method, which avoids the local minima problems plaguing many competing methods (e.g., those based on the expectation maximization (EM) algorithm). The goal of LLE is to preserve a local structure of the data, that is, to map close points in the original space into close points in the embedded space. It is also assumed that a manifold can be covered by a set of (possibly overlapping) patches, each of which, in turn, can be approximated by a linear hyperplane. Thus, each datum viewed as a point in a high-dimensional space can be represented as a weighted linear combination of its nearest neighbors, and coefficients of this approximation characterize local geometries in the high-dimensional space. These coefficients are then used to find a low-dimensional embedding best preserving these geometries in the low-dimensional space. This space is spanned by several bottom eigenvectors of a sparse matrix. The time complexity of LLE in the worst case is $O(DN^2) + O(DNK^3) + O(dN^2)$.

Nowadays, neural networks (NNs) (Haykin 1999) are widely used for feature extraction. For example, a feed-forward neural network with hidden layers can produce a number of new feature sets, since each hidden layer may be interpreted as a set of linear/nonlinear features (Lowe & Webb 1991). Moreover, neural networks, using a learning algorithm and appropriate architecture, can implement a large number of feature extraction algorithms (Haykin 1999). As a result, these neural networks possess the following advantages over classical feature extraction algorithms (Mao & Jain 1995): 1) most learning algorithms and neural networks are adaptive in nature, thus they are well-suited for many real environments where adaptive systems are required; 2) for real-time implementation, neural networks provide good architectures, which can be easily implemented using current technologies; 3) neural network implementations can overcome the drawbacks inherent in the classical algorithms, e.g., the generalization ability of projecting new data. Unfortunately, in order to build NN one has to define its structure such as architecture of NN (single-layer feedforward, multilayer feedforward, recurrent); the number of hidden layers in the NN; the number of units in the hidden layers; type of activation function (threshold, piecewise-linear, sigmoid etc.). Moreover, many parameters of NN have to be tuned during the training process, which is time-consuming. Therefore, the time complexity of training neural networks depends on many factors (the number of input features, hidden layers, weights, connections etc) and varies for different learning algorithms.

A neural network called the self-organizing map (SOM) or Kohonen map can also be used for nonlinear feature extraction (Kohonen 1997). The main goal of SOM is to transform a high-dimensional pattern into a low-dimensional discrete map in a topologically ordered fashion (Haykin 1999). In SOM, the neurons are placed at the nodes of an m-dimensional grid (lattice), where m is usually equal to 1 or 2. Higher-dimensional maps are also possible but not common. Each neuron of the map is connected to all elements of an input feature vector. The weight vector formed by the weights on the connections for each neuron is of the same dimensionality as the input data. At the first stage, the weight vectors of the SOM are initialized. Then, three essential processes, called competition, cooperation, and synaptic adaptation, build a topology preserving map. The disadvantage of SOM is that the number of parameters required to describe the mapping from latent variables to observations grows exponentially with the dimension of the latent space. The conventional time complexity of SOM is about $O(N^2)$.

While the SOM algorithm provides an unsupervised approach to building a topology

preserving map, learning vector quantization (LVQ) is a supervised learning technique that uses class information to exploit the underlying structure of input data for the purpose of data compression (Kohonen 1990). LVQ does not construct a topographical ordering of the data set, since there is no concept of explicit neighborhood in LVQ as there is in the SOM algorithm. Instead, LVQ approximates the distribution of a class using a reduced number of codebook vectors, while minimizing classification errors. The time complexity of LVQ is the same as for SOM.

Table 1 summarizes the feature extraction methods described above.

Table 1. Summary of feature extraction methods.

Method	Characteristics
Principal component analysis (PCA)	Unsupervised linear mapping based on eigenvectors search; good for Gaussian data.
Singular value decomposition (SVD)	Unsupervised eigenvector based linear mapping; better to use for sparse matrices; good for Gaussian data.
Latent semantic indexing (LSI)	Unsupervised linear mapping designed for text document data; based on the PCA or SVD computational schemes.
Projection pursuit (PP)	Unsupervised linear mapping; suitable for non-Gaussian data distribution; capable of dealing with outliers.
Independent component analysis (ICA)	Unsupervised linear reconstruction of independent components from their mixture; uses more information than just second-order statistics.
Locality preserving projection (LPP)	Unsupervised linear graph based mapping; relatively insensitive to outliers and noise; uses more information than just second-order statistics.
Linear discriminant analysis (LDA)	Supervised eigenvector based linear mapping; the performance can be degraded for non-Gaussian data.
Orthogonal centroid method (OCM)	Supervised QE decomposition based linear mapping.
Kernel PCA	Unsupervised nonlinear mapping; eigenvector based; a kernel is used to replace inner product of data vectors.
Multidimensional scaling (MDS), Sammon's, and Niemann's projections	Unsupervised nonlinear mapping; noise sensitive; difficult to generalize to new data.
Isometric feature mapping (Isomap)	Unsupervised eigenvector-based nonlinear mapping; insensitive to noise and outliers; closely related to MDS.
Curvilinear distance analysis (CDA)	Unsupervised nonlinear mapping; minimizes an energy function by a gradient descent method.
Locally linear embedding (LLE)	Unsupervised eigenvector-based nonlinear mapping preserving local data structure.
Neural networks (NNs)	A number of feature extraction algorithms can be implemented by using NNs trained with a learning algorithm and possessing an appropriate architecture; good generalization ability.
Self-organizing map (SOM)	Nonlinear unsupervised iterative mapping; based on grid of neurons in the feature space.
Learning vector quantization (LVQ)	Nonlinear supervised mapping.

2.2.2 Feature selection

Feature selection methods select the most valuable features from a given set of candidate features. In other words, given a set of D features, the task of a feature selector is to chose a subset of size d that optimizes a particular evaluation criterion. The evaluation criteria depends on the characteristics of the features and specific objectives of further data analysis, e.g., classification, visualization etc. A number of the evaluation criteria have been proposed in the literature: gain-entropy (Quinlan 1986), relevance (Baim 1988), contingency table analysis (Zar 1996), χ^2- and T-statistics (Liu *et al.* 2002), correlation based (Hall 1998) etc.

Assume that $\mathscr{X}$ is the given set of original features with cardinality D, $\mathscr{Y}$ is the selected subset of d features, $\mathscr{Y} \subseteq \mathscr{X}$, and $J(\mathscr{Y})$ is the feature selection criterion for the subset $\mathscr{Y}$. Without loss of generality, let us assume that a higher value of $J(\mathscr{Y})$ indicates a better feature subset. Hence, the problem of feature selection is to find a subset $\mathscr{Y} \subseteq \mathscr{X}$ such that $\mid \mathscr{Y} \mid = d$ and $J(\mathscr{Y}) = \max_{\mathscr{Z} \subseteq \mathscr{X}, \mid \mathscr{Z} \mid = d} J(\mathscr{Z})$. The most straightforward, so-called exhaustive, approach to the feature selection problem contains the following steps (Jain & Zongker 1997, Jain *et al.* 2000): 1) examine all $\left(\begin{array}{c} D \\ d \end{array} \right)$ possible subsets of size d selected from the subset $\mathscr{X}$, and 2) choose the subset leading to the largest value $J(\cdot)$. Here, the number of all possible subsets grows exponentially, making the exhaustive feature selection impractical even for moderate values of D and d. In Cover & Van Campenhout (1977), the authors showed that no nonexhaustive sequential feature selector can certainly produce the optimal subset. They also showed that any ordering of the error probabilities of each of the 2^D feature subsets is possible. Hence, in order to guarantee the optimality of, for example, a 14-dimensional feature subset out of 32 available features, one needs to evaluate about 471.4 million possible subsets. In order to avoid the enormous calculations of the exhaustive method, the complexity of which is $O(D^d)$, a number of procedures have been proposed in the literature: greedy optimization, branch-and-bound, tabu search, simulated annealing, Gibbs sampling, evolutionary programming, genetic algorithms, ant colony optimization, particle swarm optimization, floating search, best-first search, node pruning etc.

The branch-and-bound feature selection algorithm (Narendra & Fukunaga 1977) can obtain the optimal subset of features much faster than the exhaustive search by using intermediate results for obtaining bounds on the final criteria. In other words, this algorithm performs the exhaustive search in an orderly fashion but stops the search along a particular branch if some limit or bound is exceeded, or if the sub-solution does not look very promising. The *indispensable* requirement used in the branch-and-bound algorithm is that the feature selection criterion function $J(\cdot)$ should be monotone, i.e., given two feature subsets $\mathscr{Y}_1, \mathscr{Y}_2 \subseteq \mathscr{X}$ such that $\mathscr{Y}_1 \subseteq \mathscr{Y}_2$, then $J(\mathscr{Y}_1) \leq J(\mathscr{Y}_2)$. The monotonicity property requires growing of the criterion function whenever a feature is added to the initial feature subset. Therefore, the algorithm is guaranteed to obtain an optimal solution only for criterion functions satisfying the monotonicity property. Unfortunately, this is not always the case, since adding a new feature might lead to decreasing of the value of the function. The exact complexity of the algorithm depends on the criterion function and the number of features, however, in the worst case it is as high as that of an exhaustive search, which is impractical for very large feature sets (Jain & Zongker 1997).

Other feature selection algorithms called floating search algorithms begin with a single feature subset and iteratively add or remove features until a termination criterion is met (Pudil *et al.* 1994). These algorithms can be divided into two groups (Jain & Zongker 1997): those that take an empty set and obtain a resulting feature subset by adding features (forward selection) and those that start with a complete set of features and delete them in order to find a result (backward selection). The disadvantage of these methods is that if the feature is retained (deleted), it cannot be discarded from (brought back into) the resulting subset. This is called the nesting problem. In Somol *et al.* (1999) the authors proposed an adaptive version of the sequential floating forward search algorithm that obtains better performance than a classical floating search, but it also does not guarante obtaining an optimal solution. Another possibility to overcome the nesting problem is to use stepwise forward-backward selection that combines a forward and backward selection strategy (Stearns 1976). The time complexity of the floating search is $O(2^D)$.

Monte Carlo algorithms are stochastic and always terminate with an answer to a problem, but this answer may occasionally be incorrect. A fundamental aspect of Monte Carlo algorithms is that the probability of the incorrect answer can be lowered by repeated runs and a majority voting strategy (Esposito 2003). Genetic algorithms belong to the class of Monte Carlo methods and have been widely used for feature selection (Siedlecki & Sklansky 1988, Brill *et al.* 1992, Yang & Honavar 1998, Raymer *et al.* 2000). The genetic algorithm is biologically inspired and can imitate natural evolution (Holland 1992). Basically, it represents a given feature subset as a binary string, a so-called chromosome. The i^{th} position in the chromosome is either zero or one, and denotes the presence or absence of the i^{th} feature in the set. The length of the binary string denotes the total number of available features. Three genetic operators: 1) selection, 2) crossover, where parts of two different parent chromosomes are mixed to create offsprings, and 3) mutation, where the bits of a single parent are perturbed to create a child, are applied to the set of chromosomes to make them evolve and so create a new generation (Jain & Zongker 1997). After several generations, the genetic algorithm converges to a solution represented by the best chromosome (Goldberg 1989). In Oh *et al.* (2004), it has been shown that performance of the genetic algorithm is almost the same as those of the sequential floating forward search algorithm, and it is difficult to say which of these algorithms is better. The complexities of the algorithms are different and depend on the particular problem. The computational scheme for estimating complexities is given in (Rylander & Foster 2001).

Another approach to feature selection is based on the examination of the parameters of a trained classifier, e.g., a trained neural network. As in Mao *et al.* (1994), the authors use a multilayer feed forward network with a back propagation learning algorithm for data classification (Rumelhart *et al.* 1986). They define a *node saliency* measure, which is used for pruning the least salient nodes. As a result, the corresponding features are removed from the feature set. Actually, the node pruning feature selection method obtains both the optimal feature set and optimum classifier by training a network and removing the least salient node of input or hidden layer. The reduced network is trained again, followed by removal of the least salient node. These two steps are repeated until the desired tradeoff between the size of the network and classification error is achieved (Jain & Zongker 1997). The complexity of this algorithm depends on many characteristics of NNs.

The last method belongs to the wrapper approach of the feature selection, where the features are chosen in order to improve the performance of the particular machine learning

algorithm (classifier, clustering, visualization etc). Whereas other methods represent a filter paradigm, where the goal is to choose features based on a general characteristic of the data rather than a learning algorithm to evaluate the merit of feature subset. As a consequence, filter methods are generally faster than wrapper methods, and, as such, are more practical for use on high-dimensional data (Hall 1998).

A summary of the feature selection methods described above is given in Table 2.

Table 2. Summary of feature selection methods.

Method	Characteristics
Exhaustive search	Evaluates all $\begin{pmatrix} D \\ d \end{pmatrix}$ possible subsets; guaranteed to find the optimum solution; impractical for large data sets.
Branch-and-bound search	Explores only a fraction of all possible subsets; guaranteed to find the optimum solution if it uses the criterion function possessing the monotonicity property.
Floating search	Begins with a single empty / complete feature set and iteratively adds / removes features until a termination criterion is met; once a feature is retained (deleted), it cannot be discarded from (brought back into) the resulting subset.
Monte Carlo algorithms	Stochastic; always converges to a solution, which is not necessarily optimal.
Genetic algorithms	Biologically inspired; iterative; converges to the suboptimum solution.
Node pruning	Based on the examination of the parameters of a trained neural network; iterative; obtains the suboptimal feature subset and classifier.

3 The locally linear embedding algorithm and its extensions

The locally linear embedding (LLE) algorithm has been recently proposed as a powerful eigenvector method for the problem of nonlinear dimensionality reduction (Roweis & Saul 2000). The key assumption related to LLE is that, even if the manifold embedded in a high-dimensional space is nonlinear when considered as a whole, it still can be assumed to be locally linear if each data point and its neighbors lie on or close to a locally linear patch of the manifold. That is, the manifold can be covered with a set of locally linear (possibly overlapping) patches which, when analyzed together, can yield information about the global geometry of the manifold.

Because of the assumption that local patches are linear, i.e., each of them can be approximated by a linear hyperplane, each data point can be represented by a weighted linear combination of its nearest neighbors (different ways can be used in defining the neighborhood). Coefficients of this approximation characterize local geometries in a high-dimensional space, and they are then used to find low-dimensional embeddings preserving the geometries in a low-dimensional space. The main point in replacing the nonlinear manifold with the linear hyperplanes is that this operation does not bring significant error, because, when locally analyzed, the curvature of the manifold is not large, i.e., the manifold can be considered to be locally flat. The result of LLE is a single global coordinate system.

Consider an illustrative example of the nonlinear dimensionality reduction problem, which is demonstrated by the two-dimensional manifold in Figure 2. In this example, the linear and nonlinear methods, called principal component analysis (PCA) and LLE, respectively, are applied to the data in order to discover the true structure of the manifolds. Figures 2 (b) and (c) corresponding to the two-dimensional PCA and LLE projections, allow us to conclude that LLE succeeds in recovering the underlying manifolds whereas, PCA creates local and global distortions by mapping faraway points to nearby points in the planes.

During the last four years, many LLE based algorithm have been developed and introduced in the literature: original LLE (Roweis & Saul 2000), kernelized LLE (KLLE) (DeCoste 2001), Laplacian Eigenmap (LEM) (Belkin & Niyogi 2003) and Hessian LLE (HLLE) (Donoho & Grimes 2003). This chapter describes LLE and the number of extensions to this algorithm that enhance its performance.

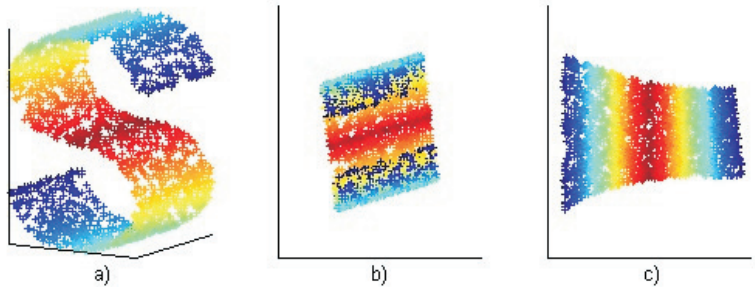


Fig. 2. Nonlinear dimensionality reduction problem: a) initial nonlinear data manifold; b) result obtained with linear method (PCA); c) result obtained with nonlinear method (LLE).

3.1 The conventional locally linear embedding algorithm

As input, LLE takes a set of N vectors, each of dimensionality D, sampled from a smooth underlying manifold. Suppose that the manifold is well-sampled, i.e., each data point and its neighbors lie on or close to a locally linear patch of the manifold. More precisely, by "smooth" and "well-sampled" one means that for data sampled from a d-dimensional manifold, the curvature and sampling density are such that each data point has in the order of $2d$ neighbors which define a roughly linear patch on the manifold with respect to some metric in the input space (Saul & Roweis 2003). The input vectors (each of which may represent an image, for example) are assembled in a matrix $\mathbf{X} = \{\boldsymbol{x}_i\}, i = 1,\ldots,N$ of size $D \times N$. LLE finds and outputs another set of N d-dimensional vectors ($d << D$) assembled in a matrix $\mathbf{Y} = \{\boldsymbol{y}_i\}, i = 1,\ldots,N$ of size $d \times N$. As a result, the i^{th} column vector of $\mathbf{Y}$ corresponds to the i^{th} column vector of $\mathbf{X}$. Further the term point will stand for a vector either in $\mathbb{R}^D$ or in $\mathbb{R}^d$, depending on the context.

The LLE algorithm consists of three main steps as follows.

Step 1: Finding sets of points constituting local patches in a high-dimensional space

The first step of the LLE algorithm is to find the neighborhood of each data point $\boldsymbol{x}_i$, $i = 1,\ldots,N$. This can be done either by identifying a fixed number of nearest neighbors K per data point in terms of Euclidean distances or by choosing all points within a ball of fixed radius. In the latter case the number of neighbors may vary for each data point. In both cases, one parameter should be specified (a procedure of automatic determination of the optimal number of nearest neighbors is considered in Section 3.2). The number of neighbors can be restricted to be the same within a certain radius, or one can take up to a certain number of neighbors but not outside a maximum radius (Saul & Roweis 2003). Moreover, one can search for the neighbors by utilizing a non-Euclidean metric in a non-Euclidean space, e.g., using kernel distances in the kernel's feature space (DeCoste 2001).

The squared distance Δ between pairs of points $\boldsymbol{x}_i, \boldsymbol{x}_j$ is computed according to Eq. 1:

$$\Delta(\boldsymbol{x}_i,\boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i,\boldsymbol{x}_i) - 2\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) + \kappa(\boldsymbol{x}_j,\boldsymbol{x}_j), \tag{1}$$

where

$$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j). \tag{2}$$

Eq. 2 defines a Mercer kernel (DeCoste 2001), where $\phi(\boldsymbol{x}_i)$ is a mapping function from the original, D-dimensional space into a high, possibly infinite dimensional feature space. This mapping function may be explicitly defined or may be implicitly defined via a kernel function (Srivastava 2004).

When $\phi(\boldsymbol{x}_i) = \boldsymbol{x}_i$ and $\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$ (linear kernel), Δ corresponds to the Euclidean distance. This is the case of the original LLE. Other kernels are possible as well (Smola & Schölkopf 1998).

It should be noted that it may be sometimes appropriate to vary K from point to point, especially if data points are non-uniformly sampled from the manifold. But this may need the analysis of the data distribution which is a complex and data-dependent task. Further, unless otherwise stated, it is assumed that the underlying manifold is sufficiently uniformly sampled so that the neighborhood of each point is properly defined by its K nearest neighbors.

The complexity of the nearest neighbor step scales in the worst case as $O(DN^2)$ if no optimization is done. But for the type of data manifolds considered here (smooth and thin), the nearest neighbors can be computed in $O(N \log N)$ time by constructing ball or K-D trees (Friedman *et al.* 1977, Omohundro 1989, 1991). In Karger & Ruhl (2002), the authors specifically emphasize the problem of computing nearest neighbors for data lying on low-dimensional manifolds. Approximate methods also allow reducing of the time complexity of searching for the nearest neighbors up to $O(\log N)$ with various guarantees of accuracy (Datar *et al.* 2004).

Step 2: Assigning weights to pairs of neighboring points

Having found K nearest neighbors for each data point, the next step is to assign a weight to every pair of neighboring points. These weights form a weight matrix $\mathbf{W} = \{w_{ij}\}, i, j = 1,\ldots,N$, each element of which (w_{ij}) characterizes a degree of closeness of two particular points ($\boldsymbol{x}_i$ and $\boldsymbol{x}_j$).

Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ be neighbors, the weight values w_{ij} are defined as contributions to the reconstruction of the given point from its nearest neighbors. In other words, the following minimization task must be solved (Roweis & Saul 2000):

$$\varepsilon(\mathbf{W}) = \sum_{i=1}^{N} \| \boldsymbol{x}_i - \sum_{j=1}^{N} w_{ij} \boldsymbol{x}_j \|^2, \tag{3}$$

subject to constraints $w_{ij} = 0$, if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are not neighbors, and $\sum_{i=1}^{N} w_{ij} = 1$. The first sparseness constraint ensures that each point $\boldsymbol{x}_i$ is reconstructed only from its neighbors, while the second condition enforces translation invariance of $\boldsymbol{x}_i$ and its neighbors. Moreover, as follows from Eq. 3, the constrained weights are invariant to rotation and rescaling, but not to local affine transformation, such as shears.

The optimal reconstruction weights minimizing Eq. 3 can be computed in a closed form (Saul & Roweis 2003). Eq. 3 for a particular data point $\boldsymbol{x}_i$ with its nearest neighbors $\boldsymbol{x}_j$ and reconstruction weights w_{ij} that sum to one can be rewritten as follows:

$$\varepsilon(\boldsymbol{w}_i) = \| \boldsymbol{x}_i - \sum_{j=1}^{N} w_{ij} \boldsymbol{x}_j \|^2 = \| \sum_{j=1}^{N} w_{ij} (\boldsymbol{x}_i - \boldsymbol{x}_j) \|^2 = \sum_{j,k=1}^{N} w_{ij} w_{ik} g_{jk}, \tag{4}$$

where $\mathbf{G} = \{g_{ij}\}$, $i, j = 1, \ldots, N$ is the local Gram matrix with elements defined by Eq. 5:

$$g_{jk} = (\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_i - \boldsymbol{x}_k), \tag{5}$$

where $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ are neighbors of $\boldsymbol{x}_i$.

The case for $\phi(\boldsymbol{x}_i) = \boldsymbol{x}_i$ is considered in Eq. 5, i.e., the reconstruction weights are calculated for the original feature space; otherwise, Eq. 6 defines the kernel Gram matrix:

$$\begin{aligned} g_{jk} &= (\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)) \cdot (\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_k)) = \\ &= \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) - \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) - \kappa(\boldsymbol{x}_i, \boldsymbol{x}_k) + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_k), \end{aligned} \tag{6}$$

where $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ are neighbors of $\boldsymbol{x}_i$.

In terms of the inverse Gram matrix, the optimal reconstruction weights are given by:

$$w_{ij} = \frac{\sum_{k=1}^{N} g_{jk}^{-1}}{\sum_{l,m=1}^{N} g_{lm}^{-1}}, \tag{7}$$

where g_{ij}^{-1} is an element of the inverse Gram matrix $\mathbf{G}^{-1}$, which is a time consuming task in practice. Therefore, in order to facilitate the second step, the reconstruction weights can be found by solving the following system of linear equations:

$$\sum_{k=1}^{N} g_{jk} w_{ik} = 1. \tag{8}$$

When $K > D$, i.e., the original data is itself low-dimensional, then the local reconstruction weights are no longer uniquely defined, since the Gramm matrix in Eq. 8 is singular or nearly singular. To break the degeneracy, a regularization of $\mathbf{G}$ is required, which is done by adding a small positive constant to the diagonal elements of the matrix:

$$g_{jk} \leftarrow g_{jk} + \delta_{jk} \left(\frac{\theta^2}{K} \right) Tr(\mathbf{G}), \tag{9}$$

where $\delta_{jk} = 1$ if $j = k$ and 0 otherwise, $Tr(\mathbf{G})$ denotes the trace of $\mathbf{G}$, and $\theta << 1$. The regularization term acts to penalize large weights that exploit correlations beyond some level of precision in the data sampling process. Moreover, in some cases this penalization may be useful for obtaining projection that is robust to noise and outliers (Saul & Roweis 2003).

The second step of the LLE algorithm is the least expensive one. Searching for optimal weights needs to solve a $K \times K$ set of linear equations for each data point that can be performed in $O(DNK^3)$ time. To store the weight matrix one can consider it as a sparse matrix, since it has only NK non-zero elements.

Step 3: Computing the low-dimensional embedding

The goal of the LLE algorithm is to preserve a local linear structure of a high-dimensional space as accurately as possible in a low-dimensional space. In particular, the weights w_{ij} that reconstruct point $\boldsymbol{x}_i$ in D-dimensional space from its neighbors should reconstruct its projected manifold coordinates in d-dimensional space. Hence, the weights w_{ij} are kept

fixed and embedded coordinates $\boldsymbol{y}_i$ are sought by minimizing the following cost function (Roweis & Saul 2000):

$$\Phi(\mathbf{Y}) = \sum_{i=1}^{N} \| \boldsymbol{y}_i - \sum_{j=1}^{N} w_{ij} \boldsymbol{y}_j \|^2 . \tag{10}$$

To make the problem well-posed, the following constraints are imposed:

$$\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i \boldsymbol{y}_i^T = \mathbf{I}_{d \times d}, \tag{11}$$

$$\sum_{i=1}^{N} \boldsymbol{y}_i = 0_{d \times 1}, \tag{12}$$

where $\mathbf{I}_{d \times d}$ is a $d \times d$ identity matrix and $0_{d \times 1}$ is a zero-vector of length d. The embedded coordinates have to have normalized unit covariance as in Eq. 11 in order to remove the rotational degree of freedom and to fix the scale. Eq. 12 removes the translation degree of freedom by requiring the outputs to be centered at the origin. As a result, a unique solution is obtained.

To find the embedding coordinates minimizing Eq. 10, which are composed in the matrix $\mathbf{Y} = \{\boldsymbol{y}_i\}, i = 1, \ldots, N$ of size $d \times N$, under the constraints given in Eqs. 11, 12, a new matrix is constructed, based on the weight matrix $\mathbf{W}$:

$$\mathbf{M} = (\mathbf{I}_{N \times N} - \mathbf{W})^T (\mathbf{I}_{N \times N} - \mathbf{W}). \tag{13}$$

The cost matrix $\mathbf{M}$ is sparse, symmetrical, and positive semidefinite. LLE then computes the bottom $(d+1)$ eigenvectors of $\mathbf{M}$, associated with the $(d+1)$ smallest eigenvalues. The first eigenvector (composed of 1's) whose eigenvalue is close to zero is excluded. The remaining d eigenvectors yield the final embedding $\mathbf{Y}$. These eigenvectors form rows of the matrix $\mathbf{Y}$ and now, unless otherwise stated, the notation $\boldsymbol{y}_i$ defines the i^{th} point in the embedded space.

Because $\mathbf{M}$ is sparse, eigenvector computation is quite efficient, though for large N's it anyway remains the most time-consuming step of LLE. Computing d bottom eigenvalues without special optimization scales as $O(dN^2)$ (Saul & Roweis 2003). This complexity can be reduced by utilizing one of the specialized methods for sparse, symmetric eigenproblems (Fokkema *et al.* 1998). Alternatively, for very large problems, the embedding cost function can be optimized by the conjugate gradient method (Press *et al.* 1993), iterative partial minimization, Lanczos iterations or stochastic gradient descent (LeCun *et al.* 1998).

Summary of the original LLE

Since we will often refer to the original LLE presented in (Roweis & Saul 2000), here is a brief summary of it:

1. Finding K nearest neighbors for each data point $\boldsymbol{x}_i, i = 1, \ldots, N$.
2. Calculating the weight matrix $\mathbf{W}$ of weights between pairs of neighbors.
3. Constructing the cost matrix $\mathbf{M}$ (based on $\mathbf{W}$) and computing its bottom $(d+1)$ eigenvectors.

A linear kernel (see its definition in the description of Step 1) is used so that nearest neighbors are determined based on the Euclidean distance.

Different LLE-based approaches also leading to an eigenvector problem are presented in (Belkin & Niyogi 2003, Donoho & Grimes 2003). Belkin and Niyogi propose Laplacian eigenmaps: they define the Laplacian operator in tangent coordinates $\Delta^{(tan)}(f) = \sum_{i=1}^{N} \frac{df^2}{du_i^2}$, where f is a smooth function on a manifold and $\boldsymbol{u}_i, i = 1, \ldots, N$ are tangent coordinates of the manifold $\mathscr{M}$, and then define the functional $\mathscr{L}(f) = \int_{\mathscr{M}} \left(\Delta^{(tan)}(f)\right)^2 dm$. On the other hand, Donoho and Grimes propose Hessian eigenmaps where they estimate a quadratic form $\mathscr{H}(f) = \int_{\mathscr{M}} \| H_f(m) \|_F^2 \, dm$, here H_f denotes the Hessian of f calculated by using tangent coordinates of the manifold $\mathscr{M}$. The first functional computes the average of the Laplacian operator over the manifold, while the second one estimates the average of the Frobenius norm of Hessian over the manifold. The embedding coordinates in these cases are found by computing $(d+1)$ smallest eigenvectors of $\mathscr{L}$ and $\mathscr{H}$ and discarding one eigenvector composed of ones that corresponds to the smallest eigenvalue.

3.2 Extensions of the conventional locally linear embedding algorithm

In this section, three extensions of the original LLE are decribed. They are 1) automatic determination of the optimal number of nearest neighbors, 2) estimation of the intrinsic dimensionality of data, and 3) generalization of LLE to new data.

3.2.1 Automatic determination of the optimal number of nearest neighbors

The original LLE has two parameters to be adjusted: the number of nearest neighbors for each data point, K, and the dimensionality of the embedded space, d (intrinsic dimensionality of the data manifold or, equivalently, the minimal number of degrees of freedom needed to generate the original data).

One of the goals of multidimensional data analysis is visualization, which often helps us to see clustering tendencies in underlying data. Visualization means that d is fixed (it is either 1, 2 or 3), so that the only parameter to be estimated is K. The reason for choosing the right K is that a large number of nearest neighbors causes smoothing or eliminating of small-scale structures in the manifold. In contrast, too small neighborhoods can falsely divide the continuous manifold into disjointed sub-manifolds.

In Kouropteva *et al.* (2002b), a procedure for the automatic selection of the optimal value for the parameter K is proposed. The key idea consists of the following: a set of potential candidates to become K_{opt} is first selected without proceeding through all steps of LLE, followed by computing a measure of optimality for each candidate and picking that candidate for which this measure is optimal. As a result, the most time-consuming operation of LLE - eigenvector computation - is carried out only a few times, which leads

to a significant speed-up.

To select the candidates for K_{opt}, let us consider Eq. 3, which defines the reconstruction error $\varepsilon(\mathbf{W})$ resulting from the approximation of small parts of the nonlinear manifold by the linear hyperplanes. This function explicitly depends on weights calculated for data, and implicitly depends on the number of nearest neighbors, since $w_{ij} = 0$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are not neighbors. Therefore, Eq. 3 can be rewritten as follows:

$$\varepsilon(\mathbf{W}) = \sum_{i=1}^{N} \| \boldsymbol{x}_i - \sum_{j=1}^{N} w_{ij} \boldsymbol{x}_j \|^2 = \sum_{i=1}^{N} \| \boldsymbol{x}_i - \sum_{j=1}^{K} w_{ij}^* \boldsymbol{x}_i^j \|^2, \tag{14}$$

where $\{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^K\}$ are the K nearest neighbors of $\boldsymbol{x}_i$ with the corresponding non-zero weights $\{w_{i1}^*, w_{i2}^*, \ldots, w_{iK}^*\}$.

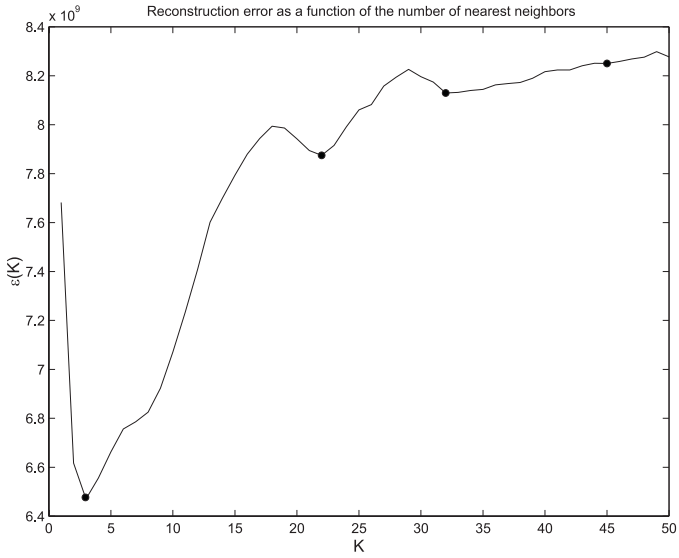
Eq. 14 is chosen as a criterium for selecting the potential candidates for K_{opt} that corresponds to the local and global minima of this function: $S = \{K_1^*, K_2^*, \ldots, K_{N_S}^*\}$, where N_S is data dependent and equals to the number of local and global minima for a data set. Figures 3 (a, b) shows the potential candidates S for face and wood data (for the data sets description, see Appendix 1), correspondingly. The presence of several minima of this function can be explained as follows: the existence of a global minimum for small values of K is caused by the fact that for many points their first few neighbors are all indeed close to them and adding a new neighbor decreases the reconstruction error every time. However, as K grows, this error begins to alternate (it rises and falls), because the Euclidean distance becomes an unreliable indicator of proximity.

After candidates for K_{opt} are selected, it is necessary to choose only one value to be used in LLE. In general, there can be different definitions of optimality. We rely on a quantitative measure called residual variance (Tenenbaum *et al.* 2000), which must be computed for each $K_i^* \in S, i = 1, \ldots, N_S$. This quantitative measure illustrates how well the distance information is preserved. It is defined as $1 - \rho^2_{\mathbf{D}_x\mathbf{D}_y}$, where $\rho_{\mathbf{D}_x\mathbf{D}_y}$ is the standard linear correlation coefficient, taken over all entries of $\mathbf{D}_x$ and $\mathbf{D}_y$, where $\mathbf{D}_x$ and $\mathbf{D}_y$ are the matrices of Euclidean distances (between pairs of points) in the high-dimensional and corresponding low-dimensional spaces, respectively. The lower the residual variance is, the better high-dimensional data is represented in the embedded space. Hence, the optimal value for K, defined as K_{opt}, can be determined as

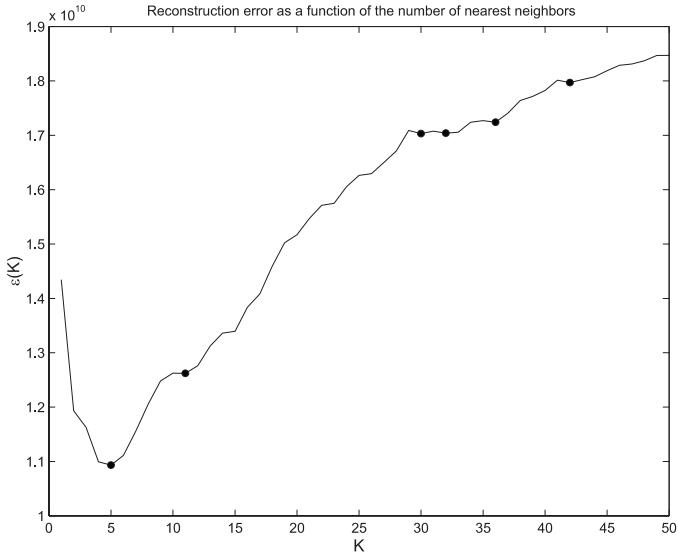
$$K_{opt} = \arg_i \min_{K_i^*} (1 - \rho^2_{\mathbf{D}_x\mathbf{D}_y}). \tag{15}$$

In summary, the automatic hierarchical procedure for finding K_{opt} is as follows:

1. Choose K_{max} - the maximal possible value of K_{opt} (it is problem-dependent, but usually K_{max} does not exceed 50).
2. Calculate $\varepsilon(\mathbf{W})$ for each K, $K \in [1, K_{max}]$ according to Eq. 3. This is done by the first two steps of LLE.
3. Find all minima of $\varepsilon(\mathbf{W})$ and corresponding K's which compose the set S of initial candidates.
4. For each $K^* \in S$, run LLE and compute a quantitative measure.
5. Select K_{opt} according to Eq. 14.



a) $S = \{3, 22, 32, 45\}$.



b) $S = \{5, 11, 30, 32, 36, 42\}$.

Fig. 3. **Reconstruction error ($K \in [1, 50]$) for a) face and b) wood data. Points on the curve corresponding to the local and global minima are marked with filled circles.**

The determination of K_{opt} was successfully applied in (Kouropteva *et al.* 2002b) to the visualization of face and wood surface images (Figures 4, 5). The values found for K are the same, or very close to those obtained with a straightforward method. The essential

difference in the straightforward method is that it needs to solve an eigenproblem for all $K \in [1, K_{max}]$ in order to find K_{opt}. Such an approach, however, is computationally demanding.

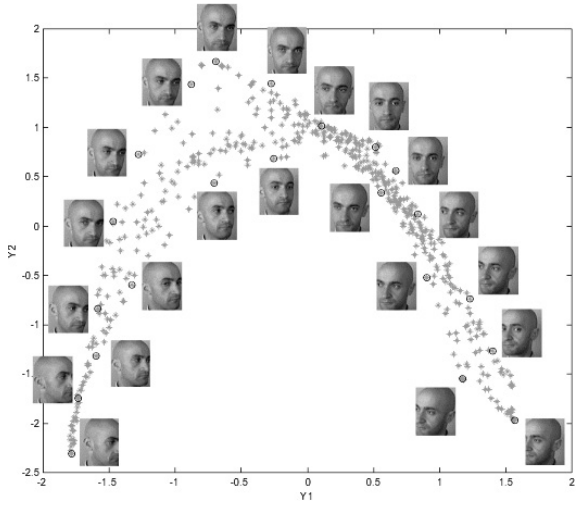


Fig. 4. **Face images mapped into 2D space by LLE using $K_{opt} = 22$. Circles indicate places where the images nearby are mapped into by LLE. The data set has a total $N = 500$ grayscale images at 120×100 resolution $(D = 12,000)$.**

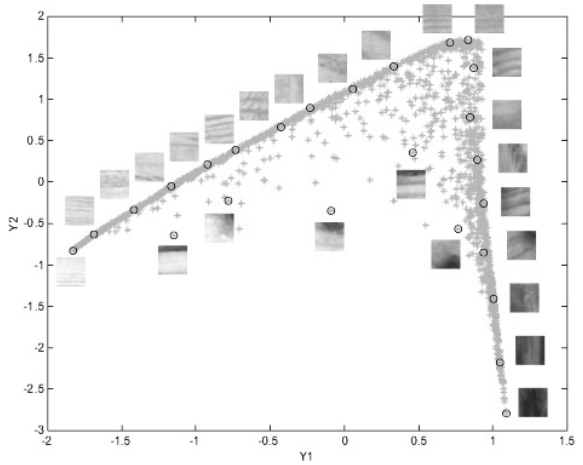


Fig. 5. **Wood images mapped into 2D space by LLE using $K_{opt} = 11$. Circles indicate places where the images nearby are mapped into by LLE. The data set has a total $N = 1,900$ RGB images at 32×32 resolution $(D = 3 \times 32 \times 32 = 3,072)$.**

Tables 3 and 4 list the potential candidates found with the hierarchical method for the face and wood data sets with the corresponding values of residual variance. According to the table $K_{opt} = 22$ for the face data set and $K_{opt} = 11$ for the wood data set. In its turn, K for the face data set obtained with the straightforward method is equal to 23, resulting in the embedded space shown in Figure 6, which looks much the same as that for $K = 22$ (Figure 4). In terms of quantitative criteria, the measures of residual variance for $K = 23$ and $K = 22$ correspond to its first (0.430) and second (0.438) minimal values (Figure 7 (a)). For the wood data set K found with the straightforward method is equal to 11, and coincides with the value obtained with the hierarchical method. Figure 7 (b) demonstrates the residual variance for the wood data with its minimum at $K = 11$. These results demonstrate that the hierarchical method for finding K_{opt} is quite accurate.

Table 3. The potential candidates with the corresponding values of residual variance $(1 - \rho^2_{D_x D_y})$ *found with the hierarchical method for the face data set; the best result is shown in bold and underlined.*

K^*	$(1 - \rho^2_{\mathbf{D}_x \mathbf{D}_y})$
$K^*_1 = 3$	1.000
$K^*_2 = 22$	**0.438**
$K^*_3 = 32$	0.608
$K^*_4 = 45$	0.653

Table 4. The potential candidates with the corresponding values of residual variance $(1 - \rho^2_{D_x D_y})$ *found with the hierarchical method for the wood data set; the best result is shown in bold and underlined.*

K^*	$(1 - \rho^2_{\mathbf{D}_x \mathbf{D}_y})$
$K^*_1 = 5$	0.546
$K^*_2 = 11$	**0.512**
$K^*_3 = 30$	0.692
$K^*_4 = 32$	0.706
$K^*_5 = 36$	0.600
$K^*_6 = 42$	0.693

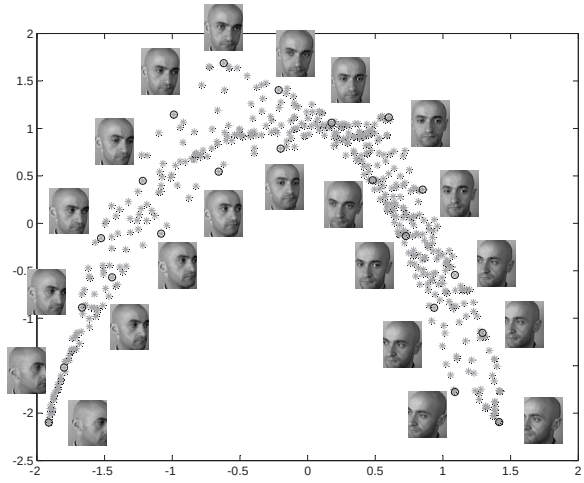


Fig. 6. **Face images mapped into 2D space by LLE using $K = 23$ found by the straightforward method. Circles indicate places where the images nearby are mapped into by LLE. The data set has a total $N = 500$ grayscale images at 120×100 resolution $(D = 12,000)$.**

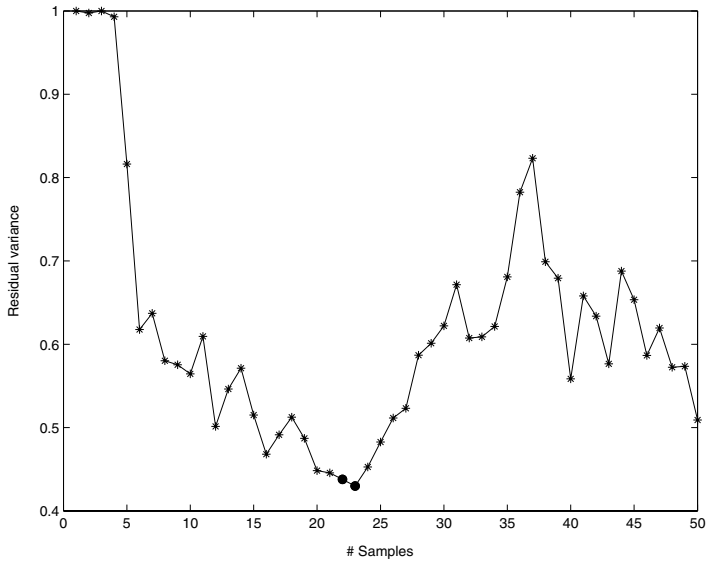
The complexity of the hierarchical method for finding the optimal number of nearest neighbors for individual steps is as follows:

- Finding nearest neighbors - $O(DN^2)$.
- Computing reconstruction weights - $O(K_{max}DNK^3)$.
- Computing bottom eigenvectors - $O(N_S dN^2)$.

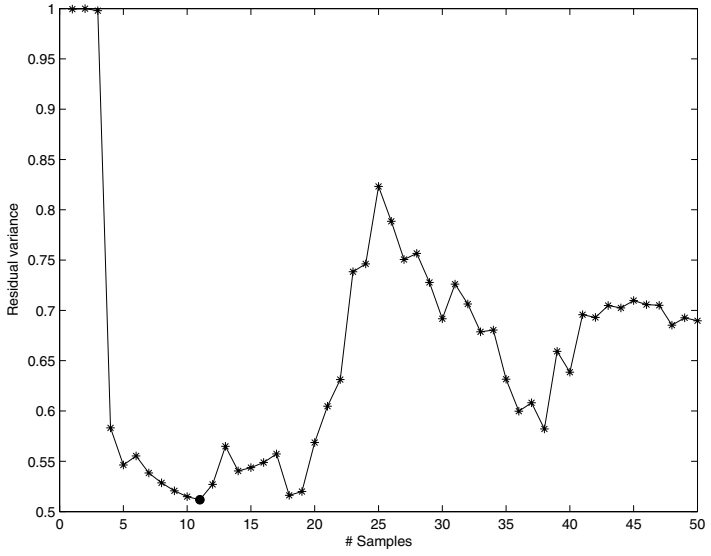
The cost of individual steps for the straightforward method:

- Finding nearest neighbors - $O(DN^2)$.
- Computing reconstruction weights - $O(K_{max}DNK^3)$.
- Computing bottom eigenvectors - $O(K_{max}dN^2)$.

As can be seen, the costs of the two methods depend on the values of N_S and K_{max}. It is clear that the hierarchical method is faster than the straightforward one, since the most expensive step of LLE - the eigenvector computation - is done only N_S times for the hierarchical method, where $N_S << K_{max}$. As a result, a total decreased cost is obtained (see Figure 8). A particular gain is problem dependent. For example, for the wood data it was more than four-fold.



a)



b)

Fig. 7. Residual variance for the a) face and b) wood data sets. Filled circles correspond to the values of K obtained with the hierarchical and straightforward methods.

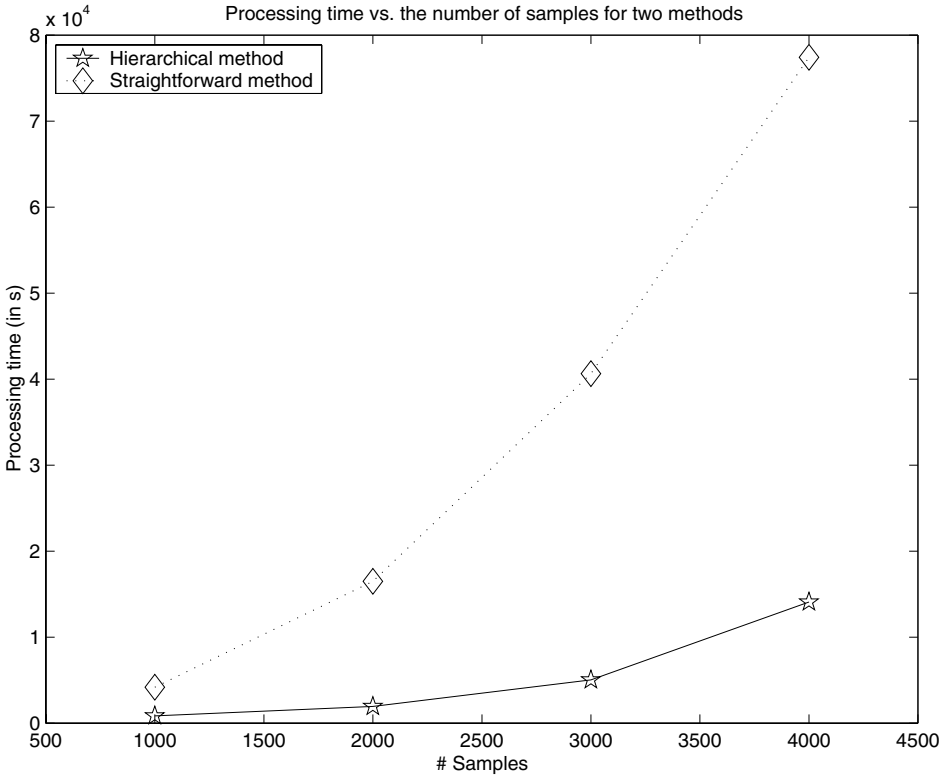


Fig. 8. **Processing time in seconds versus the number of samples (data points), N, for the straightforward and hierarchical methods, respectively. Both methods were run for artificial 3D data, where N varied from 1,000 to 4,000 and K_{max} was equal to 50.**

3.2.2 Estimating the intrinsic dimensionality of data

Intrinsic dimensionality (ID) of data is equal to the number of free parameters needed to represent the data patterns. Having a high-dimensional data set, one might be interested in estimating[1] its intrinsic dimensionality before applying any dimensionality reduction technique that needs the parameter to be exactly specified. The PCA strategy to find the intrinsic dimensionality for linear manifolds is based on computing the number of eigenvalues of the data covariance matrix that are comparable in magnitude to the largest eigenvalue. In Polito & Perona (2002) the authors proposed using a similar strategy by LLE - that is, to estimate d by the number of eigenvalues that are appreciable in magnitude to the second smallest nonzero eigenvalue of the cost matrix $\mathbf{M}$ from Eq. 13, and for which the corresponding eigenvectors have zero-error approximation: $\boldsymbol{y}_i = \sum_{j=1}^{N} w_{ij}\boldsymbol{y}_j$. The main

[1] Here, we are not talking about enforcing the intrinsic dimensionality to 1, 2 or 3 as for visualization.

difference between PCA and LLE is that the former method concentrates on finding eigenvectors corresponding to the largest eigenvalues, while the latter one searches for the bottom eigenvectors. Hence, in the case of LLE, one has to deal with ill-conditioned eigenvalues and eigenvectors (see Appendix 2).

In spite of the ill-conditioning of individual eigenvectors, those corresponding to a cluster of close eigenvalues are better-conditioned together, and they span the invariant subspace (Bai *et al.* 2000). As evidenced by the foregoing, it is natural to estimate the intrinsic dimensionality of the data manifold by the number of the smallest eigenvalues that form a cluster. In Figure 9, the two lowest eigenvalues form the cluster and the intrinsic dimensionality of the data manifold is equal to two.

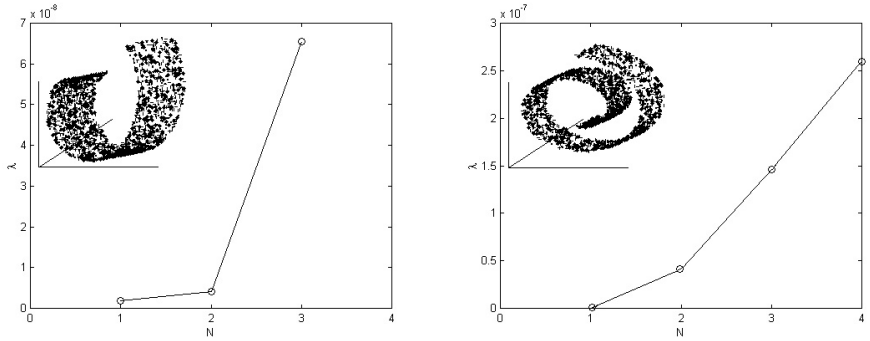


Fig. 9. **Eigenvalues of the cost matrix M for a) c-curve and b) swiss-roll data.**

In Saul & Roweis (2003), it was empirically shown that sometimes this procedure of estimating intrinsic dimensionality does not work for data that has been sampled in a non-uniform way. This situation might occur in a case where the gap between ill-conditioned eigenvalues is large. Hence, the eigenspace obtained with the eigenvectors corresponding to the cluster formed by the closest eigenvalues is no longer invariant, leading to wrong intrinsic dimensionality estimations. Therefore, in order to exactly estimate the intrinsic dimensionality of the data, classical methods designed for this purpose should be used prior to the final step of LLE in order to set the number of embedded coordinates (Pettis *et al.* 1979, Brand 2003, Kégl 2003).

In our experiments, the intrinsic dimensionality was calculated by LLE and packing numbers (Kégl 2003) for ten different data sets with original dimensionality D varying from 3 to 6,300 (for the data sets description, see Appendix 1; for the description of the intrinsic dimensionality estimation algorithm using packing numbers, see Appendix 3). The results coincide for six data sets, while in the other four cases, the intrinsic dimensionalities calculated by LLE are rather larger than those obtained with packing numbers (Table 5). Since the latter algorithm is especially designed for the intrinsic dimensionality estimation, we assume that its outputs are more exact. Due to the fact that overestimation is safer than underestimation, and it does not lead to the loss of important information, one can use the LLE based intrinsic dimensionality estimator for calculating an approx-

imate dimensionality of the data manifold[2]. Moreover, having found the LLE based intrinsic dimensionality estimation of the data, the resulting LLE embedding is obtained by searching the eigenvectors corresponding to the smallest eigenvalues, which does not involve additional computations.

Table 5. Ten data sets with corresponding original and estimated intrinsic dimensionalities.

Data	D	Packing numbers ID	LLE ID
c-curve	3	2	2
s-curve	3	2	2
iris	4	2	2
liver	6	3	3
wine	13	3	3
vehicle	18	1	4
natural textures	144	2	2
wood	768	1	4
Oleg's faces	6,300	1	5
Olga's faces	6,300	1	2

3.2.3 Generalization to new data

The conventional LLE operates in a batch mode, that is, it obtains a low-dimensional representation for the fixed amount of high-dimensional data points to which the algorithm is applied. In the case of new data points arriving, one needs to completely rerun the original LLE for the previously seen data set, augmented by the new data points. In other words, the original LLE lacks generalization[3] to new data. This makes the algorithm less attractive, especially for large data sets of high dimensionality in a dynamic environment, where a complete rerun of LLE becomes prohibitively expensive.

As described in Bengio *et al.* (2004), it is much more challenging to make LLE incremental than the other manifold learning algorithms. Moreover, LLE searches for the bottom eigenvectors and eigenvalues, which are frequently ill-conditioned. Ill-conditioning means that eigenvalues or/and eigenvectors are susceptible to small changes of a matrix for which they are computed (for details, see Appendix 2). As a result, the problems one faces when making LLE incremental are more formidable than those for other manifold

[2]Let us consider the following situation: one is given D-dimensional data and by using any feature extraction algorithm we can obtain its low-dimensional representations of dimensionality d_1 and d_2. Then, two possibilities exist: 1) when $d_1 < d_2$ then the representation of dimensionality d_1 is less informative than that of dimensionality d_2 because the latter representation includes information contained in the former one; and 2) when $d_1 \ll d_2$ then we have an opposite case - underestimation is safer. Here we consider a situation when $d_1 \equiv d_2$, i.e., the first case.

[3]Generally speaking, two different types of generalization are possible: interpolation and extrapolation. Interpolation, in most cases, can be done reliably, while extrapolation is mostly unreliable. Therefore, extrapolation is beyond the interest of the thesis. Further, the term generalization assumes interpolation.

learning algorithms.

In Kouropteva (2001), an attempt was made to adapt LLE to the situation where the data comes incrementally point-by-point. Two simple techniques were proposed, where the adaptation to a new point can be done by updating either the weight matrix $\mathbf{W}$ or the cost matrix $\mathbf{M}$. In both 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 an incremental version of LLE that follows the latter approach (Kouropteva *et al.* 2005a,b).

Suppose we are given already projected data $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, corresponding projected points $\mathbf{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$, and a new point $\boldsymbol{x}_{N+1} \in \mathbb{R}^D$, which is sampled from the same data manifold as $\mathbf{X}$. We are asked to find a new embedding point $\boldsymbol{y}_{N+1} \in \mathbb{R}^d$ corresponding to $\boldsymbol{x}_{N+1}$.

3.2.3.1 Linear generalization

In order to obtain the new embedded coordinates, 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 for linear generalization (Kouropteva *et al.* 2002a, Saul & Roweis 2003):

LG1: Let us put the K nearest neighbors of $\boldsymbol{x}_{N+1}$ and the corresponding embedded points into the matrices: $\mathbf{X}^{N+1} = (\boldsymbol{x}^1_{N+1}, \boldsymbol{x}^2_{N+1}, \ldots, \boldsymbol{x}^K_{N+1})$ and $\mathbf{Y}^{N+1} = (\boldsymbol{y}^1_{N+1}, \boldsymbol{y}^2_{N+1}, \ldots, \boldsymbol{y}^K_{N+1})$. 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 $\mathbf{Z} = \mathbf{Y}^{N+1}(\mathbf{X}^{N+1})^{-1}$, where $(\mathbf{X}^{N+1})^{-1}$ is pseudoinverse matrix of $\mathbf{X}^{N+1}$. Because $\mathbf{X}^{N+1}$ is the neighborhood of $\boldsymbol{x}_{N+1}$ 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 $\boldsymbol{y}_{N+1} = \mathbf{Z}\boldsymbol{x}_{N+1}$. 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 (Kouropteva *et al.* 2002a).

LG2: To find $\boldsymbol{y}_{N+1}$, first, the K nearest neighbors of $\boldsymbol{x}_{N+1}$ are detected among points in the high-dimensional space: $\boldsymbol{x}_i \in \mathbf{X}, i = 1, \ldots, N$. Then, the linear weights, $\boldsymbol{w}_{N+1}$, that best reconstruct $\boldsymbol{x}_{N+1}$ from its neighbors, are computed using Eq. 3 with the sum-to-one constraint: $\sum_{j=1} w_{N+1\,j} = 1$. Finally, the new output $\boldsymbol{y}_{N+1}$ is found: $\boldsymbol{y}_{N+1} = \sum_{j=1} w_{N+1\,j}\boldsymbol{y}_j$, where the sum is over the $\boldsymbol{y}_i$'s corresponding to the K nearest neighbors of $\boldsymbol{x}_{N+1}$ (Saul & Roweis 2003).

3.2.3.2 The incremental locally linear embedding algorithm

In order to construct embedding, LLE searches for the bottom eigenvectors of an $N \times N$ Hermitian matrix. Hence, in the case of LLE, one has to deal with the ill-conditioned eigenproblem. 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 λ_i by at most ε, i.e., computing $\lambda_i = 10^{-5}$ to within $\pm\varepsilon = 10^{-4}$ means that no leading digits of the computed λ_i may be correct (Bai *et al.* 2000, Appendix 2).

When a new data point arrives, the incremental LLE (ILLE) computes 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. 3 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 $\mathbf{M}\tilde{\mathbf{y}}_i^T = \lambda_i \tilde{\mathbf{y}}_i^T$ or in matrix form $\mathbf{M}\mathbf{Y}^T = \mathbf{Y}^T diag\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$, where $\tilde{\mathbf{y}}_i$ is the $(i+1)^{th}$ bottom eigenvector of the cost matrix $\mathbf{M}$ of size $1 \times N$: $\mathbf{Y} = \begin{pmatrix} \tilde{\mathbf{y}}_1 \\ \tilde{\mathbf{y}}_2 \\ \vdots \\ \tilde{\mathbf{y}}_d \end{pmatrix}$. Since typical eigenvectors are orthogonal, we can rewrite the eigenproblem: $\mathbf{Y}\mathbf{M}\mathbf{Y}^T = \mathbf{Y}\mathbf{Y}^T diag\{\lambda_1, \lambda_2, \ldots, \lambda_d\} = diag\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$. Further, we assume that the eigenvalues of the new cost matrix, $\mathbf{M}_{new}$, are the same as those for the cost matrix computed for N points, and $\mathbf{Y}_{new} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N, \boldsymbol{y}_{N+1})$. This can be done since the eigenvalues we are dealing with are very close to zero, usually they are of the order 10^{-p}, where p is large enough (practically about 10). Therefore we can write $\mathbf{Y}_{new}\mathbf{M}_{new}\mathbf{Y}_{new}^T = diag\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$, where $\{\lambda_i\}, i = 1, \ldots, d$ are the smallest eigenvalues of the cost matrix, starting from the second one, computed for N points. The new coordinates are obtained by solving the $d \times d$ minimization problem:

$$\min_{\mathbf{Y}_{new}} (\mathbf{Y}_{new}\mathbf{M}_{new}\mathbf{Y}_{new}^T - diag\{\lambda_1, \lambda_2, ..., \lambda_d\}). \tag{16}$$

The LLE constraints imposed on the embedding coordinates should be respected, too, leading to slight displacement of all emmbedded coordinates. Thus, the $N \times N$ problem of the third LLE step is 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. As a result, embedded coordinates of the new data point are obtained.

As might be noted, ILLE is actually a good generalization of new data, which solves the incremental eigenproblem for the smallest eigenvectors of the cost matrix $\mathbf{M}$. Usually, in the incremental eigenproblem (even for eigenvectors corresponding to the largest eigenvalues), the accuracy is lost compared to batch methods. The inaccuracy is small when only a few incremental updates are made, and it is acceptable for the great majority of applications (Hall *et al.* 2000). In order to obtain a more exact solution, it is necessary to update the embedding by recalculating the eigenvalues and eigenvectors of the cost matrix $\mathbf{M}$ in a number of iterations of ILLE.

In the experiments, we applied the LG1, LG2 and ILLE to the following data sets: s-curve, wine, Frey faces, MNIST digits (3 & 8), Coil-20, Oleg's faces, Olga's faces (for the data sets description, see Appendix 1). All data sets were divided into training (70%) and test (30%) sets. The training sets were projected by the original LLE to two-dimensional spaces, and the test sets were mapped to the corresponding spaces by the LG1, LG2, and ILLE algorithms.

The first experiment is done with the swiss-roll data set (Figure 10 (a)). In the beginning, we project the initial data containing 1,400 points using the conventional LLE algorithm ($K = 15$), and then we add the test data points in a random order. In Figures 10 (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.

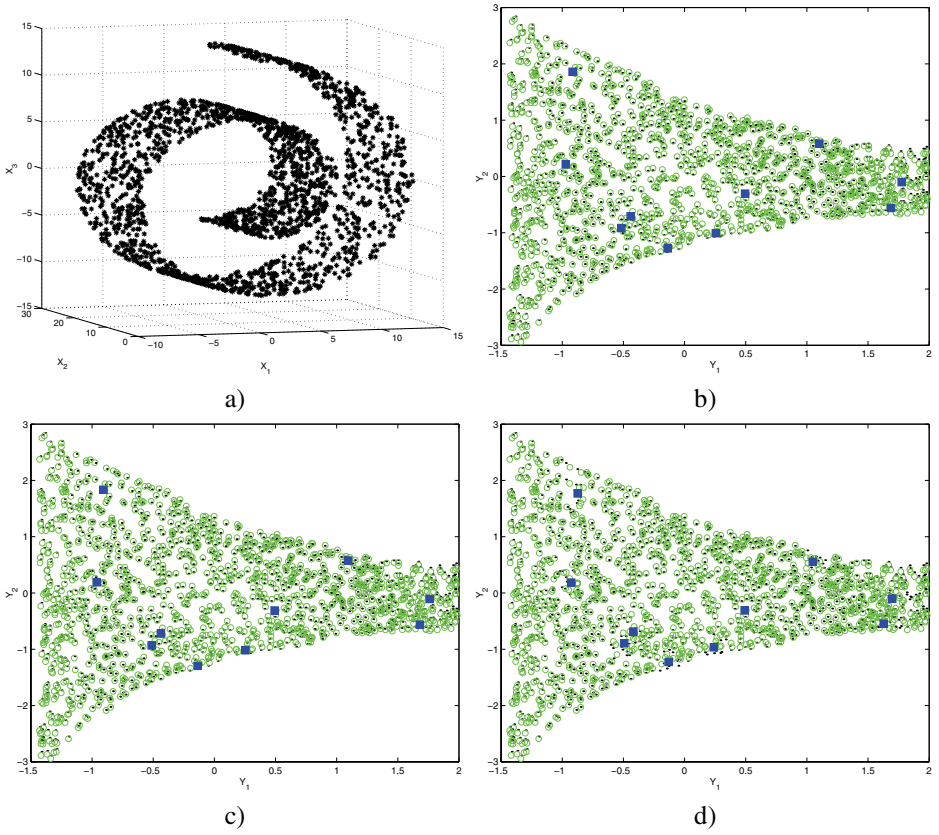


Fig. 10. LLE generalization on the swiss-roll data set. (a) The original 3D data points sampled from the corresponding data manifold. The dots (·) depict the target coordinates, i.e. those which are obtained by applying the conventional LLE to the pooled data set, including both old and new data points. The circles (○) 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 by the incremental methods.

In order to quantitatively compare the generalization methods, we compute two characteristics, namely, Spearman's rho (Siegel & Castellan 1988) and the procrustes measure (Seber 1984) (for a description of the criteria, see Section 4.3). 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 the procrustes measure, the better fitting is obtained. Both Spearman's rho and the procrustes measure are commonly used for estimating topology preservation when performing dimensionality reduction.

We estimate Spearman's rho and the procrustes measure after each iteration for all projected data points. Consider the experiment with the wine data set: 119 training points are projected by the original LLE ($K = 15$) and the other 51 test points are added during 17 iterations ($n = 3$ points at a time) by LG1, LG2, and ILLE. In Figure 11, plots show Spearman's rho and the procrustes measure estimated after each iteration. One can see that ILLE performs better than LG1 and LG2 as the number of new samples increases.

Table 6. The number of iterations for which a particular method (LG1, LG2, ILLE) outperforms others in terms of Spearman's rho (ρ_{Sp}) and the procrustes measure (Procr). The largest resulting values are underlined.

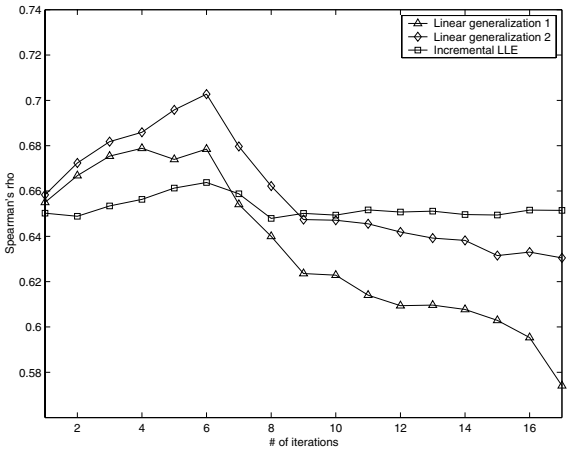
Data	# of iterations	# of points per iteration		LG1	LG2	ILLE
S-curve	60	10	ρ_{Sp}	<u>47</u>	12	1
			Procr	<u>43</u>	16	1
Wine	17	3	ρ_{Sp}	0	8	<u>9</u>
			Procr	0	<u>10</u>	7
Frey faces	28	21	ρ_{Sp}	1	8	<u>19</u>
			Procr	0	<u>14</u>	<u>14</u>
MNIST digits	22	27	ρ_{Sp}	0	0	<u>22</u>
			Procr	0	<u>22</u>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

The same test is done for all data sets. We count the number of iterations for which a particular method outperforms others in terms of Spearman's rho and the procrustes measure and place the results into Table 6. This table also lists the number of iterations and the number of points per iteration for all data sets. The largest resulting values are underlined. By looking at Table 6, a number of interesting observations can be made:

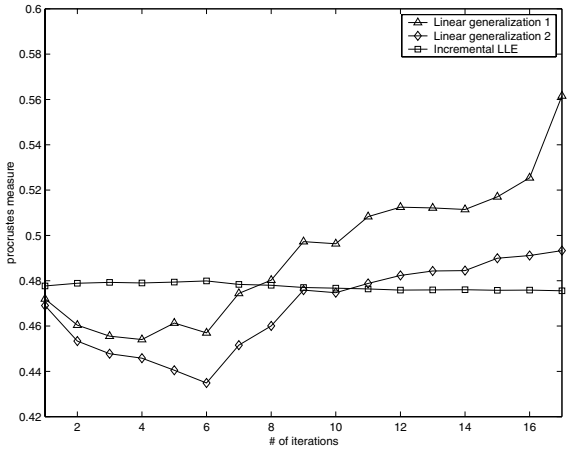
– When the manifold is well and evenly sampled, and the relationships between the

original and embedded data sets are close to being locally linear, as in the case of the S-curve and Oleg's faces, LG1 and LG2 are sufficient for successful generalization of test points, and this fact is confirmed by both Spearman's rho and the procrustes measure.

– However, if the data manifold is non-uniformly sampled, and the relationships are locally nonlinear as in the case of Olga's and Frey faces, ILLE emerges as a clear winner, since it does not only rely on linear relationships. This is again supported by both Spearman's rho and the procrustes measure.



a)



b)

Fig. 11. **Spearman's rho (a) and the procrustes measure (b) for the wine data set.**

3.3 Summary

LLE belongs to the group of manifold learning algorithms. Its ability to deal with large sizes of high-dimensional data and its non-iterative way of finding the embeddings make it more attractive to researchers. A disadvantage of LLE is that it is suitable only for well-sampled manifolds. In this case, the assumption that the manifold can be approximated by a set of linear hyperplanes is valid. However, not all manifolds in practice meet such a condition. Examples are spheres, torus and cylinders, which are impossible to unfold using LLE.

An advantage of LLE comes from few parameters to be set: the number of nearest neighbors, K, and the dimensionality of an embedded space, d. In Section 3.2.1, we proposed a hierarchical method for automatic selection of the K_{opt} and applied it to the visualization of high-dimensional data in a two-dimensional embedded space. The results obtained indicate that the hierarchical method is feasible and can be used in combination with a suitable cluster detection technique at the next level of analysis. Compared to the straightforward method, the hierarchical one is generally less time-consuming, while yielding very similar results.

In order to find an embedding, LLE needs a specified number of dimensions that will be used. This parameter might be chosen by a user or found a priori by any intrinsic dimensionality estimation algorithm. LLE based intrinsic dimensionality estimation of data (Polito & Perona 2002) was discussed in Section 3.2.2. According to the results obtained, this method is comparable with the classical one and, therefore, can be used in practice.

An important drawback of LLE is a lack of generalization when new points are to be added, i.e., in this case, we have to rerun LLE on pooled (both old and new) data in order to obtain the embeddings. In Section 3.2.3, we solved this formidable task and proposed an incremental version of LLE. Moreover, it was compared to two linear generalization methods. The results demonstrate that ILLE is a powerful method for data whose distribution is not uniform and the local linearity constraints do not hold. In contrast, the linear generalization methods deal well with evenly-sampled manifolds of artificially generated data, but may have problems with real-world data sets.

4 The locally linear embedding algorithm in unsupervised learning: a comparative study

Learning is the process of estimating an unknown input-output dependency or structure of a system using a limited number of observations. The general learning scenario involves three components: a generator of random input vectors, a system which returns an output for a given input vector, and the learning machine which estimates an unknown input-output mapping of the system from the observed samples. This formulation describes many practical learning problems such as interpolation, regression, classification, clustering, and density estimation.

Supervised and unsupervised learning are two subclasses of learning algorithms. For supervised learning, both input and target output data are known. Learning is done in accordance with the direct comparison of the actual output with known correct answers. However, in many real world tasks the outputs are not well defined, since it might be time-consuming to label thousands of samples, or experts could assign different labels to the same item. Thus, in such circumstances unsupervised learning is more suitable. The characteristic feature of unsupervised learning is that the training set only contains input samples. An unsupervised learning machine is expected to create categories of similar input patterns from their correlations to produce outputs corresponding to the input categories.

Many scientists believe that the human brain operates more like an unsupervised machine. For instance, there are around 10^6 photoreceptors in each eye and their activities are constantly changing with the visual world thus providing a human being with information about the visual world: what the objects are around, of which color and size, how they are placed in the environment, what the lighting conditions are etc. However, essentially no information about the scene contents is available during learning. Therefore, in order to attain learning capabilities close to those of the brain, unsupervised learning should be used.

In this chapter, unsupervised learning and its applications are discussed. The practical applications of LLE to real world data sets with different features are described. These results are compared with those obtained by applying the Isomap, SOM and S-LogMap algorithms to the same data sets.

4.1 Tasks involving unsupervised learning

A data mining process serves for discovering and visualizing the regularities, structures and rules that are hidden in high-dimensional data. Usually, the mining process is made on data without a teacher for supervision, i.e., in an unsupervised way. Hence, data mining is the primary goal of unsupervised learning, and it is used for: dimensionality reduction, data compression, finding clustering structure in the data, modeling the data density, detecting outliers in the data, finding features that are useful for the data categorization etc. (Xu 1999).

Clustering

The objective of clustering is to construct decision boundaries of an unlabeled data set. In other words, clustering attempts to automatically generate data labels by searching for natural groupings, or clusters, in high-dimensional data based on estimated similarities among the data samples. This is a challenging task, since data can reveal clusters with different shape, size and structure. Moreover, the number of clusters in the data depends on the resolution with which the data is considered, features used for data analysis (pixel values, histograms or special kind of features) etc.

Formally, the problem of clustering can be stated as follows: given N data samples in a D-dimensional space, determine a partition of the data into k clusters in such a way that each cluster contains samples which are more similar to each other than to samples belonging to other clusters (Jain & Dubes 1988). A large number of clustering algorithms is described in the literature (Hartigan 1975, Jain & Dubes 1988, Duda *et al.* 2001). Most of them are based on hierarchical and partitional approaches. Techniques belonging to the former case organize data in a nested series of groups (which can be displayed in the form of a tree), while the latter methods produce only one.

Dimensionality reduction

The dimensionality reduction operation obtains a compact representation of high-dimensional data and highlights meaningful relationships while trying to eliminate noisy factors affecting the results of higher order data analysis, e.g., visualization, clustering etc. In practice, unsupervised learning algorithms are often used for dimensionality reduction, since they do not need to have label information to project the high-dimensional data into low-dimensional space. The absence of labels is a big advantage of unsupervised learning due to the fact that labeling real-world data is a difficult and time-consuming task.

There are two main reasons for keeping data dimensionality (i.e., the number of features) as small as possible: measurement cost and classification accuracy. A limited, yet salient feature set simplifies both the pattern representation and the classifiers that are built on the selected representation. Consequently, the resulting classifier will be faster and will use less memory. Moreover, as stated earlier in Chapter 2, a small number of features can alleviate the curse of dimensionality when the number of training samples is limited. On the other hand, reduction in the number of features may lead to a loss in the discrimination power and thereby lower the accuracy of the resulting recognition system (Jain *et al.* 2000).

Finding regularities in data

Finding and capturing regularities hidden in high-dimensional data plays an important role for successful data interpretation. First of all, by exploiting regularities in the data, it is possible to reduce the total amount of information necessary to represent it. Secondly, discovering regularities in the data helps make a prediction about future inputs. Moreover, finding regularities facilitates further data analysis such as extracting the most representative patterns, clustering, noise and outliers detection etc.

The simplest kind of regularity is the equivalence of objects. That is, one object is regarded in a particular sense and context to be the same as another object (Aldridge 1998). For example, a chair and a stool serve the same purpose of seating, but they are not identical. Therefore, equivalence relation is not the same as identity relation, i.e., objects are identical if and only if they are equivalent in all respects, which is impossible unless the domain of possible equivalences is sufficiently constrained.

Indexing and retrieving

Indexing and retrieving play an important role in data usability. Indexing is done by assigning a particular index to each data pattern. These indices are used to provide access to information contained in the data, e.g., to perform information retrieval. Since the indices are used for retrieving data information, the time spent on retrieving is low. Moreover, a well-designed index is an exceptional navigation aid for data analysis and processing.

Nowadays, indexing and retrieving are widely used in practice: i) document and text retrieval, where a user can search and retrieve an archived document from a database by entering a key word; ii) image retrieval, where a desired image can be found by using RGB values or other features; iii) sound and audio retrieval, where a desired sound or music is obtained by describing the sound situation, the sound itself or the sound name etc.

Modeling density distribution

Density estimation refers to the problem of estimating density (or probability) that a member of certain category will be found to have particular features (Duda *et al.* 2001). A major class of unsupervised learning methods consists of maximum likelihood density estimation methods, which cover a wide spectrum of the principles that have been suggested for unsupervised learning. This includes versions of the notion that the outputs should convey most of the information in the input; that they should be able to reconstruct the inputs well, perhaps subject to constraints such as being independent or sparse; and that they should report on the underlying causes of the input. Many different mechanisms apart from clustering have been suggested for each of these, including forms of Hebbian learning, the Boltzmann and Helmholtz machines, sparse-coding, various other mixture models, and independent components analysis (Dayan 1999).

Outlier detection

In the literature, outliers are defined as observations in a data set which appear to be inconsistent with the remainder of that data set, or which deviate strongly from other observations so as to arouse suspicions that they were generated by a different mechanism (Barnett & Lewis 1994). The identification of outliers can lead to the discovery of unexpected knowledge hidden in data sets, facilitate further data anal-

ysis and has a number of practical applications such as credit card frauds, medical analysis, weather prediction etc.

In most cases, outliers are considered as noise, which has to be eliminated in order to improve the predictive accuracy of the learning algorithms. However, outliers themselves might be of great interest, since "one person's noise could be another person's signal" (Han & Kamber 2001). Unsupervised learning based methods for outliers detection can be categorized into several groups: i) statistical-based methods, which assume that a distribution model of data is known; ii) deviation-based methods that inspect characteristics of objects; iii) density-based techniques, where outliers are detected based on the density of the local neighborhood; iv) distance-based methods are based on ordinary distances.

Visualization

Humans are not able to perceive the intrinsic structure of a high-dimensional data set without the use of any visualization technique that serves to produce a one, two or three-dimensional plot of the data. Visualization is a useful tool providing humans with an ability to understand, analyze and explore the specificity, structure and clustering tendency of the data. Over the last decades, many visualization techniques have been proposed and developed. They can be classified into five general groups: scatterplots or thematic maps, glyphs, parallel coordinates, hierarchical techniques and feature extraction methods. Data visualization is considered in Chapter 5.

4.2 Unsupervised learning algorithms

In this section, three unsupervised learning methods used for manifold learning are described in detail. The first one is isometric feature mapping (Isomap) (Tenenbaum *et al.* 2000). It is closely related to LLE, since both of these algorithms attempt to preserve the local properties of high-dimensional data while obtaining a nonlinear projection in a low-dimensional space; therefore, the comparison between them is of prime interest. The self-organizing map (SOM) is the second method presented here (Kohonen 1997). It projects high-dimensional data into a two or three-dimensional grid. SOM is widely used in practice (Kaski *et al.* 1998, Oja *et al.* 2003); hence, it is interesting to compare the performance of LLE and SOM. The third method described in this section is called S-LogMap (Brun *et al.* 2005). This is a novel and promising technique for manifold learning that is based on Riemannian normal coordinates. The performance of this method has been explored only on smooth manifolds with low intrinsic dimensionality possessing uniform distribution, i.e., artificially generated manifolds. Thus, it is worth comparing the performance of LLE, Isomap, SOM, and S-LogMap.

4.2.1 Isometric feature mapping

The purpose of Isomap is to discover an intrinsic structure of data sampled from a nonlinear manifold embedded into high-dimensional space while preserving the topology of the

data. This non-iterative algorithm finds a local minima-free solution, which is not always the case for many iterative feature extraction techniques. The main distinguishing feature of Isomap from other algorithms is that it obtains a low-dimensional representation of data by preserving geodesic distances calculated over high-dimensional space.

As input, Isomap requires N D-dimensional data points (one point per pattern) assembled in a matrix $\mathbf{X}$: $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N), \boldsymbol{x}_i \in \mathbb{R}^D, i = 1, \ldots, N$. As output, it produces N d-dimensional points ($d << D$) assembled in a matrix $\mathbf{Y}$: $\mathbf{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N), \boldsymbol{y}_i \in \mathbb{R}^d, i = 1, \ldots, N$. The i^{th} column of $\mathbf{X}$ corresponds to the i^{th} column of $\mathbf{Y}$.

The following three steps represent the Isomap algorithm (Tenenbaum *et al.* 2000):

1. For each $\boldsymbol{x}_i \in \mathbb{R}^D, i = 1, \ldots, N$ find its K nearest neghbors: $\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^K$ by using the Euclidean distance as a similarity measure. Then the graph $\mathscr{G}$ is constructed over all data points by connecting points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if $\boldsymbol{x}_j \in \{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^K\}$ and these points organize in $\mathscr{G}$ the nodes i and j, correspondingly. The edge lengths are equal to the Euclidean distances between points $d_x(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
2. Estimate the geodesic distances between all pairs of points on the data manifold by computing their shortest path distances in the graph $\mathscr{G}$: $d_{\mathscr{G}}(i, j)$ using Floyd's algorithm (Foster 1995). Initialize $d_{\mathscr{G}}(i, j) = d_x(\boldsymbol{x}_i, \boldsymbol{x}_j)$ if the nodes i and j are connected and ∞ otherwise. Then for each node k, set each $d_{\mathscr{G}}(i, j) = \min\{d_{\mathscr{G}}(i, j), d_{\mathscr{G}}(i, k) + d_{\mathscr{G}}(k, j)\}$. The calculated geodesic distances form a symmetric matrix $\mathbf{D}_{\mathscr{G}}$.
3. Apply the classical multidimensional scaling (MDS) algorithm to the geodesic distance matrix $\mathbf{D}_{\mathscr{G}}$. Assume that the outputs are translation-invariant, i.e., $\sum_{i=1}^{N} \boldsymbol{y}_i = 0$. The target inner product matrix $\mathbf{B}$ can be found by $\mathbf{B} = \mathbf{H}\mathbf{D}_{\mathscr{G}}\mathbf{H}$, where $\mathbf{H} = \{h_{ij}\}, i, j = 1, \ldots, N, h_{ij} = \delta_{ij} - 1/N$, and the delta function $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Coordinates of the low dimensional space are found as $\mathbf{Y} = (\sqrt{\lambda_1}\boldsymbol{v}_1, \sqrt{\lambda_2}\boldsymbol{v}_2, \ldots, \sqrt{\lambda_d}\boldsymbol{v}_d)^T$, where $\{\lambda_i\}, i = 1, \ldots, d$ are the d largest eigenvalues of $\mathbf{B}$ with corresponding eigenvectors $\{\boldsymbol{v}_i\}, i = 1, \ldots, d$.

The low-dimensional embedding found preserves the geodesic distances of the data as closely as possible. Hence, Isomap preserves the global geometric properties of manifold as well as the local ones, all of which are characterized by the geodesic distances (see Figure 12).

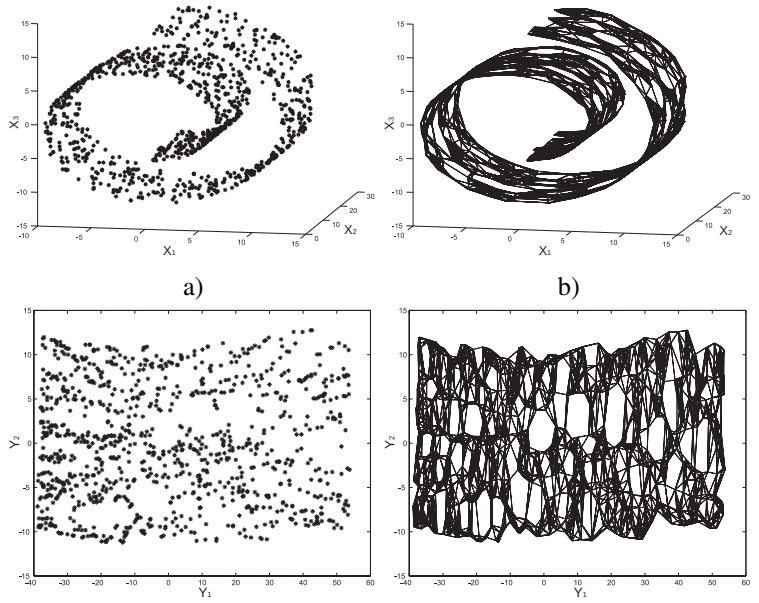


Fig. 12. a) 2D swiss-roll data set embedded into 3D space; b) graph $\mathscr{G}$ constructed over the swiss-roll data points; c) 2D Isomap resulting projection of the swiss-roll data; d) corresponding geodesic distances preserved in the 2D Isomap projection.

4.2.2 The self-organizing map

SOM preserves the topology of high-dimensional data by using a completely different approach, which is based on training a neural network (Kohonen 1997). It builds a two (one, three, or multi) dimensional regular lattice of nodes, the size of which is defined in advance (let M be the number of nodes). These nodes represent the high-dimensional data in a vector quantization way, i.e., each node $i = 1,\ldots,M$ is associated with a codeword $\boldsymbol{m}_i \in \mathbb{R}^D$ corresponding to and representing a part of the input space. SOM uses these codewords as weights vectors of the neural network. Figure 13 schematically represents the structure of a SOM lattice, where each neuron is fully connected to all elements of the input vector through the corresponding weight vector.

In the simplest case, an input vector $\boldsymbol{x}$ is connected to all neurons in parallel via variable weights $\boldsymbol{m}_i, i = 1,\ldots,M$, which are in general different for different neurons. In an abstract scheme, it may be imagined that the input $\boldsymbol{x}$, by means of some parallel computing mechanisms, is compared with all the $\boldsymbol{m}_i$, and the location of the best match is defined as the location of the "response". In many practical applications, the best matching node or so called best matching unit (BMU) is defined based on the smallest Euclidean distances between $\boldsymbol{x}$ and $\boldsymbol{m}_i, i = 1,\ldots,M$. Then the weight vectors of this node and its nearest neighbors are updated. Thus, the nonlinear projection of the high-dimensional input vector $\boldsymbol{x}$ onto a two-dimensional space is realised.

The sequential SOM algorithm consists of the following steps:

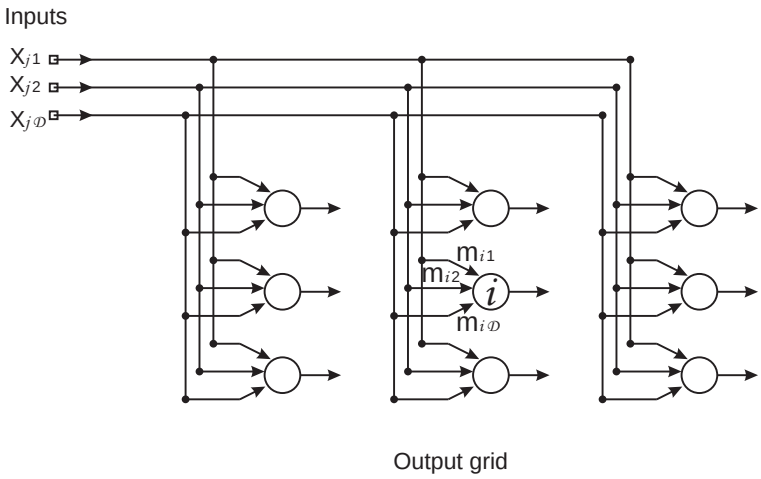


Fig. 13. **A 2D grid of neurons schematically representing SOM structure. In practice, the lattice type of the node array can be rectangular, hexagonal, or even irregular.**

1. Randomly choose initial values for the weight vector $\boldsymbol{m}_i$ of the node $i, i = 1, \ldots, M$: $\boldsymbol{m}_i(0)$. The 2D initial grid for s-curve data (Figure 14 (a)) is shown in Figure 14 (c).
2. Take $\boldsymbol{x}$ from a set of input samples and find the best-matching unit b. The choice of this node b is based on the nearest neighbor principle:

$$b = \arg\min_i \{\| \boldsymbol{x} - \boldsymbol{m}_i \|\}, \tag{17}$$

   where $\| \cdot \|$ is a distance measure, typically Euclidean.
3. Adjust the weight vectors of the neighboring nodes of the best matching node b. This is achieved with the following rule:

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + \alpha(t) h_{bi}(t) [\boldsymbol{x}(t) - \boldsymbol{m}_i(t)], \tag{18}$$

   where $t = 0, 1, 2, \ldots$ is an iteration time step, $\alpha(t)$ is the learning rate and $h_{bi}(t)$ is a neighborhood function around the winner unit b, at time t. For convergence, it is necessary that $h_{bi}(t) \to 0$ when $t \to \infty$. The average width and form of h_{bi} define the stiffness of the elastic surface to be fitted to the data points.
4. Repeat Steps 2, 3 until the algorithm converges. In the illustrative example, the final 2D grid is depicted in Figure 14 (d).

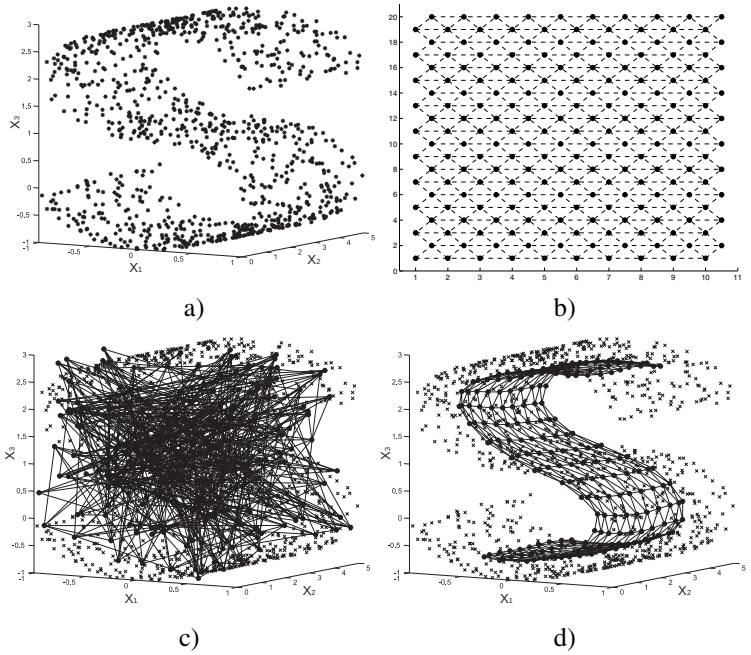


Fig. 14. a) 2D s-curve data set embedded into 3D space; b) 2D grid in visualization space; c) 2D grid at the initialization stage in the 3D feature space; d) 2D grid of the trained SOM in the 3D feature space. (The pictures of the SOM grid are obtained by SOM Toolbox for MatLab: http://www.cis.hut.fi/projects/somtoolbox/).

4.2.3 The sample LogMap

The main idea behind the S-LogMap algorithm is derived from the concept of Riemannian normal coordinates, which contain information about the direction and distance from a particular point to other neighboring points. Such information is useful for navigating on the data manifold and deriving its correct structure. Moreover, the Riemannian normal coordinates are closely related to the geodesics and the exponential and logarithmic maps of Lie-groups, which are used for signal processing (Fletcher *et al.* 2003).

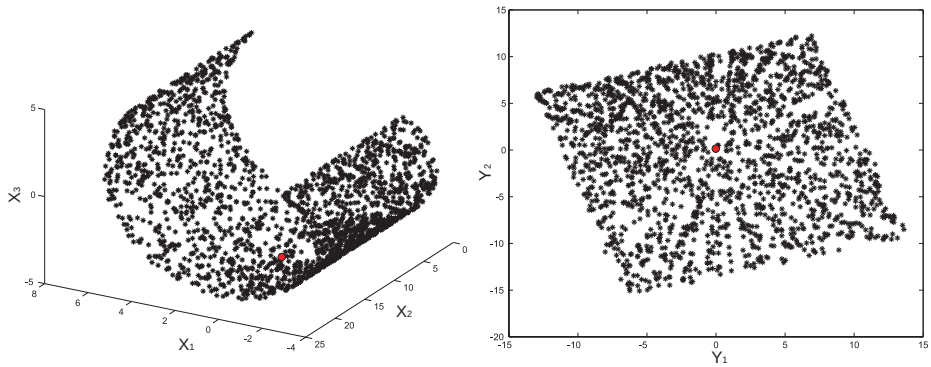
Let the input data be represented by a matrix $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, where each $\boldsymbol{x}_i$ is sampled from a manifold $\mathscr{M}$ embedded into D-dimensional space. Suppose we are given a basis point $\boldsymbol{x}_p \in \mathbf{X}$ and an orthonormal basis of the tangent space at $\boldsymbol{x}_p$ to the manifold $\mathscr{M}$, $T_p\mathscr{M}$. The goal of S-LogMap is to express all data points $\boldsymbol{x}_i, i = 1, \ldots, N$ via $T_p\mathscr{M}$ using Riemannian normal coordinates, which is equivalent to measuring the distance and direction from $\boldsymbol{x}_p$ to every other point in the data set. This is done by using a definition of the logarithmic function on the manifold (Brun 2006): the logarithmic function maps the manifold's points to the corresponding tangent space, while the exponential function (inverse of the logarithmic function) maps points from the tangent space to points on the manifold.

The S-LogMap algorithm consists of the following five steps (Brun *et al.* 2005):

1. Choose a base point $\boldsymbol{x}_p$ from the data set $\mathbf{X}$.

2. Select a ball $B(\boldsymbol{x}_p)$ of the K closest points around $\boldsymbol{x}_p$. Then by applying PCA in the ambient space to $B(\boldsymbol{x}_p)$ a tangent space $T_p\mathscr{M}$ is obtained. In this space, an orthonormal basis $\{\hat{\boldsymbol{e}}_i\}, i = 1,\ldots,N$ is chosen and used to map all points $\hat{\boldsymbol{x}}$ belonging to $B(\boldsymbol{x}_p)$ to $T_p\mathscr{M}$ by projection on $\{\hat{\boldsymbol{e}}_i\}$ in the ambient space. This mapping is called ψ-mapping.
3. Build graph $\mathscr{G}_L$ on the manifold $\mathscr{M}$ in the same manner as in the Isomap using L neighboring points. Then approximate geodesic distances on $\mathscr{M}$ by applying Dijkstra's algorithm for finding the shortest paths in the graph. This gives estimates of pairwise distances $d_{\mathscr{G}_L}(\boldsymbol{x},\hat{\boldsymbol{x}})$ for $(\boldsymbol{x},\hat{\boldsymbol{x}}) \in \mathbf{X} \times B(\boldsymbol{x}_p)$.
4. Estimate $\breve{\boldsymbol{g}} = \sum g^i \hat{\boldsymbol{e}}_i = \bigtriangledown_{\hat{\boldsymbol{x}}} d^2_{\mathscr{G}_L}(\boldsymbol{x},\hat{\boldsymbol{x}}) \mid_{\hat{\boldsymbol{x}}=\boldsymbol{x}_p}$ numerically using values obtained in the previous step. This is done in order to calculate the direction from $\boldsymbol{x}_p$ to every point $\boldsymbol{x} \in \mathbf{X}$ (see Appendix 4).
5. Estimate Riemannian normal coordinates for all points $\boldsymbol{x}$ using equation $\boldsymbol{x}^i = d_{\mathscr{G}_L}(\boldsymbol{x},\hat{\boldsymbol{x}}) \frac{g^i}{|\breve{\boldsymbol{g}}|}$.

S-LogMap can obtain a discontinuity in the mapping, which might be considered as a problem in some applications, but it can also be seen as a useful feature, e.g., for unfolding such manifolds as a torus or even a Klein bottle where LLE and Isomap fail to find a continuous embedding (Brun *et al.* 2005). Figure 15 shows the projection obtained with S-LogMap applied to the simple flat C-curve manifold.



a) b)

Fig. 15. a) 2D c-curve data set embedded into 3D space; filled red circle corresponds to the $\hat{x_p}$; b) 2D S-LogMap resulting projection of the c-curve data; filled red circle corresponds to the $\psi(\hat{x_p})$.

4.3 Evaluation criteria

In this section, the evaluation criteria for estimating the performance of the mapping methods are described. These characteristics are Spearman's rho (Siegel & Castellan 1988), the procrustes measure (Seber 1984), classification rate reduction and clustering performance (Niskanen & Silven 2003). The former two estimations demonstrate how well

the data topology is preserved in the projected space, while the latter two estimate the classification capability of the data projection obtained.

Spearman's rho is a product-moment correlation coefficient between two sets of variables, which is calculated from the relative rank of each value rather than the value itself. Spearman's rho is computed by using the following equation:

$$\rho_{Sp} = 1 - \frac{6\left(\sum_{i=1}^{N} \check{d}_i^2\right)}{N(N^2-1)}, \tag{19}$$

where $\check{d}_i$ is the difference in rank of corresponding variables[1], and N is the number of elements to be compared. The interval of ρ_{Sp} is $[-1,1]$. If $\rho_{Sp} = 1$, then there is a perfect positive correlation between the two sets of variables. If $\rho_{Sp} = -1$, then there is a perfect negative correlation between the two sets of variables. If $\rho_{Sp} = 0$, then there is generally no correlation between these sets. In other words, the closer ρ_{Sp} to 1 is, the better the data topology is preserved in the projected space.

The procrustes measure reflects the matching of two sets of variables in terms of distances. It 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 measure of goodness of fit of the original data points $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ to the projected points $\mathbf{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)$ is

$$\begin{aligned} Procr(\mathbf{X},\mathbf{Y}) &= \sum_{i=1}^{N} \| \boldsymbol{x}_i - (\mathbf{Q}\boldsymbol{y}_i + \boldsymbol{c}_i) \|^2 = \\ &= Tr[(\mathbf{X} - (\mathbf{QY} + \mathbf{C}))(\mathbf{X} - (\mathbf{QY} + \mathbf{C}))^T], \end{aligned} \tag{20}$$

where $\mathbf{Q}$ is a matrix of size $D \times d$ containing scale, orthogonal rotation and reflection components; $\mathbf{C}$ is a matrix of size $D \times N$ containing a translation component; $\boldsymbol{c}_i$ is the i^{th} column of the matrix $\mathbf{C}$; Tr denotes the trace of the matrix given in squared brackets. Since the dimensionality of the original data space D is larger than the dimensionality of the projected data space d, a column of $(D-d)$ zeroes is added to each $\boldsymbol{y}_i$ in order to obtain matrices of the same size $(D \times N)$. The smaller the value of the procrustes measure, the better the fitting obtained.

Classification rate reduction is used to estimate the classification capability of the projected data, i.e., answer the question: are the samples belonging to the same class mapped closely or not? In order to calculate the measurement, the following formula is used:

$$R = \frac{N_x^{correct} - N_y^{correct}}{N_x^{correct}}, \tag{21}$$

where $N_x^{correct}$ and $N_y^{correct}$ define the number of correctly classified data samples in the original and projected spaces, correspondingly. To obtain these values, an ordinary K nearest neighbor (KNN) classifier is used with $K = 1,3,5$. The smaller the R, the better the classification capability of the projected data.

The clustering performance coefficient shows how the data clusters are separated from each other in a particular data space, i.e., it demonstrates how well the data clusters

[1] In the experiment, we compare the ranks of the squared pairwise distances calculated for the original and projected data points.

are organized. The within-cluster Eq.22 and between-cluster Eq.23 scatter matrices are used to evaluate Eq.24.

$$\mathbf{S}_W = \sum_{i=1}^{N_{classes}} \sum_{\mathbf{y} \in \omega_i} (\mathbf{y} - \mu_i)(\mathbf{y} - \mu_i)^T, \tag{22}$$

$$\mathbf{S}_B = \sum_{i=1}^{N_{classes}} N_i (\mu_i - \mu)(\mu_i - \mu)^T, \tag{23}$$

$$C = \frac{\mid \mathbf{S}_W \mid}{\mid \mathbf{S}_W \mid - \mid \mathbf{S}_B \mid}, \tag{24}$$

where $N_{classes}$ denotes the number of classes, μ_i refers to the mean of class ω_i composed of N_i samples, and μ is the mean of all the data. Here, we calculate the clustering performance for the original (C_x) and projected (C_y) data spaces in order to see the clustering tendency of a mapping algorithm. A good clustering tendency of the algorithm corresponds to the smallest C.

The criteria given in Eqs. 21 and 24 in many cases correspond to a human judgment of good clustering, where a between classes boundary can be drawn quite robustly (Niskanen & Silven 2003).

In the following experiments, these two characteristics are calculated only for data sets having several known classes, since label information is used to estimate R and C. Note that all data projections are made in an unsupervised way.

4.4 Experiments

In the experiments, LLE, Isomap, SOM and S-LogMap are applied to different data sets (for the data sets description, see Appendix 1). For convenience, Table 7 describes some properties of the data sets: the original dimensionality D[2], the number of samples, and number of classes in the case of labeled data. In order to have equivalent conditions, LLE, Isomap, SOM and S-LogMap project all data sets into two-dimensional spaces.
To compare the performance of the algorithms, Spearman's rho and the procrustes measure are calculated for all data sets, and the corresponding projections obtained with LLE, Isomap, SOM and S-LogMap. The results are represented in Tables 8 and 9. There are two variants of the topology preservation coefficients - local and global. In order to obtain the local estimations for a particular data set, these measurements are calculated for each point and its neighborhood, and then the average values are taken as the results. Global estimations mean that the measurements are calculated for all points in a data set. The best values in the resulting tables are shown in bold.

As can be seen from Tables 8 and 9, Isomap obtains the best topology preservation, while LLE and SOM yield slightly worse results. This can be explained by the design of the Isomap algorithm, since it preserves local topology by using geodesic distances and global topology by applying MDS to these distances. S-LogMap perfectly deals with

[2]Oleg's and Olga's faces data sets are reduced by factor 2. Hence, the difference between the dimensionalities of the original data spaces in comparison to those represented in Table 5.

Table 7. Short description of data sets used in the experiments.

Data	*D*	# of samples	# of classes
c-curve	3	2,000	–
s-curve	3	2,000	–
swiss-roll	3	2,000	–
iris	4	150	3
wine	13	178	3
natural textures	144	2,998	6
wood	768	1,947	8
MNIST digits 0&1	784	2,115	2
MNIST digits 3&8	784	1,984	2
MNIST digits 1&5&8	784	3,001	3
Oleg's faces	1,575	1,130	–
Olga's faces	1,575	1,200	–

artificial and simple data sets, but does not achieve good results with real-world ones, such as digits and faces. LLE, Isomap and S-LogMap are better in local topology preservation than in global. This is obvious, since these methods serve to preserve local topology by utilizing a particular approach (nearest neighbors, geodesic distances). In its turn, SOM behaves differently, since its goal is to achieve better fitting of a grid to a global structure of data manifold, as shown in Figure 14 (d). Hence, the resulting projections of SOM can possess a topological distortion in a local sense.

Table 8. Spearman's rho (ρ_{Sp}) estimated locally (L) and globally (G) for the different data sets. The local measure is obtained by taking an average over all measures computed for each point and its neighborhood. The best values are bold.

Data		LLE	Isomap	SOM	S-LogMap
c-curve	L	0.9807	**0.9921**	0.5287	0.9691
	G	0.9055	0.9085	0.8955	**0.9090**
s-curve	L	0.9061	**0.9883**	0.5201	0.9662
	G	0.9213	0.8383	**0.9336**	0.8402
swiss-roll	L	0.7840	0.9707	0.5463	**0.9734**
	G	0.2751	0.3601	**0.6801**	0.3593
iris	L	**0.7415**	0.7330	0.4671	0.6793
	G	0.7865	**0.9864**	0.8944	0.9810
wine	L	0.6553	**0.9552**	0.3861	0.2689
	G	0.8625	**0.9997**	0.8167	0.2666
natural textures	L	0.3262	0.3904	0.2932	**0.4172**
	G	0.5541	**0.7155**	0.5708	0.5536
wood	L	0.4135	**0.4506**	0.3296	0.1745
	G	0.8870	**0.9290**	0.6474	0.0194
MNIST digits 0&1	L	0.4181	**0.4978**	0.4233	0.1960
	G	0.4216	**0.8316**	0.6448	0.0618
MNIST digits 3&8	L	0.4148	**0.4760**	0.4109	0.1620
	G	0.5508	0.6466	**0.6474**	0.0256
MNIST digits 1&5&8	L	0.4082	**0.4387**	0.3915	0.1607
	G	0.0402	**0.6294**	0.4676	-0.0180
Oleg's faces	L	**0.9765**	0.8202	0.5165	0.5292
	G	0.8919	**0.9494**	0.7780	0.0902
Olga's faces	L	0.8728	**0.9097**	0.4900	0.5741
	G	0.5229	**0.9696**	0.8099	0.2815

Table 9. The procrustes measure (Procr) estimated locally (L) and globally (G) for the different data sets. The local measure is obtained by taking an average over all measures computed for each point and its neighborhood. The best values are bold.

Data		LLE	Isomap	SOM	S-LogMap
c-curve	L	0.0158	**0.0066**	0.6048	0.0208
	G	0.2061	0.2134	**0.2019**	0.2123
s-curve	L	0.0800	**0.0103**	0.6111	0.0250
	G	0.1565	0.2499	**0.1444**	0.2474
swiss-roll	L	0.2090	0.0206	0.5930	**0.0181**
	G	0.7584	0.8906	**0.4720**	0.8906
iris	L	0.3815	**0.3783**	0.6235	0.4389
	G	0.3832	**0.0577**	0.1582	0.0687
wine	L	0.3719	**0.0935**	0.7384	0.9592
	G	0.4851	**0.0032**	0.2699	0.8064
natural textures	L	0.7842	**0.7358**	0.7968	0.7717
	G	0.9032	**0.6654**	0.7434	0.9646
wood	L	0.7954	**0.7439**	0.8115	0.9555
	G	0.5075	**0.2617**	0.4968	0.9786
MNIST digits 0&1	L	0.7598	**0.6832**	0.7596	0.9460
	G	0.6522	**0.6150**	0.6651	0.9387
MNIST digits 3&8	L	0.7591	**0.7358**	0.7807	0.9298
	G	0.8214	**0.8076**	0.8355	0.9626
MNIST digits 1&5&8	L	0.7855	**0.7286**	0.7933	0.9539
	G	0.8340	0.7741	**0.7416**	0.9737
Oleg's faces	L	**0.2641**	0.3288	0.8134	0.6677
	G	**0.4898**	0.6073	0.4967	0.8747
Olga's faces	L	0.3717	**0.2714**	0.8121	0.6534
	G	0.8290	0.6027	**0.4164**	0.7752

For data sets with label information, two more characteristics are measured: classification rate reduction and clustering performance. The corresponding results for these estimations are represented in Tables 10 and 11. One can see from the results that in most cases LLE obtains better classification performance than the other methods. This fact leads to the suggestion of using LLE as a preprocessing step for dimensionality reduction before data classification[3].

[3]Chapter 6 describes the application of LLE for classification

Table 10. Classification rate reduction (R) calculated for the different data sets. The best values are bold.

Data	LLE	Isomap	SOM	S-LogMap
KNN = 1				
iris	0	**-0.0139**	0.0208	0.0347
wine	**0.0465**	0.1679	0.0469	0.3511
natural textures	0.4978	0.4104	0.4866	**0.4043**
wood	0.3733	**0.3527**	0.3988	0.3644
MNIST digits 0&1	0.0018	**0.0009**	**0.0009**	0.1355
MNIST digits 3&8	**0.0850**	0.1121	0.0855	0.2924
MNIST digits 1&5&8	**0.0034**	0.1028	0.0617	0.6090
KNN = 3				
iris	0	**-0.0280**	0.0069	0.0345
wine	**-0.0690**	0.0339	0.0238	0.0984
natural textures	0.4537	0.3597	**0.3281**	0.3565
wood	**0.3209**	0.3493	0.4010	0.6742
MNIST digits 0&1	**0**	0.0009	0.0009	0.1348
MNIST digits 3&8	**0.0659**	0.0801	0.1053	0.4624
MNIST digits 1&5&8	**0.0034**	0.0812	0.0555	0.5459
KNN = 5				
iris	**-0.0069**	0	0.0138	0.0276
wine	-0.0569	-0.0083	**-0.1111**	0.1074
natural textures	0.3783	0.2925	0.2984	**0.2864**
wood	0.2640	0.2657	0.3049	**0.1996**
MNIST digits 0&1	**0**	0.0027	0.0018	0.1356
MNIST digits 3&8	**0.0627**	0.0753	0.0789	0.2942
MNIST digits 1&5&8	**0.0014**	0.0686	0.0519	0.5349

From the point of view of cluster organization, the leader is SOM. In the case of iris data, SOM even obtains better clustering than in the original space (Table 11). This is an expected fact, since initially SOM was designed to perform good clustering of data. The LLE, Isomap and S-LogMap methods behave similarly. They do not obtain a perfect clustering, but still their results are not distinct from those of SOM.

The overall observations are as follows:

- Isomap preserves data topology in the projected space better than LLE, SOM and S-LogMap.
- LLE projections lead to the better classification rate than the other methods.
- SOM shows the best clustering organization of high-dimensional data points in a low-dimensional space.
- S-LogMap deals well with data sets of low original dimensionalities, smooth manifold structure and uniformly distributed data points (no outliers and noise).

Table 11. Clustering performance coefficient for the original (C_x) and projected (C_y) data sets. The best values are bold.

	C_x	C_y			
Data		LLE	Isomap	SOM	S-LogMap
iris	0.1680	0.3972	0.2482	**0.1346**	0.2335
wine	0.0757	0.4461	0.3491	**0.2660**	0.3593
natural textures	0.0051	0.4079	0.3362	**0.2621**	0.4913
wood	0.0010	0.4699	0.4799	0.4645	**0.4145**
MNIST digits 0&1	0.0012	**0.1287**	0.3600	0.3988	0.4153
MNIST digits 3&8	0.0013	0.4279	0.4362	**0.3990**	0.4696
MNIST digits 1&5&8	0.0011	**0.3548**	0.3817	0.4311	0.4992

4.5 Summary

Unsupervised learning is widely used for data mining, which includes such tasks as clustering, dimensionality reduction, discovering regularities, outlier detection, visualization etc. Unsupervised means that there is no knowledge about the data on hand (labels, target outputs etc.), hence data processing should be performed by a method which does not employ label information as an input. LLE, Isomap, SOM and S-LogMap are such methods. Moreover, they are suitable for manifold learning task, the goal of which is to discover a low-dimensional data manifold from high-dimensional data.

This chapter represents a comparison of the performance of the LLE, Isomap, SOM and S-LogMap methods applied for a manifold learning, where the task is to answer the following questions: i) How well do these methods preserve the topology of the original data in the obtained low-dimensional representation? ii) What is the clustering structure they can obtain in the projected space? How good is it? iii) How well can a simple classifier work in the reduced space? The experimental part shows that these four algorithms behave similarly, while each of them has particular benefits over the others. Thus, LLE achieves better classification performance in the reduced data space; Isomap better preserves original data topology in the projected data; SOM shows the best clustering organization of data points in the low-dimensional representation; S-LogMap demonstrates results comparable with those obtained for other methods only in the case of well-organized manifolds of data sets that are mostly artificially generated.

5 Quantitative evaluation of the locally linear embedding based visualization framework

Humans are not able to perceive the intrinsic structure of high-dimensional data without the use of some visualization technique that serves to produce a one, two or three-dimensional plot of the data. Visualization is a data mining tool providing humans with the ability to understand, analyze, explore the inherent information (structure, specificity, clustering tendency etc.) of high-dimensional data, and can assist them in identifying regions of interest and appropriate parameters for more focused quantitative analysis. In an ideal system, visualization harnesses the perceptual capabilities of the human visual system (Grinstein & Ward 2002).

Visualization can be used for the following tasks: i) to explore data, ii) to confirm a hypothesis, or iii) to manipulate a viewer. In exploratory visualization, the user is searching for structure or trends, and is attempting to achieve some hypothesis, which is not necessarily known in advance. In confirmatory visualization, the user has a hypothesis that needs to be tested. Here, the scenario is more stable and predictable than in the previous case. Therefore, some parameters for the visualization system can be defined a priori. In production visualization, the user has a validated hypothesis and knows exactly what is to be presented, and focuses on refining the visualization to optimize the presentation. This is the most stable and predictable of visualization (Grinstein & Ward 2002).

Visualization techniques can focus on either geometric or symbolic representation. The former case includes such kinds of data representation as scatter plot, lines, surfaces, or volumes. Here, the data typically has several fields that can be mapped to axes for these graphs to be displayed, e.g., the data is often the result of some physical model, simulation, or computation. The latter case concentrates on representing especially non-numeric data using pixels, icons, arrays, or graphs (networks, relations etc.). The focus here is on displaying the relationships between the data elements.

Another key characteristic of the visualization techniques is whether they present static or dynamic data. Depending on this characteristic the visualization system may need a number of interactions with a user.

This chapter presents a new visualization framework, which uses LLE as its intermediate component. This framework achieves a one, two or three-dimensional representation of high-dimensional data while trying to eliminate an enormous loss of information content of the data.

5.1 Visualization of high-dimensional data

Over the last decades there has been increasing interest in visualization and numerous approaches to solving the high-dimensional data visualization problem have been proposed and elaborated, but it still remains a challenging and rewarding task for scientists. Visualization methods can be divided into different groups possessing particular properties of obtained representations (Ward 1994, Bentley & Ward 1996, Grinstein & Ward 2002). The first group includes algorithms which construct multiple views of the data using scatterplots or thematic maps (Ahlberg & Shneiderman 1994). These are among the oldest and best known techniques, which are widely used to represent high-dimensional data with two-dimensional planes. The construction of scatterplots of the data is performed by creating $D(D-1)/2$ two-dimensional plots (D is the dimensionality of the data), each of which corresponds to one bivariate observation, i.e., a plot of the data represented by axes corresponding to any two of the D features. The obtained plots are organized in a grid structure, a so called scatterplot matrix, to facilitate the users' understanding of the result. Many variations on scatterplots are used in practice in order to increase the information content of the data visualization and to improve data exploration. These include using different symbols to distinguish classes of the data and presence of overlapping points, using color, shading etc. to represent a third dimension within each projection, rotating the data cloud etc.

Figure 16 depicts a four-dimensional iris data set (for the data set descriptions, see Appendix 1) using a scatterplot matrix, where each row and column corresponds to a particular feature of the data. The plots on the intersections of rows and columns depict the correlations of the corresponding features. Note that the diagonal plots provide distribution information on the individual dimensions.

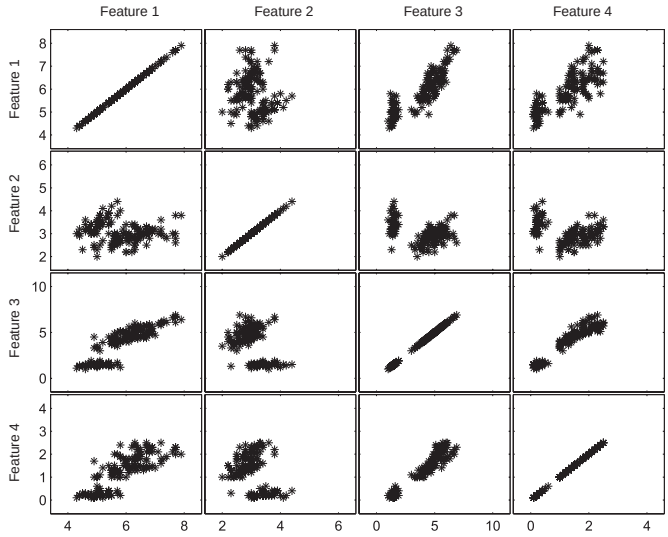


Fig. 16. Example of scatterplot.

A major limitation of the scatterplot method is that it is effective if and only if the data has a small number of dimensions. In a case where the data dimensionality increases, the number of elements in the scatterplot matrix also increases, leading to a cumbersome and incomprehensible visualization. To overcome this limitation, three-dimensional plots may be used instead of two-dimensional ones, but this replacement does not completely solve the problem. Other limitations are the restriction of scatterplots to orthogonal views and difficulties in discovering relationships that span more than two dimensions. The advantages of the scatterplots are insensitivity to the number of data samples and the simplicity of the data explanation.

The second class of visualization methods is based on glyphs. The definition of glyph or iconic visualization involves a large number of methods representing the high-dimensional data with different graphical objects whose attributes, such as size, shape, color, orientation, 3D size, etc. are bound to the data (Ribarsky *et al.* 1994, Post *et al.* 1995). There are many variants of glyph representations, some of them are:

- Scalar or vector glyphs. Scalar (1D) glyphs convey a single value, while vector (2D) glyphs are used to depict flow information. In the former case, the data information is associated with, for example, the size of a cube or radius of a sphere. In its turn, arrows are the most prevalent form of glyphs for the latter situation (Grinstein & Ward 2002). Figure 17 demonstrates 2D glyphs.
- Faces, where different data dimensions control graphical attributes of the icons specified by size, location and shape of facial features. The usage of such visualization was first proposed by Herman Chernoff (Chernoff 1973). One can see from Figure 18 (a) that faces represent high-dimensional data in a discernible manner.
- Star diagrams, where each dimension of data controls the length of a ray emanating from a center point. Those spokes are placed at uniformly separated angles. The neighboring endpoints of the rays are connected to each other forming stars of different shapes, as shown in Figure 18 (b) (Ribarsky *et al.* 1994) etc.

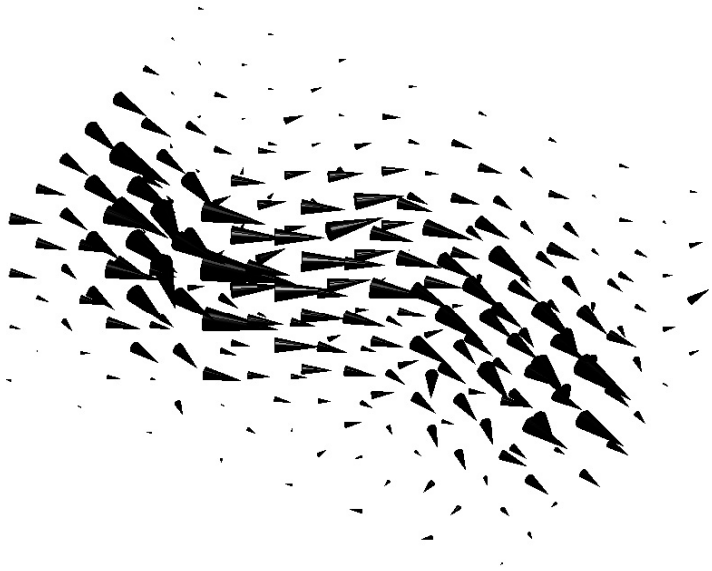


Fig. 17. **Example of vector glyphs.**

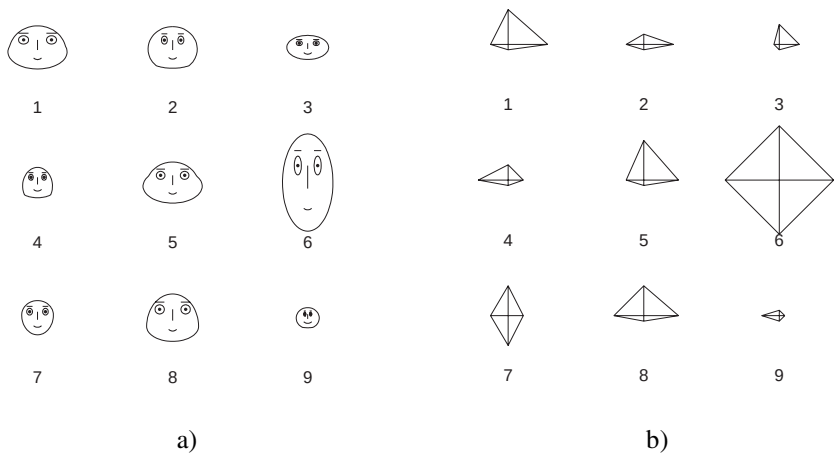


Fig. 18. **9 samples from the iris data depicted by a) faces and b) stars.**

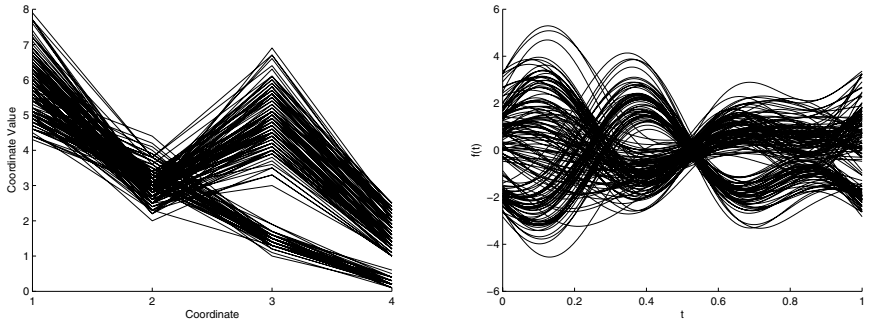
The characteristics of glyph visualization are as follows: i) It is easy to compare, classify and sort glyphs due to the fact that their structures allow the user to display and correlate several features at once; ii) Glyphs are flexible but limited to the number of features and samples of the data which can be displayed simultaneously; iii) The density and size constraints of glyphs depend on the perceptual accuracy required; iv) It is difficult to

compare glyphs separated in space.

The third group, called parallel coordinates, differs from all other visualization approaches, since it utilizes a point-to-line duality instead of the conventional point-to-point one. Hence, parallel coordinates yield graphical representation of multidimensional relations rather than finite point sets (Inselberg & Dimsdale 1990). This method produces a two-dimensional plot with D uniformly spaced parallel axes corresponding to the features of D-dimensional data. In this plot, a high-dimensional data point appears as a line connecting a set of points (one on each axis) associated with particular feature value. Figure 19 (a) demonstrates parallel coordinate visualization for the iris data (for the data sets description, see Appendix 1).

The display obtained with parallel coordinate method shows some properties of the data and supplies information on the proximity of the data points to a boundary. The major limitation of the parallel coordinates technique arises in data interpretation when the number of points is very large. In other words, lots of points can organize a clutter on the visualization plane because of a point-to-line duality. Also, relationships between adjacent axes are easier to understand than between distant axes.

Another similar type of multivariate visualization that relates to the parallel axes is Andrews plot (Andrews 1972). This plot represents each point as a smooth function placed over the interval [0,1]. Figure 19 (b) demonstrates an example of the Andrews plot obtained for the iris data. Here each function is a Fourier series with coefficients equal to the corresponding observation's values. Another possibility is to use, for example, trigonometric function (Ward 1994).



a) b)

Fig. 19. **Iris data visualized by a) parallel coordinates and b) Andrews plot.**

Hierarchical techniques introduce the forth class of visualization tools. They suppose that a single two-dimensional visual plane representing projection of complex data living in high-dimensional space cannot describe the structure and specificities of the data (Bishop & Tipping 1998). Hierarchical visualization algorithms generate several two-dimensional plots. The top-level projections display the whole data set, while the lower-level projections display an internal structure of clusters in order to reveal sub-clusters and groups of outliers which might not be visible in the higher-level plots. The usage of this class of visualization algorithm compensates for the lack of the individual model flexibility by the general flexibility of the complete hierarchy (Bishop & Tipping 1998). The main prob-

lem with hierarchical methods is the difficulty to determine spatial relationships between points in non-adjacent dimensions (Ward 1994).

The next visualization approach is based on dimensionality reduction techniques which project a set of points belonging to the high-dimensional data sets into an equal number of points in one, two or three-dimensional spaces. A number of linear and nonlinear dimensionality reduction algorithms are described in the literature: principal component analysis (PCA) (Hotelling 1933, Jolliffe 1989), linear discriminant analysis (LDA) (Belhumeur *et al.* 1997, He *et al.* 2005), multidimensional scaling (MDS) (Bentley & Ward 1996), self-organizing map (SOM) (Kohonen 1997), curvilinear distant analysis (CDA) (Lee *et al.* 2000) etc. All these methods merely project high-dimensional data to two- or three-dimensional space while trying to preserve the data topology. By doing this, some information contained in the data is lost, especially if linear dimensionality reduction methods, which cannot discover the intrinsic structure of nonlinear manifolds, are utilized. Therefore, nonlinear feature extraction and selection algorithms are more frequently used for data visualization. However, even these nonlinear methods can suffer from two problems. (1) Dimensionality reduction depends on a metric, i.e., if this metric does not properly capture the similarity of data, visualization results are poor (Peltonen *et al.* 2004, Peltonen & Kaski 2005). (2) Since the dimensionality of the original (unprocessed) data is typically large, determining correct similarity relationships in high-dimensional space and preserving them in low-dimensional (visualization) space can be difficult because of dramatic dimensionality reduction.

We propose an LLE based visualization framework, which can remedy these two problems. Its main features are that (1) it uses metric learning to complement dimensionality reduction, so that non-Euclidean similarity relationships of data can be adequately captured, and (2) it reduces the data dimensionality via a two-level scheme, where the original dimensionality is first reduced to the intrinsic dimensionality, which afterwards is reduced to two or three, so that changes in the data dimensionality are softened, thus preventing severe topology distortions during visualization.

Thanks to the fact that metric learning normally needs only a fraction of the whole data to be labeled (the precise amount is data-driven), the proposed framework occupies an intermediate place between unsupervised and supervised visualization techniques, i.e., it can be called *semi-supervised*. Usually the unsupervised methods cannot take advantage of the benefits of metric learning, since no labels are provided, whereas the supervised techniques may not need metric learning, since they rely completely on labels instead. However, it is quite rare that labels are known for the whole data set, because it is laborious and time-consuming to attach labels, especially if performed manually. Labels are rather available for a limited (and often small) fraction of the whole data set. This fact makes the semi-supervised framework a new and useful tool in data exploration.

5.2 The visualization framework

The visualization framework described here assumes that extracting two or three features by applying a dimensionality reduction technique to the high-dimensional data is not sufficient to obtain a good visualization. Therefore, it consists of a combination of four

components: metric learning, intrinsic dimensionality estimation, feature extraction and feature selection. Each of these components is intended to enhance the final visualization by performing a particular task (Kouropteva *et al.* 2004a,b, 2005c).

Given labels for the fraction of data, the task of the first step is to learn an appropriate metric function over the high-dimensional space (Kaski & Sinkkonen 2000, Xing *et al.* 2003, Zhang *et al.* 2004). This allows visualizing the same data in different ways depending on the assigned data labels (e.g., data of articles can be labeled either by a keyword or authors' names), which is not possible if the Euclidean metric is used. By using prior information as above, a metric learning procedure automatically obtains the distance function that assigns small/large distances to pairs of similar/dissimilar points. The metric function found is used in the next steps of the framework: distances, calculated over the input space by using this function, improve the abilities and the performance of the intrinsic dimensionality estimation and feature extraction/selection algorithms, and therefore positively affect the final result.

Real-world data sets, which are typically high-dimensional, do not completely fill the spaces where they reside. Hence, we assume that data points lie on a manifold of lower dimensionality embedded into a high-dimensional space. The local intrinsic dimensionality of the data embedded in a high-dimensional space, d_L, refers to the minimum number of independent parameters needed to represent the data (Bruske & Sommer 1998). While the global one, d_G, is equal to the dimensionality of the space filled by data points. Figure 20 gives an illustrative example of these definitions: a one-dimensional curve fills three-dimensional space, hence for this situation $d_L = 1$ and $d_G = 3$. To calculate d_L, an intrinsic dimensionality estimation algorithm using packing numbers is utilized in the second step of the framework (Pettis *et al.* 1979, Kégl 2003, Appendix 3). A PCA based approach is used to estimate d_G (Hotelling 1933, Appendix 5).
Given intrinsic dimensionality (d_L or d_G) which is usually larger than 1, 2 or 3, the visualization results are obtained by a two-level dimensionality reduction procedure including either nonlinear feature extraction followed by feature selection as suggested in (Jain *et al.* 2000), or linear feature extraction followed by nonlinear as proposed in (Kouropteva *et al.* 2005c). For simplicity of explanation let us define by VF1 and VF2 the former and the latter variants of the visualization framework, correspondingly:

- **VF1:** A nonlinear feature extractor obtains a low-dimensional representation of the original data by constructing new d_L features reflecting important data information. Since d_L is usually larger than 2 or 3, it is difficult to define which pair of features should be used to visualize the data (Saul & Roweis 2003). In order to solve the problem, a feature selector is used to choose two or three of the most informative features for the final visualization. In this case, the desirable combination is picked out from the $\frac{d_L!}{(d_L-v)!v!}$ possible choices, where $v = 2$ or 3. Thus, there is a richer choice of target dimensions in VF1 than in the conventional feature extraction technique used for visualization. Of course, one would have a richer choice in a high-dimensional space, but in this case the computational demands would dramatically increase. By lowering the data dimensionality from several hundreds or even thousands, which is typical if intensity values of image pixels are used as features, to a couple of dozen at most, these demands are significantly reduced and a sufficient number of features are left to choose from. The flow chart of VF1 is shown in Figure 21 (a).
- **VF2:** In the second alternative, the reasoning is as follows. Suppose, the global

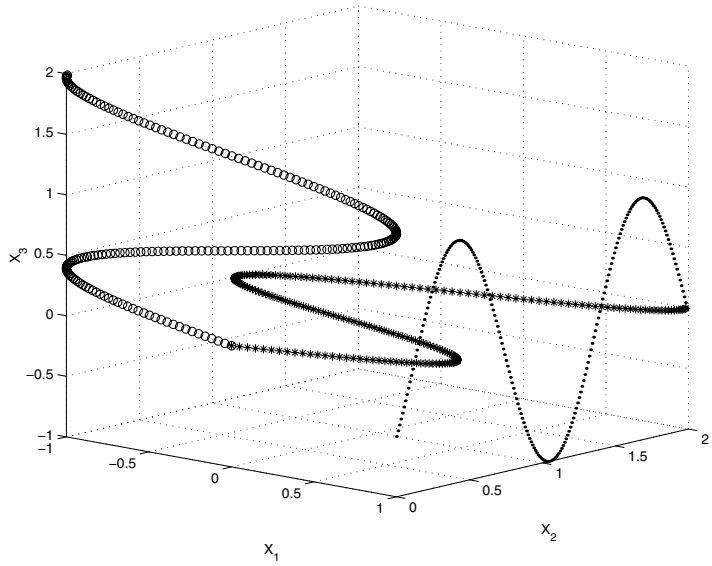


Fig. 20. **Example of a 1D curve embedded into 3D space. Different point markers correspond to the parts of curve belonging to a particular plane.**

data's intrinsic dimensionality is found to be equal to d_G. Without loss of generality one can assume that the data resides in the d_G-dimensional manifold, in the same way as in high-dimensional space. Hence, a linear feature extraction method can be used in the third step to reduce the data dimensionality to d_G without losing valuable information. Moreover, by applying this step, irrelevant and noisy information is removed from the data. Finally, if d_G is larger than the dimensionality of the visual space, we need, a one, two or three-dimensional visualization space is obtained by applying a nonlinear feature extraction method to the d_G-dimensional data [1]. Figure 21 (b) demonstrates the flow chart of VF2.

If a linear feature extraction algorithm were used alone to visualize the data, i.e., to reduce dimensionality to two or three, the resulting data representation might not always discover the intrinsic structure and specificities of the data. In its turn, if the data is visualized by nonlinear feature extraction alone, the number of operations to be executed grows dramatically, since it depends on the dimensionality of the original data space. These harmful effects are smoothed out by the proposed two-level dimensionality reduction scheme of VF2.

The main advantage of the framework is the possibility to choose any suitable algorithm performing metric learning, intrinsic dimensionality estimation, feature extraction and feature selection. Thus, a user can explore different combinations of the algorithms and select the best one (Kouropteva *et al.* 2004a,b, 2005c).

[1] Note that in respect to this thesis, feature extraction refers to the forming of new features by transforming and combining the original feature set, while feature selection refers to the selecting of a particular feature out of the given ones.

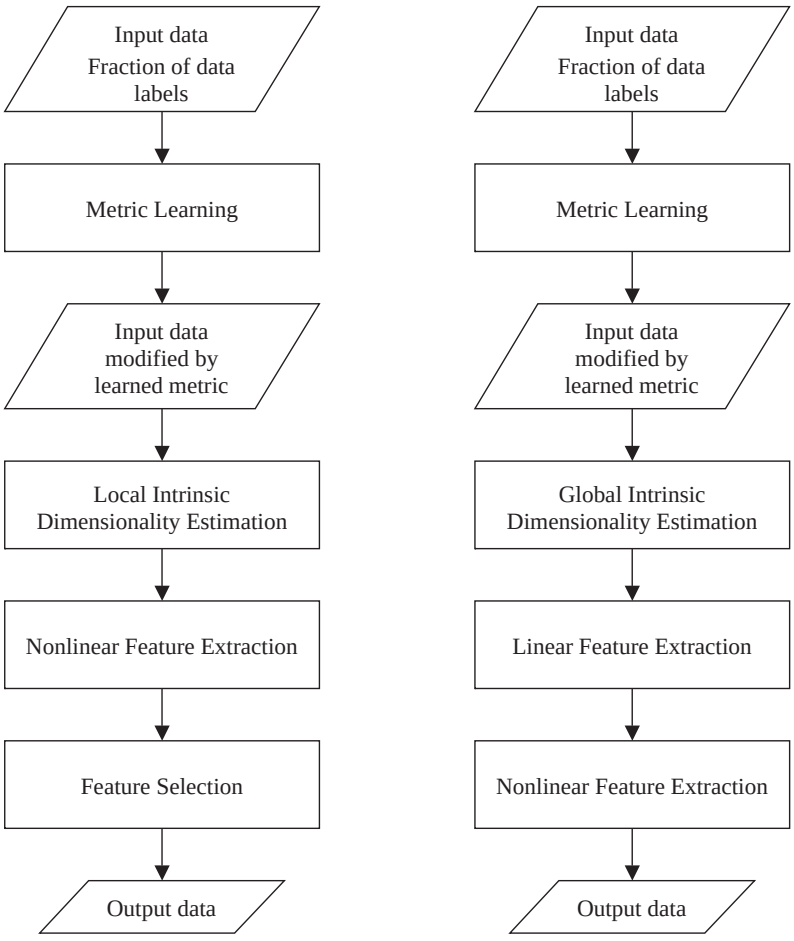


Fig. 21. **Two variants of the visualization framework: a) VF1, b) VF2.**

5.3 Experiments

In order to investigate the performance of the proposed framework, the following algorithms were chosen for VF1 and VF2:

VF1:

Step 1 Metric learning algorithm proposed by Xing (Xing *et al.* 2003)
Step 2 Intrinsic dimensionality estimation using packing numbers (Kégl 2003)
Step 3 LLE (Roweis & Saul 2000)
Step 4 Maximization of the joint mutual information (Moddemeijer 1989, Cover & Thomas 1991)

VF2:

Step 1 Metric learning algorithm proposed by Xing (Xing *et al.* 2003)

Step 2 PCA-based intrinsic dimensionality estimation (Hotelling 1933)
Step 3 PCA (Hotelling 1933, Jolliffe 1989)
Step 4 LLE (Roweis & Saul 2000)

Both variants of the framework are applied to five different data sets. Table 12 describes some of the properties of the data sets: original dimensionality (D), number of samples, and the number of different classes (for the data sets description, see Appendix 1).

Table 12. Short description of data sets used in the experiments.

Data	D	# of samples	# of classes
liver	6	345	2
wine	13	178	3
vehicle	18	846	4
MNIST digits 0&1	784	2,115	2
MNIST digits 3&8	784	1,984	2

First, the metrics are learned for all data sets by using only part of the data labels: half of the labels for the liver, wine, vehicle and digits data sets. For the MNIST data, we manually choose the 100 most representative samples from each class and learn the metric function based on this information. Then, the intrinsic dimensionalities of the data sets are estimated using the corresponding learned metrics and the obtained results are represented in Table 13[2]. The use of the metric learning function in the following steps of the visualization framework is done by multiplying the data matrix by scaling elements found with the algorithm of Xing et al. Note that this approach fits only for the particular metric learning algorithm.

Table 13. Local (d_L) and global (d_G) intrinsic dimensionalities estimated for the data sets modified by the corresponding metric.

Data	d_L	d_G
liver	3	3
wine	3	2
vehicle	6	1
MNIST digits 0&1	7	15
MNIST digits 3&8	8	18

Two last steps of the visualization framework are explained below for the example of the wine data. In the first variant (VF1), the dimensionality of the wine data is reduced to 3 by applying LLE to the data with a learned metric, and joint mutual information is estimated for all pairs of the obtained features: 0.0614 (first-second features), 0.0895 (first-third features) and 0.0372 (second-third features). The corresponding two-dimensional plots are given in Figure 22 (a,b,c). The visualization plane shown in Figure 22 (b) is the result

[2]The intrinsic dimensionalities of some data sets in Table 13 differ from those described in Table 5 of Chapter 3 and Table 17 of Chapter 6, since the metrics used in the original data spaces are different (learned metric vs. Euclidean).

of VF1, since the first and third features give the highest value of mutual information. Indeed, one can see that this visual plane better represents a clustering structure of the data.

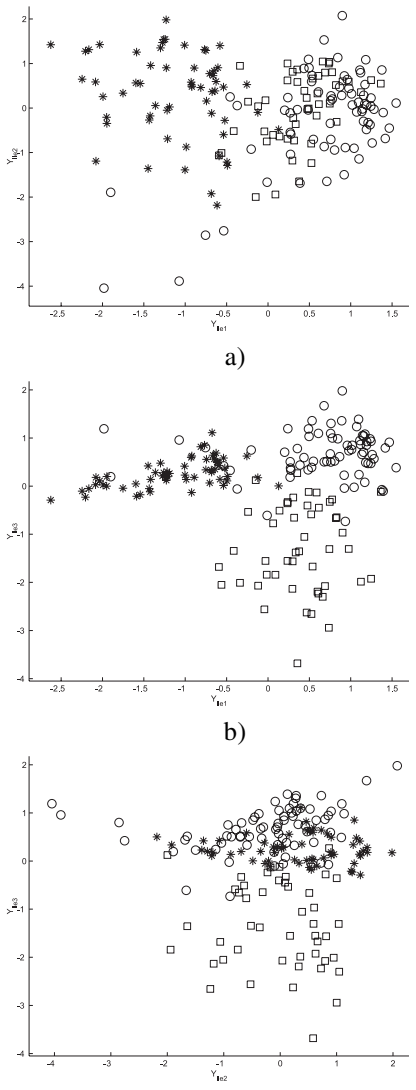


Fig. 22. Visualization planes represented by all possible pairs of the features obtained with the third step of VF1 (LLE) for the wine data: (a) first-second, (b) first-third, (c) second-third. The final result of VF1 corresponds to the plot (b).

For a quantitative estimation of the clustering structure, the measures called classification rate reduction R with K nearest neighbors optimized by a leave-one-out procedure and clustering performance C (Section 4.3) are calculated for each pair of features obtained with LLE. Table 14 demonstrates these estimations; the best results are shown in bold. One can see that the best values R and C correspond to the combination of features chosen for visualization by the feature selector. The experiments with other data sets confirm this fact (Kouropteva *et al.* 2004a).

Table 14. Classification rate reduction (R) with K nearest neighbors optimized by a leave-one-out procedure and clustering performance (C) for all pairs of features obtained with the third step of VF1 (LLE) for the wine data. Best values are shown in bold.

	1 vs. 2	1 vs. 3	2 vs. 3
R	0.036	**-0.199**	-0.015
C	0.46	**0.40**	0.46

The third step of the second variant of the visualization framework (VF2) uses PCA and a learned metric to reduce the dimensionality of the wine data to 2 (d_G). Figure 23 (a) demonstrates the final result for VF2. Since two-dimensional representation is obtained, we do not need to perform the fourth step of VF2. But this is not the case for the digits data sets ($d_G > 3$) and for the liver data, if visualization of dimensionality one or two is desired.

In order to show that the proposed framework obtains better data visualization than a dimensionality reduction algorithm used alone for this purpose, the PCA and LLE algorithms are applied to the data sets. Figure 23 (b) and (c) show the PCA and LLE visualization planes for the wine data. One can see from the pictures that for the wine data the results obtained with the framework are visually comparable with those found for PCA and LLE. To quantitatively discriminate between these results, the classification rate reduction R with K nearest neighbors optimized by the leave-one-out procedure, and clustering performance C are calculated. Table 15 represents these values calculated for the wine data set; the best values are shown in bold.

Table 15. Classification rate reduction (R) with K nearest neighbors optimized by the leave-one-out procedure and clustering performance (C) of the resulting visualization spaces obtained with VF1, VF2, PCA and LLE for the wine data set. The best values are shown in bold.

	VF1	VF2	PCA	LLE
R	**-0.199**	-0.015	0.058	0.037
C	0.400	**0.352**	0.353	0.497

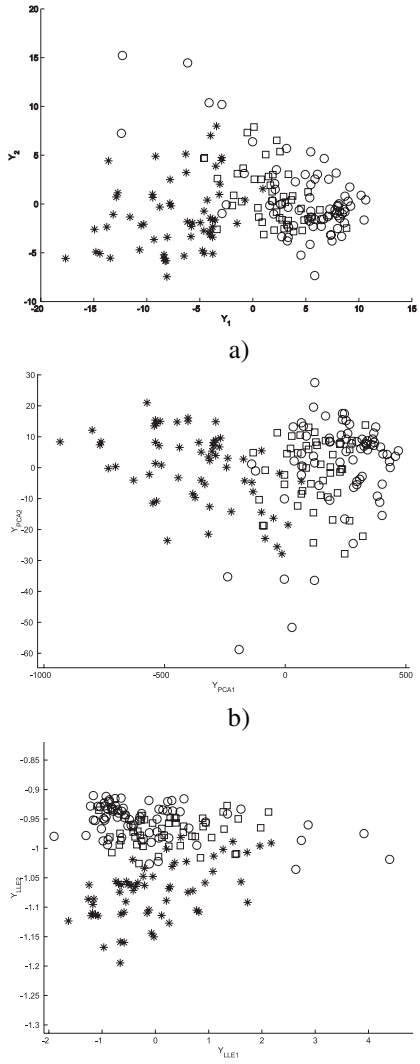


Fig. 23. Visualization planes obtained with (a) VF2, (b) PCA and (c) LLE for the wine data.

The same operations are carried out for the liver, vehicle, and digits data sets. Table 16 represents the estimations R and C calculated for the resulting visualization planes obtained for these data sets. According to the results, the visualization framework gives better data representations providing humans with an ability to more easily discriminate between data classes.

Table 16. Classification rate reduction (R) with K nearest neighbors optimized by the leave-one-out procedure and clustering performance (C) of the resulting visualization spaces obtained with VF1, VF2, PCA and LLE for the liver, vehicle and digits data sets. The best values are shown in bold.

		VF1	VF2	PCA	LLE
Liver	*R*	0.025	0.096	0.092	**0.017**
	C	**0.324**	0.488	0.497	0.488
Vehicle	*R*	0.186	**0.184**	0.186	0.245
	C	**0.495**	0.788	0.788	0.907
MNIST digits	*R*	**0**	**0**	0.001	**0**
0 & 1	*C*	0.238	**0.234**	0.352	0.238
MNIST digits	*R*	**0.008**	0.041	0.112	0.019
3 & 8	*C*	0.375	**0.323**	0.375	0.338

5.4 Summary

Visualization of high-dimensional data is one of the directions of data mining. Its goal is to visually represent the data structure in a way that is informative and understandable for humans. Two variants of the semi-supervised visualization framework described in this chapter are designed for the same purpose. Both variants contain four components, the first two of which are the same: metric learning and intrinsic dimensionality estimation. Metric learning followed by intrinsic dimensionality estimation helps to discover the true information about the data at hand. Additionally, the intrinsic dimensionality estimation procedure ensures that the minimum of useful information is lost during the following two steps (feature extraction followed by feature selection in VF1 and linear feature extraction followed by non-linear feature extraction in VF2), since the number of features extracted is sufficient to evaluate and explore the data properties. As a result, representative features containing the most part of the data information are found.

The framework provides a promising tool for visualization that is better than dimensionality reduction alone, because the one, two or three-dimensional projections obtained in the latter case may not entirely describe the intrinsic data structure. One of the main novelties of the framework is its semi-supervised approach to high-dimensional data visualization, which lies in utilizing metric learning on the partly labeled data in order to preserve non-Euclidean similarity relationships between the data points. The second new feature is that it uses a two-level scheme to produce the final visualization space. It makes the changes in the data during the dimensionality reduction process insignificant and it

thus allows avoiding severe topology distortions of the visualized data. Additionally, the framework is flexible to the substitution of components, i.e., each of them can be replaced by any method performing the corresponding operations. Hence, different combinations can be explored in future studies.

6 The locally linear embedding algorithm in classification

Unsupervised or supervised classification is a primary goal of pattern recognition. The difference between these kinds of classification is that the data labels are not predefined in the former case, and they are provided as an input in the latter one (Duda *et al.* 2001).

Unsupervised classification (or clustering) partitions the given data into a set of groups (clusters) in a such a way that the data samples belonging to one of these clusters possess properties which are different from those of others clusters. In its turn, supervised classification (henceforth referred to as classification) builds a model that captures the intrinsic associations between the class labels and the features, so that an unknown class of an unseen sample can be accurately predicted from its features. For this purpose, the data is usually divided into training and test sets, where the training set is used to build a classifier, which is validated by the corresponding test set (Wang & Yang 2005).

In many real-world classification problems, high-dimensional data sets are collected, for example, from sensors. Often, the ideal decision boundary between different classes in such sets is highly nonlinear. A classifier should therefore have many degrees of freedom, and consequently a large number of parameters. As a result, training a classifier on such data sets is quite complicated: a large number of parameters has to be estimated using a limited number of samples. This is the well-known curse of dimensionality (Section 2.1).

To eliminate these unfavorable consequences two different approaches might be applied. The first one is to use kernel-based techniques, which map the data into a higher-dimensional space, where the classes become almost linearly separable. An example of this approach is support vector machines (SVMs) (Vapnik 1995). The second possibility is to apply a dimensionality reduction technique to the data. Of course, by doing this, some information contained in the data might be lost. In spite of this, a better performance in classification might be achieved, since the number of parameters to be estimated is reduced. Many linear and nonlinear methods for performing dimensionality reduction are well-established in the literature (Chapter 2).

To extend the concept of LLE to multiple manifolds, each representing data of one specific class, two supervised variants of LLE were independently proposed in (Kouropteva *et al.* 2002a, De Ridder & Duin 2002). In this chapter, these both supervised methods are described and applied as a feature extractor on a number of benchmark data sets.

6.1 Supervised locally linear embedding

The variants of LLE described in Section 3.1 belong to the unsupervised methods. Such methods are mostly intended for data mining and visualization when the number of classes and relationships between elements of different classes are unknown and a user wants to see the data structure in order to make a decision about what to do next.

Being unsupervised, the original LLE does not make use of the class membership of each point to be projected. To complement the original LLE, a supervised LLE is proposed. Its name implies that membership information influences which points are included in the neighborhood of each data point. That is, the supervised LLE employs prior information about a task to perform feature extraction.

The question may arise: when may one find the supervised LLE useful? First of all, this is the case in off-line or batch learning, where all data has been already collected beforehand. Because a complete data set is available in this case, it is quite natural to expect that a specific and unique label is attached to each pattern. It is worth mentioning in this respect that in large databases, wrongly labeled patterns cannot be completely avoided since, as a rule, the labels are assigned by a human. However, we assume here that even if this is the case, the number of incorrectly labeled patterns is negligibly small compared to the total number of patterns in the database. Second, the supervised LLE can deal with data sets containing multiple (often disjoint) manifolds, corresponding to classes.

So far, two approaches to the supervised LLE have been proposed. The first approach (abbreviated as 1-SLLE) forms the neighborhood of a data point only from those points that belong to the same class (Okun *et al.* 2002, Kouropteva *et al.* 2003a,b). The second approach (abbreviated α-SLLE) expands the interpoint distance if the points belong to different classes; otherwise, the distance remains unchanged (De Ridder & Duin 2002). Either approach modifies only the first step of the original LLE (Section 3.1), while leaving the other two steps unchanged.

The modification of the first step is done by changing the elements of the pre-calculated data distance matrix. The distances between samples belonging to different classes are increased, but they are left unchanged if samples are from the same class:

$$\Delta = \Delta + \alpha \max(\Delta)\Lambda, \text{where } \alpha \in [0,1], \tag{25}$$

where $\Lambda_{ij} = 0$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same class, and 1 otherwise. At one extreme, when $\alpha = 0$, we get unsupervised LLE. At the other extreme, when $\alpha = 1$, we get the fully supervised LLE (1-SLLE). As α varies between 0 and 1, a partially supervised LLE (α-SLLE) is obtained.

Figure 24 illustrates how the feature extraction process depends on α. Three classes of the iris data set (Figure 24 (a)) are mapped by α-SLLE into two-dimensional space. Different values of α affect the results as follows:

- $\alpha = 0$
  In this case, the conventional LLE (see Chapter 3) is applied to the data set. Figure 24 (b) demonstrates two-dimensional LLE projection of the iris data.
- $0 < \alpha < 1$
  The value of the α chosen in this range controls the amount of supervision used by

LLE. The mapping obtained by α-SLLE preserves some of the manifold structure while it introduces separation between classes. Figure 24 (c) shows how the embedding changes with the assigning α to be equal 0.01. As can be seen from the plot, even for this small amount of supervision, two overlapping classes become linearly separable while preserving some of the class structure. This allows supervised data analysis, but may also lead to better generalization of the mapping on new, previously unseen data samples.

– $\alpha = 1$

As a result, we have the fully supervised version of LLE. Here, the distance matrix represents n fully disconnected classes, since the distances between data points belonging to different classes are equal to the maximum distance in the entire data set. In other words, K nearest neighbors of the data sample from a particular class are selected from the same class; hence, we do not have to compute Eq. 25. Therefore, 1-SLLE is a non-parameterized supervised LLE.

Figure 24 (d) demonstrates the embedding obtained with 1-SLLE, where the classes are separable due to the constraint $\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i \boldsymbol{y}_i^T = \mathbf{I}_{d\times d}$ and each of the classes are mapped into a single point in $\mathbb{R}^{n-1}$. Therefore, the optimal dimensionality of the embedding for 1-SLLE is $(n-1)$. But this is not true for $0 \leq \alpha < 1$.

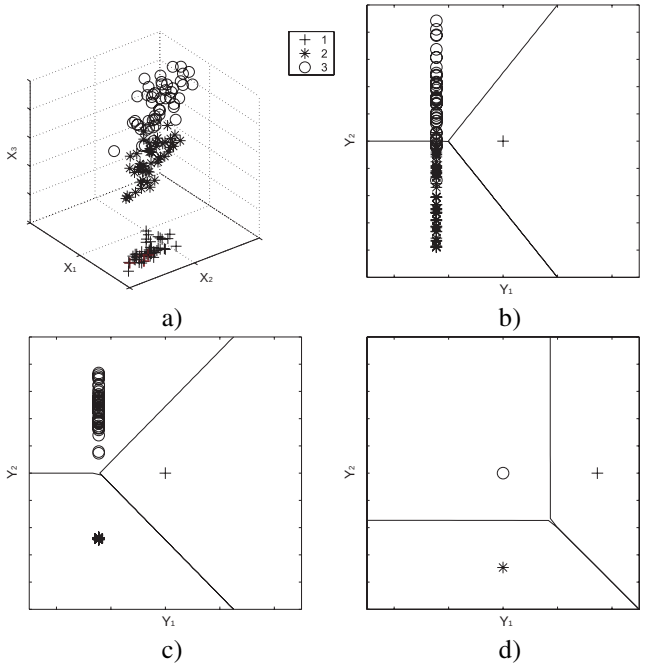


Fig. 24. (a) First, third and fourth feature of the iris data set containing 150 samples of dimensionality 4 divided into 3 classes. Two-dimensional projections obtained for the iris data set by (b) LLE, (c) α-SLLE (α=0.01), and (d) 1-SLLE algorithms with nearest mean classification boundaries.

The idea behind SLLE is related to that of spectral clustering (Weiss 1999). There, first an affinity matrix between all samples is calculated. If clusters are present, this matrix will have a block-diagonal structure. An eigen-decomposition of the (normalized) affinity matrix then gives an embedding in a small number of dimensions in which clusters are more clearly separated than in the original space. SLLE uses class label information to construct an artificial off-diagonal block matrix Λ, and applies this to change the distance matrix used as the basis for LLE. The resulting matrix $\mathbf{W}$, which is already sparse (containing only K non-zero entries in each row), is changed towards a block-diagonal matrix. As a result, a mixture between unsupervised LLE and supervised spectral clustering is obtained (De Ridder *et al.* 2003a).

6.2 Experiments

In the following experiments[1], both variants of SLLE are applied to a number of data sets varying in the number of samples N, dimensions D and classes n. A brief description of the data sets is given in Table 17 (for more detailed data description, see Appendix 1). The table also lists the global, d_G, and local, d_L, intrinsic dimensionalities of the data sets. The global dimensionality for a particular data set is equal to the number of features obtained by PCA which retain 90% of the variance (Hotelling 1933), whereas the local one is found by demanding that locally 90% is retained in the remaining number of dimensions (De Ridder & Duin 2002, Appendix 5).

Table 17. Short description of data sets used in the experiments.

Data	D	# of samples	# of classes	d_G	d_L
iris	4	150	3	1	3
diabetes	8	768	2	2	4
glass	9	214	6	4	3
wine	13	178	3	1	2
vehicle	18	846	4	1	5
hepatitis	19	80	2	2	3
chromosomes	30	2,520	24	8	8
ionosphere	34	351	2	18	4
splice	60	3,188	3	51	18
sonar	60	208	2	12	8
optdigits	64	5,620	10	21	10
natural textures	144	3,000	6	33	6
structured textures	144	3,000	6	33	17
NIST digits	256	6,250	10	52	12
paper	857	1,004	4	2	7

[1] The author carried out the experiments only for the paper data set. The results are published here with Dick De Ridder's authorization.

The experiments are set up as follows: a data set is randomly split 10 times into training and test sets containing 80% and 20% of the data samples, correspondingly. The data sets are projected to d_L-dimensional spaces by PCA (Hotelling 1933, Jolliffe 1989), LDA (Belhumeur *et al.* 1997, He *et al.* 2005), MDS (Cox & Cox 1994), LLE Roweis & Saul (2000), α-SLLE, and to $(n-1)$-dimensional space by 1-SLLE. Mappings were calculated for a range of values of K, the neighborhood size parameter, and α (if applicable). The first three techniques are used in order to compare (S)LLE methods to more traditional feature extractors.

PCA, LDA, MDS and LLE project all data points to the low-dimensional space, whereas α-SLLE and 1-SLLE first project the training data sets into low-dimensional spaces, and then map the corresponding test data points by the generalization procedure LG2 (Subsection 3.2.3.1). Note that the generalization procedure does not use label information, hence the test data points do not have labels. The following classifiers are trained in the low-dimensional spaces, and the corresponding test data points are classified (Duin 2004):

- **Nearest mean classifier** (NMC) is a simple classification algorithm that summarizes the sample data from each class using the class mean vector, or centroid, $\bar{\boldsymbol{x}} = \frac{1}{N_l}\sum_{j=1}^{N_l}\boldsymbol{x}_j$, where $\boldsymbol{x}_j$ is the j^{th} feature vector from class l containing N_l vectors. An unknown object is classified as a class l if it is much closer to the mean of this class than to any other class mean vector. This classification method is simple and fast, and will work in some problems where the feature vectors from each class are compact and far from those of the other classes. Difficulties can arise when the structure of the class samples is complex, e.g., multimodal, elongated, nested (Duda *et al.* 2001).
- **K-nearest neighbor classifier** (KNN), with K optimized by the leave-one-out procedure on the training set. KNN is a more flexible method of classification. It classifies an unknown sample into the class containing more individual samples closest to the new one. The nearest-neighbor classifier can be effective even when classes have a complex structure or overlap. No assumptions need to be made about the models for the distribution of feature vectors in space; the algorithm uses only existing training samples (Duda *et al.* 2001).
  The leave-one-out procedure is an N-fold cross-validated estimator. A single observation is left out and a classifier is designed from the remaining part of the samples. Then, the built classifier is tested on the left sample. The classification error is estimated by averaging all N errors obtained for each sample (Duda *et al.* 2001).
- **Bayes plug-in (linear and quadratic) classifier** (LDC and QDC, correspondingly). The Bayes plug-in classifier, also called the Gaussian maximum likelihood classifier, is one of the most common parametric methods applied to statistical pattern recognition systems. It is based on similarity measures that involve the inverse of the true covariance matrix of each class or group. In practice these matrices are not known, hence estimates must be computed based on patterns available in training set. The ordinary choice for estimating the true covariance matrix is the maximum likelihood estimator defined by its corresponding sample group covariance matrix (Thomaz 2004).

Tables 18 and 19 present average errors on the test set (in %) over the 10 random set splits, with the standard deviation given between brackets. The best result is shown in bold and

underlined; all results not significantly worse (paired t-test with 9 degrees of freedom, p = 0.95) are shown in bold as well. For 1-SLLE and α-SLLE, only the best result found in the range of values for K and α is shown. Ideally, these optimal values should be found on an independent validation set, but the size of many of the data sets did not permit setting aside samples for this.

Table 18. Test error (in %), low-dimensional data sets.

	NMC	KNNC	LDC	QDC
iris				
Original	7.7 (2.7)	2.3 (1.6)	**1.7** (2.4)	3.3 (3.1)
PCA	9.3 (4.1)	3.3 (3.1)	4.0 (3.1)	4.3 (3.5)
LDA	**1.7** (2.4)	**3.0** (3.3)	**1.7** (2.4)	3.3 (3.5)
MDS	8.7 (4.5)	3.7 (1.9)	4.7 (3.6)	5.0 (3.6)
LLE	4.7 (3.9)	**2.3** (2.7)	**1.0** (1.6)	**2.3** (2.2)
1-SLLE	3.3 (3.1)	3.3 (3.1)	12.0 (6.1)	23.0 (27.7)
α-SLLE	**1.7** (2.4)	**2.3** (2.2)	**1.0** (1.6)	**2.3** (2.2)
wine				
Original	25.3 (5.3)	24.4 (5.4)	**1.1** (1.4)	**1.1** (1.4)
PCA	27.2 (7.9)	34.2 (6.3)	32.8 (6.1)	27.8 (8.5)
LDA	**1.1** (1.4)	**1.1** (1.4)	**1.1** (1.4)	**1.4** (1.5)
MDS	27.2 (7.9)	34.7 (5.9)	34.2 (6.1)	28.1 (8.1)
LLE	25.3 (5.3)	24.7 (6.2)	26.7 (5.3)	25.8 (6.4)
1-SLLE	4.7 (3.2)	4.7 (3.2)	41.1 (17.2)	46.4 (11.0)
α-SLLE	5.8 (3.6)	11.4 (3.8)	15.6 (5.1)	16.4 (5.1)
diabetes				
Original	34.5 (4.0)	24.4 (2.6)	**22.2** (1.9)	24.9 (1.9)
PCA	39.2 (2.2)	32.9 (2.9)	34.5 (1.2)	34.3 (2.0)
LDA	24.2 (2.9)	**23.1** (2.5)	**22.8** (2.3)	**22.5** (2.1)
MDS	34.5 (4.4)	34.8 (1.5)	34.2 (1.2)	34.5 (1.8)
LLE	27.1 (2.9)	24.9 (2.9)	24.5 (1.8)	28.1 (2.6)
1-SLLE	25.1 (2.4)	25.1 (2.4)	31.6 (4.6)	35.1 (0.0)
α-SLLE	25.5 (2.9)	24.9 (2.9)	24.5 (1.8)	27.5 (3.0)
glass				
Original	57.0 (6.8)	**28.4** (4.5)	36.0 (5.7)	83.5 (7.2)
PCA	51.2 (5.3)	**22.8** (6.7)	45.8 (11.0)	47.0 (4.5)
LDA	40.7 (9.8)	38.1 (7.7)	37.0 (10.0)	47.2 (5.9)
MDS	50.7 (5.1)	**24.4** (8.3)	44.4 (8.9)	47.2 (6.8)
LLE	61.2 (5.7)	33.0 (7.0)	44.0 (8.1)	50.2 (6.5)
1-SLLE	29.5 (4.8)	30.0 (4.7)	48.1 (6.2)	59.8 (5.3)
α-SLLE	36.5 (6.5)	33.0 (7.0)	39.5 (7.4)	41.2 (6.1)

Table 18 Cont.

	NMC	KNNC	LDC	QDC
vehicle				
Original	61.7 (1.7)	36.9 (2.8)	22.0 (3.9)	**14.4** (2.1)
PCA	60.1 (1.6)	46.4 (2.3)	63.5 (3.9)	57.2 (2.6)
LDA	20.6 (2.4)	20.9 (2.2)	20.7 (2.6)	**19.8** (1.9)
MDS	60.6 (2.1)	45.7 (2.0)	62.6 (2.9)	57.5 (4.0)
LLE	50.9 (3.4)	44.9 (3.7)	50.8 (3.4)	47.7 (2.9)
1-SLLE	26.6 (4.6)	26.6 (4.6)	59.5 (1.3)	59.5 (1.3)
α-SLLE	23.5 (3.3)	22.0 (3.2)	24.6 (3.7)	47.7 (2.9)
hepatitis				
Original	**29.4** (9.8)	39.4 (10.6)	**22.5** (13.9)	**32.5** (17.1)
PCA	39.4 (10.6)	48.1 (7.2)	46.2 (6.0)	46.2 (6.0)
LDA	**29.4** (11.0)	**30.0** (14.7)	**29.4** (11.4)	**29.4** (11.4)
MDS	38.8 (10.5)	46.9 (6.8)	45.6 (7.2)	45.6 (6.6)
LLE	**25.0** (10.6)	34.4 (13.3)	31.2 (5.1)	30.0 (7.1)
1-SLLE	**29.4** (7.2)	**29.4** (7.2)	**33.8** (9.4)	36.2 (9.2)
α-SLLE	**25.0** (10.6)	**24.4** (10.0)	**25.0** (13.5)	**25.6** (13.3)
chromosomes				
Original	33.2 (2.0)	23.6 (1.7)	25.1 (2.5)	27.1 (1.4)
PCA	33.0 (1.8)	23.3 (1.5)	24.1 (1.0)	21.4 (1.3)
LDA	24.9 (1.4)	24.5 (1.9)	24.9 (1.4)	21.7 (1.9)
MDS	33.1 (1.7)	23.1 (1.2)	23.8 (1.0)	**19.3** (1.7)
LLE	31.4 (1.7)	25.2 (2.0)	28.4 (1.0)	22.2 (2.2)
1-SLLE	37.4 (1.5)	37.4 (1.5)	93.5 (0.5)	94.1 (1.4)
α-SLLE	28.2 (1.7)	24.3 (1.7)	27.6 (1.8)	22.2 (2.2)

Table 19. Test error (in %), high-dimensional data sets.

	NMC	KNNC	LDC	QDC
ionosphere				
Original	29.9 (5.5)	16.3 (3.1)	16.4 (7.2)	11.4 (3.8)
PCA	28.9 (7.6)	20.9 (3.4)	44.3 (5.9)	38.0 (4.8)
LDA	12.1 (2.7)	13.0 (2.7)	13.9 (2.9)	12.6 (2.7)
MDS	28.4 (7.6)	25.0 (5.1)	34.9 (6.9)	35.4 (7.1)
LLE	21.6 (3.8)	13.0 (2.2)	19.4 (4.5)	19.0 (5.5)
1-SLLE	**7.7** (3.1)	**7.7** (3.1)	31.9 (5.7)	26.9 (2.4)
α-SLLE	**7.0** (2.5)	**7.4** (1.8)	**7.0** (2.5)	**7.3** (2.1)
splice				
Original	23.9 (2.0)	20.5 (1.4)	19.0 (1.4)	**7.0** (1.2)
PCA	44.1 (1.1)	32.8 (1.3)	33.2 (0.8)	32.1 (1.2)
LDA	21.2 (1.4)	18.9 (1.6)	19.3 (1.7)	19.0 (1.4)
MDS	37.3 (3.7)	32.8 (3.8)	30.9 (1.3)	30.5 (1.7)
LLE	36.2 (2.0)	32.8 (2.0)	35.6 (1.4)	42.5 (1.8)
1-SLLE	19.5 (2.3)	19.5 (2.3)	75.9 (0.1)	75.9 (0.1)
α-SLLE	17.2 (2.2)	18.6 (2.1)	17.6 (1.5)	27.1 (1.9)
sonar				
Original	32.4 (7.0)	18.5 (5.3)	25.4 (5.2)	27.8 (5.9)
PCA	46.8 (5.6)	51.7 (3.9)	44.6 (6.7)	43.4 (5.6)
LDA	25.4 (10.0)	24.6 (10.1)	25.6 (10.6)	25.4 (10.9)
MDS	46.8 (6.3)	49.3 (5.1)	45.4 (4.0)	44.9 (4.9)
LLE	23.4 (6.1)	18.8 (7.4)	22.0 (5.7)	26.1 (5.4)
1-SLLE	**11.7** (3.0)	**11.7** (3.0)	53.7 (0.0)	53.7 (0.0)
α-SLLE	**13.7** (4.5)	**12.9** (2.3)	**14.4** (4.8)	**14.6** (3.3)
optdigits				
Original	8.6 (0.7)	**1.2** (0.4)	55.8 (44.3)	90.1 (0.0)
PCA	10.1 (0.8)	2.8 (0.5)	8.4 (1.0)	4.2 (0.5)
LDA	4.7 (0.7)	2.9 (0.3)	4.7 (0.7)	3.1 (0.5)
MDS	10.1 (0.6)	3.0 (0.4)	8.8 (0.6)	4.2 (0.4)
LLE	15.6 (1.5)	3.2 (0.5)	10.4 (1.2)	5.1 (1.0)
1-SLLE	**1.4** (0.4)	**1.4** (0.4)	31.0 (1.0)	31.0 (1.0)
α-SLLE	**1.4** (0.4)	**1.3** (0.2)	2.0 (0.4)	3.4 (0.6)

Table 19 Cont.

	NMC	KNNC	LDC	QDC
natural textures				
Original	54.5 (2.1)	34.2 (1.0)	54.9 (1.5)	46.0 (1.8)
PCA	54.5 (1.2)	40.9 (2.7)	53.9 (1.4)	41.2 (2.2)
LDA	55.2 (1.7)	53.6 (1.4)	55.2 (1.7)	52.4 (2.1)
MDS	55.1 (1.7)	43.2 (2.2)	54.9 (1.5)	43.9 (2.0)
LLE	56.0 (2.0)	49.4 (2.4)	55.6 (1.7)	58.3 (3.3)
1-SLLE	**30.6** (2.9)	**30.6** (2.9)	79.7 (0.9)	79.7 (0.9)
α-SLLE	33.7 (2.6)	**30.6** (3.1)	38.1 (2.6)	37.5 (2.3)
structured textures				
Original	51.3 (1.4)	13.7 (1.3)	55.1 (1.7)	24.4 (0.8)
PCA	52.0 (1.5)	30.0 (1.9)	52.0 (1.5)	30.2 (1.2)
LDA	55.8 (1.9)	52.9 (1.7)	55.8 (1.9)	50.4 (1.6)
MDS	52.7 (1.7)	37.6 (1.0)	53.6 (1.2)	36.3 (1.2)
LLE	27.9 (2.1)	14.5 (1.4)	27.8 (1.7)	13.4 (1.3)
1-SLLE	9.5 (0.9)	9.5 (0.9)	49.3 (1.9)	49.3 (1.9)
α-SLLE	8.2 (0.8)	**7.5** (1.0)	8.2 (0.7)	11.7 (1.2)
NIST digits				
Original	16.5 (0.9)	**2.5** (0.2)	10.8 (0.4)	89.8 (0.4)
PCA	21.5 (0.9)	6.7 (0.7)	17.0 (1.0)	9.3 (1.0)
LDA	10.3 (0.9)	7.0 (0.8)	10.3 (0.9)	7.9 (0.8)
MDS	20.1 (0.7)	7.1 (1.0)	17.4 (0.7)	8.8 (0.9)
LLE	17.1 (0.7)	6.7 (0.5)	16.1 (0.8)	10.7 (0.6)
1-SLLE	3.4 (0.5)	3.4 (0.5)	40.0 (1.1)	40.0 (1.1)
α-SLLE	**2.3** (0.3)	**2.5** (0.4)	3.2 (0.7)	10.7 (0.6)
paper				
Original	3.3 (1.2)	**0.2** (0.3)	75.1 (0.0)	75.1 (0.0)
PCA	2.8 (1.2)	**0.1** (0.2)	1.2 (0.9)	0.5 (0.5)
LDA	26.7 (4.1)	26.7 (4.1)	75.1 (0.0)	75.1 (0.0)
MDS	1.4 (0.9)	**0.1** (0.2)	**0.1** (0.2)	**0.1** (0.2)
LLE	0.3 (0.3)	**0.2** (0.3)	**0.2** (0.3)	21.3 (29.4)
1-SLLE	**0.1** (0.2)	**0.1** (0.2)	2.9 (0.8)	2.9 (0.8)
α-SLLE	**0.2** (0.3)	**0.1** (0.2)	**0.2** (0.3)	21.3 (29.4)

The results confirm that SLLE generally leads to better classification performance than LLE and, usually, any other mapping technique. This is to be expected, as SLLE can extract nonlinear manifolds in a supervised way, and is thereby the most general of the feature extraction methods. Besides this, there are a number of interesting observations:

– SLLE does not work on low-dimensional data. For the low-dimensional sets shown in Table 18, the Bayes plug-in classifiers, LDC and QDC, often work well on the original data, as the number of parameters that need to be estimated is still reasonably low. In these cases, SLLE will not improve classification to the point where it is better than on the original data.

- SLLE works well on high-dimensional data. In some cases, performance is (nearly) as good as on the original data, but in others it is significantly better (ionosphere, sonar, the textures sets). The splice set is the exception: the quadratic classifier QDC performs surprisingly well and cannot be improved upon.
- SLLE works well when KNN works well on the original data. This can be explained by the fact that in order to obtain projections of the test points, they are mapped as follows: first, the K nearest neighbours are found in the high-dimensional space, and then, they are mapped using LG2.
- 1-SLLE vs. α-SLLE: neither consistently outperforms the other. Although α-SLLE was expected to generalize better, there are two extra parameters to be estimated: α and the embedding dimensionality, d. The performance of α-SLLE is especially sensitive to the latter. Bayes plug-in classifiers trained on 1-SLLE mapped data perform poorly: as all samples of a particular class are mapped onto a single point, there is no covariance structure left.
- The nearest mean classifier performs well. SLLE maps the data nonlinearly such that this simple classifier can do well. This is analogous to support vector machines (SVMs): after the kernel function has performed the desired nonlinear mapping, a simple linear classifier suffices.

6.3 Summary

The supervised variant of LLE, called SLLE, projects data placed on multiple manifolds to a single coordinate system. The objective is changed from representing the manifolds as well as possible to retaining class separability as well as possible. This is achieved by distorting the distance matrix on which LLE is calculated, such that distances between samples belonging to different classes are enlarged. If this enlargement is such that the K nearest neighbors of a sample will always be found in the same class (1-SLLE), the resulting mapping will collapse each class into an area close to a single point (i.e., for n-class problem, embedding in a $(n-1)$-dimensional space is sufficient). When the enlargement of the distances is less, a trade-off between embedding and class separation is achieved (α-SLLE). This may lead to better generalization, at the cost of selecting an enlargement factor α and an embedding dimensionality d (De Ridder *et al.* 2003a,b).

The extensive experiments show that simple classifiers, specifically the nearest mean classifier, trained on SLLE mapped data sets can outperform other classifiers (KNN, LDC, QDC) on the original data or data mapped by PCA, LDA, MDS or LLE. This is particularly true for high-dimensional data sets with low intrinsic dimensionalities, such as image data. In summarizing, SLLE in combination with an appropriate classifier can be quite successful in classifying high-dimensional data.

In this chapter, we have compared the classification performance of SLLE and LLE with other feature extraction techniques (PCA, LDA and MDS). But there remains an open question: which will be the winner if SLLE is compared with a simple but efficient feature selection algorithm, for example, forward selection? We leave this issue for future research.

7 Discussion

The locally linear embedding (LLE) algorithm is utilized for nonlinear dimensionality reduction of high-dimensional data. It chops a curved surface, such as that of a swiss-roll, into small patches, each of which does not involve a lot of curve so that each patch can be considered to be almost flat. These patches are then flattened and stitched together in a low-dimensional space in such a way that the resulting projection can yield information about the global geometry of the data (Martinetz & Schulten 1994, Tenenbaum 1998).

The idea used by LLE for dealing with nonlinearity by means of dividing the whole manifold into local patches that can be approximated by hyperplanes is not new (Bregler & Omohundro 1995, Ghahramani & Hinton 1996, Hinton *et al.* 1997, Kambhatla & Leen 1997, Saul & Rahim 1999). The known models, however, possess a common disadvantage: their objective functions (measuring either least square reconstruction error or log likelihood) are invariant to arbitrary rotations and reflections of the local coordinate systems in each linear model. Thus, these learning algorithms lead to inconsistent alignment of the local linear models, instead they yield internal representations that change unpredictably as one traverses connected paths on the manifold (Saul & Roweis 2003).

In contrast, LLE overcomes this problem and produces a single global coordinate system of lower dimensionality. This low-dimensional representation describes a true structure of the data due to the properties of the reconstruction weights capturing and preserving information about the local neighborhoods of the data in the second step of LLE - they are invariant to translation, rotation, and scaling. In spite of this, LLE shares a common shortcoming with mixture models: how does one define an optimal partition of the data manifold into local patches? In this thesis, this problem is solved by a hierarchical method for automatic selection of an optimal number of nearest neighbors (Section 3.2), which is less time consuming in comparison to the straightforward approach.

Another free parameter of the LLE algorithm is the number of dimensions to which high-dimensional data should be reduced. It is clear that this parameter is set to be equal to two or three in the case of visualization, since human beings can perceive at most 3D space. In data analysis, dimensionality reduction is usually used as an intermediate step before such kinds of operations as, for example, clustering, classification etc. Here, the dimensionality of the projected data is not always equal to two or three. Therefore, one needs to approximately estimate the intrinsic dimensionality of the data in order to preserve the most important information, and reduce the influence of noise and outliers in the

following steps of the data analysis. In Polito & Perona (2002) the authors try to apply LLE to estimate the upper bound of the data intrinsic dimensionality in a similar way to the PCA approach[1] - that is, to estimate it by the number of eigenvalues that are appreciable in magnitude to the smallest non-zero eigenvalue of the cost matrix calculated in the third step of LLE. But, as was empirically shown in Saul & Roweis (2003), this procedure sometimes fails to define the exact intrinsic dimensionality of non-uniform data. Therefore, a classical method should be used for this purpose (Pettis *et al.* 1979, Brand 2003, Kégl 2003). Nevertheless, many classical techniques tend to underestimate the intrinsic dimensionality of data, while the LLE-based intrinsic dimensionality estimator slightly overestimates it. Thus, from the point of view of preservation of useful information, it is safer to use the latter approach for calculating an $approximate$ dimensionality of the data manifold (Section 3.2).

A weak point of LLE is that it performs dimensionality reduction in a batch or offline mode. In other words, it obtains a low-dimensional representation for the fixed amount of high-dimensional data points to which the algorithm is applied, and when new data points arrive, one has to rerun LLE on pooled (both old and new) data in order to obtain new embeddings. This question has arisen in previous studies (Zheng 2000, Kouropteva 2001, Saul & Roweis 2003, Bengio *et al.* 2004), where authors suggested several different approaches to make LLE incremental. Some of them are time consuming, since they need to solve new high-dimensional eigen-problems; others use an explicit mapping between high-dimensional and low-dimensional data representation, which is based on the assumption that the data manifold is perfectly sampled and does not contain noisy factors. Unfortunately, the latter assumption is not always met in real-world high-dimensional data sets; therefore, application of these methods for generalization to new data points either becomes a time consuming procedure or involves distortions in the final mapping. In order to overcome these shortcomings, the incremental version of LLE, which is based on the intrinsic properties of LLE, has been proposed (Section 3.2). The experimental results demonstrate that this is a powerful generalization method for data whose distribution is not uniform and the local linearity constraints do not hold.

To provide fair comparison, LLE was tested on a number of similar methods such as Isomap and SOM. Isomap is a nonlinear dimensionality reduction algorithm that has been proposed by Tenenbaum (Tenenbaum 1998, Tenenbaum *et al.* 2000). One can think of Isomap as a nonlinear generalization of MDS in which embeddings are optimized to preserve geodesic distances between pairs of data points. These distances are calculated by computing the shortest paths through the graph built on the data points (each point corresponds to the vertex of the graph). Though Isomap aims to decrease data dimensionality in a nonlinear way, it radically differs from LLE. In particular, Isomap attempts to preserve the global geometry of the data manifold, as characterized by the geodesic distances between points, while LLE attempts to preserve the local geometry which is described by the linear coefficients of local reconstruction. Thus, the most common failure mode of LLE (if the data is noisy, sparse, weakly connected or the parameter indicating the number of nearest neighbors is set to be either too low or too high) is to map faraway input points to nearby points in the embedding space. By contrast, Isomap embedding can possess distortions in the local geometry (Saul & Roweis 2003). However, the exper-

[1] The intrinsic dimensionality of the data corresponds to the number of largest singular values of the covariance matrix of the data.

iments of Chapter 4 show that Isomap preserves local, as well as global structure better than LLE.

Another nonlinear dimensionality reduction method called SOM also discovers a low-dimensional data representation like LLE, but uses a completely different approach (Chapter 4). SOM learns nonlinear manifolds by a process involving a stochastic computation and the accuracy of the map depends on the number of iterations, whereas LLE performs manifold learning in a non-stochastic way and needs only a single run through three steps. In spite of the fact that these methods are totally different, they possess a common characteristic: their results depend on how the parameters are chosen. In the case of SOM, one has to tune two parameters - the learning rate and the neighborhood; the dimensionality of the low-dimensional space is usually set to two, but higher dimensional representation is also possible (Kohonen 1997). Unfortunately, there is no theoretical basis for the selection of these parameters and they are usually empirically determined. A number of details about the selection of these parameters, variants of the map, and many other aspects have been covered in (Kohonen 1995). In its turn, LLE parameters (number of nearest neighbors and the dimensionality of the projected space) can be found using the proposed extensions (Section 3.2).

From a practical applications point of view, SOM is widely used for data visualization (Kaski *et al.* 1998, Oja *et al.* 2003, Turtinen *et al.* 2003, Silven *et al.* 2003). Its resulting two-dimensional grid is easy to interpret and understand for a user, whereas LLE visualization, when LLE merely projects data to two or three may not be fully descriptive (Saul & Roweis 2003), especially if the intrinsic dimensionality of the data is higher than two or three. Therefore, a new visualization framework is developed, where LLE is used as an intermediate step (Chapter 5). The novelties of the framework are that it obtains a visualization result in a semi-supervised way and uses a two-level dimensionality reduction scheme. Hence, it eliminates an enormous loss of information content of high-dimensional data while obtaining its one, two or three-dimensional representation.

Whatever the application, certain limitations of the LLE algorithm should be kept in mind (Saul & Roweis 2003). First of all, the original LLE cannot recover data containing several disjointed manifolds or manifolds without boundaries (sphere or torus). The former kind of data can be processed with a parameterized supervised variant of LLE, described in Chapter 6, while the second one remains an open problem. The supervised LLE extends the concept of LLE to multiple manifolds, but changes the objective from representing the manifolds as well as possible to retaining class separability as well as possible. Therefore, it is worth reducing the dimensionality of high-dimensional data with SLLE and then applying a classifier to the projected data. The empirical study shows that even a simple classifier, specifically the nearest mean classifier, trained on SLLE mapped data can outperform other classifiers (K-nearest neighbor, linear and quadratic Bayes plug-in classifiers) on the original data or data mapped by PCA, LDA, MDS or LLE. This is particularly true for high-dimensional data with low intrinsic dimensionality.

In Hadid & Pietikäinen (2003) an attempt to represent disjoint manifolds in the same low-dimensional space was made. The authors proposed first to map the largest connected component of the high-dimensional data into low-dimensional space and then to map the rest of the data by utilizing linear generalization (Saul & Roweis 2003). This approach works if the data is composed of similar objects, for example, faces of one person. However, it may be incorrect to use it if faces of many people organize a data set, since the

main component might be composed of samples representing a person A; hence, the projections of faces belonging to other persons (B, C etc) are obtained as reconstructions of faces of the person A. It is clear that in this case, a tractable projection cannot be obtained, and it would be better to map each of the distinct components separately.

Another limitation of LLE concerns data sets whose intrinsic dimensionality is not the same in all parts of the data space, or whose structure is better described as fractal. How should one deal with this problem? Further studies are needed. The related open question - how to deal with poorly or non-uniformly sampled data - should also be investigated. One of the ideas is to apply SLLE to this sort of data but with the enlargement constant α changing depending on how dense a neighborhood of particular point is. Further, it would be very interesting to find a method for automatic determination of this constant. Additional research can be done on the incremental version of LLE. One can think of possibility of applying ILLE not only to interpolate, but also to extrapolate previously unseen data points to the already mapped data.

8 Conclusions

Dimensionality reduction is a very important operation in data mining. It aims at obtaining a low-dimensional representation of high-dimensional data, while preserving the most important characteristics of the data. Usually, dimensionality reduction is used as a preprocessing step before further data analysis (clustering, classification, visualization etc.), since it helps to eliminate unimportant and noisy factors, and to avoid the effect called the curse of dimensionality. Dimensionality reduction can be done either by feature extraction or by feature selection. A disadvantage of feature extraction is that the new features lack the physical meaning of the original features, since the new features are obtained by applying certain operations to the original ones. In spite of this fact, there is an advantage of feature extraction: given the same number of reduced features, the transformed features can provide better results in further data analysis than the selected individual features (Hall 1998).

In this thesis, a recently proposed unsupervised feature extraction algorithm called locally linear embedding was considered. It belongs to the group of nonlinear manifold learning algorithms, which reduce dimensionality with a learning structure of the manifold formed by data points. The attractive properties of LLE are: 1) only two parameters are to be set, 2) optimizations do not involve local minima, 3) local geometry of high dimensional data is preserved in the embedded space, 4) there is a single global coordinate system of the embedded space. Its ability to deal with large amounts of high dimensional data and its non-iterative way of finding the embeddings make it more and more attractive to researchers.

The purpose of this thesis was to explore and extend LLE beyond already known findings and to apply it to some data mining tasks (clusterization, visualization and classification):

- Automatic determination of the parameters of LLE was suggested, leading to good results described in Chapter 3.
- In the same chapter a new incremental version of LLE was described. It allows us to overcome the main deficiency of LLE, i.e., the need to completely re-run the algorithm when new inputs arrive.
- The clustering property of LLE was studied and compared with other nonlinear dimensionality reduction techniques (Isomap, SOM, S-LogMap) in Chapter 4. Here,

we do not use any clustering technique, we just measure the cluster organization of the known data classes.

- Visualization with LLE was suggested in Chapter 5, where a new framework was described. Its main novelties are its semi-supervised approach and two-level dimensionality reduction scheme. These innovations allow the obtaining of better visualization results than ordinary dimensionality reduction techniques, which merely project data to two-dimensional space.
- The supervised LLE described in Chapter 6 serves for preprocessing data before applying a classifier. It also can deal with data lying on disjoint manifolds.

References

Ahlberg C & Shneiderman B (1994) Visual information seeking: tight coupling of dynamic query filters with starfield displays. Proc. of Human Factors in Computing Systems, 313–317.

Aldridge C (1998) A theory of empirical spatial knowledge supporting rough set based knowledge discovery in geographic databases. Ph.D. thesis, University of Otago, Dunedin, New Zealand.

Andrews D (1972) Plots of high dimensional data. Biometrics 28: 125–136.

Bai Z, Demmel J, Dongorra J, Ruhe A & van der Vorst H (2000) Templates for the Solution of Algebraic Eigenvalue Problems. SIAM, Philadelphia.

Baim P (1988) A method for attribute selection in inductive learning systems. IEEE Trans. on Pattern Analysis and Machine Intelligence 106: 888–896.

Barnett V & Lewis T (1994) Outliers in Statistical Data: Third edition. John Wiley, New York.

Belhumeur P, Hespanha J & Kriegman D (1997) Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. IEEE Trans. on Pattern Analysis and Machine Intelligence 197: 711–120.

Belkin M & Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 156: 1,373–1,396.

Bell A & Sejnowski T (1995) An information-maximization approach to blind separation. Neural Computation 7: 1,004–1,034.

Bellman R (1961) Adaptive Control Processes: A Guided Tour. Princeton, NJ: Princeton University Press.

Bengio Y, Paiement JF, Vincent P, Delalleau O, Le Roux N & Ouimet M (2004) 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, volume 16. Cambridge, MA, MIT Press.

Bentley C & Ward M (1996) Animating multidimensional scaling to visualize N-dimensional data sets. Proc. of IEEE Information Visualization, 72–73.

Berry M (1992) Large-scale sparse singular value computations. International Journal of Super-Computer Applications 61: 13–49.

Berry M, Dumais S & Obrien G (1995) Using linear algebra for intelligent information-retrieval. Siam Review 37: 573–595.

Bingham E & Mannila H (2001) Random projection in dimensionality reduction: applications to image and text data. Proc. of the Seventh International Conference on Knowledge Discovery and Data Mining, 245–250.

Bishop C & Tipping M (1998) A hierarchical latent variable model for data visualization. IEEE Trans. on Pattern Analysis and Machine Intelligence 203: 281–293.

Boscolo R, Pan H & Roychowdhury V (2004) Independent component analysis based on nonparametric density estimation. IEEE Trans. on Neural Networks 151: 55–65.

Brand M (2003) Charting a manifold. In: Becker S, Thrun S & Obermayer K (eds) Advances in Neural Information Processing Systems, volume 15, 961–968. Cambridge, MA, MIT Press.

Bregler C & Omohundro S (1995) Nonlinear image interpolation using manifold learning. In: Tesauro G, Touretzky D & Leen T (eds) Advances in Neural Information Processing Systems, volume 7. Cambridge, MA, MIT Press.

Brill F, Brown D & Martin W (1992) Fast genetic selection of features for neural network classifiers. IEEE Trans. on Neural Networks 32: 324–328.

Brodatz P (1966) Textures, a photographic album for artists and designers. Dover Publications Inc., New York, NY. http://www.ux.his.no/∼tranden/brodatz.html.

Brun A (2006) Manifold learning and representations for image analysis and visualization. Department of Biomedical Engineering, University of Linköpings. Licentiate thesis.

Brun A, Westin CF, Haker S & Knutsson H (2005) Fast manifold learning based on Riemannian normal coordinates. In: Kälviäinen H, Parkkinen J & Kaarna A (eds) Proc. of Fourteenth Scandinavian Conference on Image Analysis, Lecture Notes in Computer Science, volume 3540, 920–929. Springer.

Bruske J & Sommer G (1998) Intrinsic dimensionality estimation with optimally topology preserving maps. IEEE Trans. on Pattern Analysis and Machine Intelligence 205: 572–575.

Chernoff H (1973) The use of faces to represent points in k-dimensional space graphically. Journal of the American Statistical Association 68: 361–368.

Comon P (1994) Independent component analysis, a new concept? Signal Processing 363: 287–314.

Cover T & Thomas J (1991) Elements of Information Theory. John Wiley and Sons.

Cover T & Van Campenhout J (1977) On the possible orderings in the measurement selection problem. IEEE Trans. on Systems, Man, and Cybernetics 79: 657–661.

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

Datar M, Immorlica N, Indyk P & Mirrokni V (2004) Locality-sensitive hashing scheme based on p-stable distributions. Proc. of the Twentieth Annual Symposium on Computational Geometry, 253–262.

Dayan P (1999) Unsupervised learning. In: Wilson R & Keil F (eds) The MIT Encyclopedia of the Cognitive Sciences. The MIT Press.

De Ridder D (2001) Adaptive methods of image processing. Ph.D. thesis, Delft University of Technology, Delft.

De Ridder D & Duin R (1997) Sammon's mapping using neural networks: comparison. Pattern Recognition Letters 1811-13: 1,307–1,316.

De Ridder D & Duin R (2002) Locally linear embedding for classification. Technical Report PH-2002-01, Pattern Recognition Group, Dept. of Imaging Science and Technology, Delft University of Technology.

De Ridder D, Kouropteva O, Okun O, Pietikäinen M & Duin R (2003a) Supervised locally linear embedding. In: Kaynak O, Alpaydin E, Oja E & Xu L (eds) Proc. of the Thirteenth International Conference on Artificial Neural Networks, Lecture Notes in Computer Science, volume 2714, 333–341. Springer.

De Ridder D, Kouropteva O, Okun O, Pietikäinen M & Duin R (2003b) Supervised locally linear embedding. Proc. of the Fifteenth Belgium-Netherlands Conference on Artificial Intelligence, 429–430.

DeCoste D (2001) Visualizing Mercer kernel feature spaces via kernelized locally-linear embeddings. Proc. of the Eighth International Conference on Neural Information Processing.

Donoho D & Grimes G (2003) Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. Proc. of National Academy of Sciences 10010: 5,591–5,596.

Duda R, Hart P & Stork D (2001) Pattern Classification: Second edition. John Wiley & Sons, Inc., New York.

Duin R (2004) PRTools, a pattern recognition toolbox for Matlab. http://www.prtools.org/.

Esposito R (2003) Analyzing ensemble learning in the framework of Monte Carlo theory. Ph.D. thesis, Dipartimento di Informatica, Universitá di Torino, C.so Svizzera 185, 10149 Torino, Italy.

Fletcher P, Joshi S, Lu C & Pizer S (2003) Gaussian distributions on Lie groups and their application to statistical shape analysis. Proc. of Information Processing in Medical Imaging, 450–462.

Fletcher P, Lu C, Pizer S & Joshi S (2004) Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Trans. on Medical Imaging 238: 995–1,005.

Fokkema D, Sleijpen G & van der Vorst H (1998) Jacobi-davidson style QR and QZ algorithms for the reduction of matrix pencils. SIAM Journal on Scientific Computing 201: 94–125.

Foster I (1995) Designing and Building Parallel Programs. Addison-Wesley.

Friedman J (1987) Exploratory projection pursuit. Journal of the American Statistical Association 82397: 249–266.

Friedman J (1995) An overview of prediction learning and function approximation. In: Cherkassky V, Friedman J & Wechsler H (eds) From statistics to neural networks: theory and pattern recognition applications. New York: Springer-Verlag.

Friedman J, Bentley J & Finkel R (1977) An algorithm for finding best matches in logarithmic expected time. ACM Trans. on Mathematical Software 33: 209–226.

Friedman J & Tukey J (1974) A projection pursuit algorithm for exploratory data analysis. IEEE Trans. on Computers 239: 881–890.

Ghahramani Z & Hinton GE (1996) The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto.

Goldberg D (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley.

Golub G & Van Loan C (1989) Matrix Computations: Second edition. Johns Hopkins University Press, Baltimore.

Gower J (1968) Adding a point to vector diagrams in multivariate analysis. Biometrika 553: 582–585.

Grinstein G & Ward M (2002) Introduction to data visualization. In: Fayyad U, Grinstein G & Wierse A (eds) Information Visualization in Data Mining and Knowledge Discovery, 21–45. Mogran Kaufmann Publishers.

Hadid A & Pietikäinen M (2003) Efficient locally linear embeddings of imperfect manifolds. In: Perner P & Rosenfeld A (eds) Proc. of Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science, volume 2734, 188–201. Springer.

Hall M (1998) Correlation-based feature selection machine learning. Ph.D. thesis, University of Waikato, New Zealand.

Hall P, Marshall A & Martin R (2000) Merging and splitting eigenspace models. IEEE Transactions on Pattern Analysis and Machine 229: 1,042–1,049.

Han J & Kamber M (2001) Data Mining, Concepts and Technique. Morgan Kaufmann, San Francisco.

Hartigan J (1975) Clustering Algorithms. Wiley.

Haykin S (1999) Neural Networks: A Comprehensive Foundation. Prentice Hall International, Inc.

He X & Niyogi P (2004) Locality preserving projections. In: Thrun S, Saul L & Schölkopf B (eds) Advances in Neural Information Processing Systems 16. MIT Press,Cambridge, MA.

He X, Yan S, Hu Y, Niyogi P & Zhang H (2005) Face recognition using Laplacianfaces. IEEE Trans. on Pattern Analysis and Machine Intelligence 273: 328–340.

Hettich S, Blake C & Merz C (1998) UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Hinton GE, Dayan P & Revow M (1997) Modeling the manifolds of images of handwritten digits. IEEE Trans. on Neural Networks 8: 65–74.

Holland J (1992) Adaptation in Nature and Artificial Systems. MIT Press.

Hotelling H (1933) Analysis of a complex of statistical variables into principal components. Journal of Education Psychology 24: 417–441.

Hyvärinen A & Oja E (2000) Independent component analysis: algorithms and applications. Neural Networks 134-5: 411–430.

Inselberg A & Dimsdale B (1990) Parallel coordinates: a tool for visualizing multi-dimensional geometry. Proc. of the First IEEE Conference on Visualization, 361–378.

Jain A & Chandrasekaran B (1982) Dimensionality and sample size considerations in pattern recognition practice. In: Krishnaiah P & Kanal L (eds) Handbook of statistics, volume 2, 835–855. Amsterdam: North-Holland.

Jain A & Dubes R (1988) Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, NJ.

Jain A, Duin R & Mao J (2000) Statistical pattern recognition: a review. IEEE Trans. on Pattern

Analysis and Machine Intelligence 221: 4–37.
Jain A & Zongker D (1997) Feature selection: evaluation, application, and small sample performance. IEEE Trans. on Pattern Analysis and Machine Intelligence 192: 153–158.
Jolliffe I (1989) Principal Component Analysis. Springer-Verlag, New York.
Kambhatla N & Leen T (1997) Dimension reduction by local principal component analysis. Neural Computation 9: 1,493–1,516.
Karcher H (1977) Riemannian center of mass and millifier smoothing. Communications on Pure and Applied Mathematics 305: 509–541.
Karger D & Ruhl M (2002) Finding nearest neighbors in growth-restricted metrics. Proc. of the Thirty Fourth ACM Symposium on the Theory of Computing, 741–750.
Kaski S, Kangas J & Kohonen T (1998) Bibliography of self-organizing map (SOM) papers: 1981–1997. Neural Computing Surveys 1: 102–350.
Kaski S & Sinkkonen J (2000) Metrics that learn relevance. Proc. of the IEEE International Joint Conference on Neural Networks, 547–552.
Kégl B (2003) Intrinsic dimension estimation using packing numbers. In: Becker S, Thrun S & Obermayer K (eds) Advances in Neural Information Processing Systems, volume 15, 681–688. Cambridge, MA, MIT Press.
Kim T & Kittler J (2005) Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image. IEEE Trans. on Pattern Analysis and Machine Intelligence 273: 318–327.
Kohonen T (1990) Improved versions of learning vector quantization. IEEE International Joint Conference on Neural Networks 1: 545–550.
Kohonen T (1995) Self-Organizing Maps. Springer, Berlin, Heidelberg.
Kohonen T (1997) Self-Organizing Maps: Second edition. Springer-Verlag, Berlin.
Kouropteva O (2001) Unsupervised learning with locally linear embedding algorithm: an experimental study. Master's thesis, University of Joensuu, Finland.
Kouropteva O, Okun O, Hadid A, Soriano M, Marcos S & Pietikäinen M (2002a) Beyond locally linear embedding algorithm. Technical Report MVG-01-2002, University of Oulu.
Kouropteva O, Okun O & Pietikäinen M (2002b) Selection of the optimal parameter value for the locally linear embedding algorithm. Proc. of 2002 International Conference on Fuzzy Systems and Knowledge Discovery, 359–363.
Kouropteva O, Okun O & Pietikäinen M (2003a) Classification of handwritten digits using supervised locally linear embedding and support vector machine. Proc. of the Eleventh European Symposium on Artificial Neural Networks, 229–234.
Kouropteva O, Okun O & Pietikäinen M (2003b) Supervised locally linear embedding algorithm for pattern recognition. In: Campilho A, Pérez N & Sanfeliu A (eds) Proc. of Pattern Recognition and Image Analysis, Lecture Notes in Computer Science, volume 2652, 386–394. Springer.
Kouropteva O, Okun O & Pietikäinen M (2004a) Data visualization with multiple machine learning methods. Proc. of the Fourth IASTED International Conference on Visualization, Imaging, and Image Processing, 190–196.
Kouropteva O, Okun O & Pietikäinen M (2004b) Semi-supervised visualization of high-dimensional data. Proc. of the Seventh International Conference on Pattern Recognition and Image Analysis: New Information Technologies, 748–751.
Kouropteva O, Okun O & Pietikäinen M (2005a) Incremental locally linear embedding. Pattern Recognition 3810: 1,764–1,767.
Kouropteva O, Okun O & Pietikäinen M (2005b) Incremental locally linear embedding algorithm. In: Kälviäinen H, Parkkinen J & Kaarna A (eds) Proc. of fourteenth Scandinavian Conference on Image Analysis, Lecture Notes in Computer Science, volume 3540, 521–530. Springer.
Kouropteva O, Okun O & Pietikäinen M (2005c) Semi-supervised visualization of high-dimensional data. Pattern Recognition and Image Analysis 153: 1–4.
Kruskal J & Wish M (1978) Multidimensional Scaling. SAGE publications, Beverly Hills.
LeCun Y (1998) The MNIST database of handwritten digits. Courant Institute, Nyu. http://yann.lecun.com/exdb/mnist/index.html.
LeCun Y, Bottou L, Orr G & Muller K (1998) Efficient backprop. In: Orr G & Muller K (eds) Neural Networks: Tricks of the trade. Springer.

Lee J, Lendasse A, Donckers N & Verleysen M (2000) A robust nonlinear projection method. In: Verleysen M (ed) Proc. Of European Symposium on Artificial Neural Networks, 13–20. D-Facto Publications, Bruges, Belgium.

Liu H, Li J & Wong L (2002) A comparative study on feature selection and classification methods using gene and protein expression profiles. In: Proceedings of Thirteenth International Conference on Genome Informatics, 51–60. Universal Academy Press, Tokyo, Japan.

Lowe D & Webb A (1991) Optimized feature extraction and the Bayes decision in feed-forward classifier networks. IEEE Trans. on Pattern Analysis and Machine Intelligence 134: 355–364.

Mao J & Jain A (1995) Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. on Neural Networks 62: 296–317.

Mao J, Mohiuddin K & Jain A (1994) Parsimonious network design and feature selection through node pruning. Proc. of the Twelfth International Conference on Pattern Recognition, 622–624.

Martinetz T & Schulten K (1994) Topology representing networks. Neural Networks 73: 507–522.

Martinez A & Kak A (2001) PCA versus LDA. IEEE Trans. on Pattern Analysis and Machine Intelligence 232: 228–233.

Moddemeijer R (1989) On estimation of entropy and mutual information of continuous distributions. Signal Processing 16: 233–248.

Morrison A, Ross G & Chalmers M (2003) Fast multidimensional scaling through sampling, springs and interpolation. Visualization 2: 68–77.

Narendra P & Fukunaga K (1977) A branch and bound algorithm for feature subset selection. IEEE Trans. on Computers C-269: 917–922.

Nene S, Nayar S & Murase H (1996) Columbia Object Image Library (COIL-20). Columbia University. http://www1.cs.columbia.edu/CAVE/research/softlib/coil-20.html.

Niemann H (1980) Linear and nonlinear mappings of patterns. Pattern Recognition 12: 83–87.

Niskanen M & Silven O (2003) Comparison of dimensionality reduction methods for wood surface inspection. Proc. of the Sixth International Conference on Quality Control by Artificial Vision, 178–188.

Oh I, Lee J & Moon B (2004) Hybrid genetic algorithms for feature selection. IEEE Trans. on Pattern Analysis and Machine Intelligence 2611: 1,424–1,437.

Oja M, Kaski S & Kohonen T (2003) Bibliography of self-organizing map (SOM) papers: 1998–2001. Neural Computing Surveys 3: 1–156.

Ojala T, Pietikäinen M & Mäenpää T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. on Pattern Analysis and Machine Intelligence 247: 971–987.

Okun O, Kouropteva O & Pietikäinen M (2002) Supervised locally linear embedding algorithm. Proc. of the Tenth Finnish Artificial Intelligence Conference, 50–61.

Omohundro S (1989) Five balltree construction algorithms. Technical Report TR-89-063, International Computer Science Institute.

Omohundro S (1991) Bumptrees for efficient function, constraint, and classification learning. In: Lippmann R, Moody J & Touretzky D (eds) Advances in Neural Information Processing, volume 3, 693–699. San Mateo, CA, Morgan Kaufmann.

Park H, Jeon M & Rosen J (2003) Lower dimensional representation of text data based on centroids and least squares. BIT 432: 1–22.

Peltonen J & Kaski S (2005) Discriminative components of data. IEEE Trans. on Neural Networks 161: 68–83.

Peltonen J, Klami A & Kaski S (2004) Improved learning of Riemannian metrics for exploratory analysis. Neural Networks 168-9: 1,087–1,100.

Pettis K, Bailey T, Jain A & Dubes R (1979) An intrinsic dimensionality estimator from near-neighbor information. IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-1: 25–37.

Polito M & Perona P (2002) Grouping and dimensionality reduction by locally linear embedding. In: Dietterich T, Becker S & Ghahramani Z (eds) Advances in Neural Information Processing Systems, volume 14, 1,255–1,262. Cambridge, MA, MIT Press.

Poskanzer J (1991) Wood image database. University of Oulu, Machine Vision Group. ftp://ftp.ee.oulu.fi/pub/tklab/.

Post F, van Walsum T, Post F & Silver D (1995) Iconic techniques for feature visualization. In: Nielson G & Silver D (eds) Visualization'95, 288–295. IEEE Computer Society Press, Los Alamitos, CA.

Press W, Teukolsky S, Vetterling W & Flannery B (1993) Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press.

Pudil P, Novovičová J & Kittler J (1994) Floating search methods in feature selection. Pattern Recognition Letters 15: 1,119–1,125.

Quinlan J (1986) Introduction of decision trees. In: Machine Learning, volume 1, 81–106. Kluwer Academic Publishers, Dordrecht.

Raymer M, Punch W, Goodman E, Kuhn L & Jain A (2000) Dimensionality reduction using genetic algorithms. IEEE Trans. on Evolutionary Computation 42: 164–171.

Ribarsky W, Ayers E, Eble J & Mukherjea S (1994) Glyphmaker: creating customized visualizations of complex data. IEEE Trans. on Computers 277: 57–64.

Roweis S (1997) EM algorithms for PCA and SPCA. In: Jordan M, Kearns M & Solla S (eds) Advances in Neural Information Processing Systems, volume 10, 626–632. Cambridge, MA, MIT Press.

Roweis S & Saul L (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 2905500: 2,323–2,326.

Rumelhart D, Hinton G & Williams R (1986) Learning internal representations by error propagation. In: Rumelhart D & McClelland J (eds) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1. MIT Press.

Rylander B & Foster J (2001) Computational complexity and genetic algorithms. Proc. of the World Science and Engineering Society's Conference on Soft Computing, Advances in Fuzzy Systems and Evolutionary Computation, 248–253.

Sammon J (1969) A nonlinear mapping for data structure analysis. IEEE Trans. on Computers 18: 401–409.

Saul L & Rahim M (1999) Maximum likelihood and minimum classification error factor analysis for automatic speech recognition. IEEE Trans. on Speech and Audio Processing, 82: 115–125.

Saul L & Roweis S (2003) Think globally, fit locally: unsupervised learning of nonlinear manifolds. Journal of Machine Learning Research 4: 119–155.

Seber G (1984) Multivariate Observations. Wiley, New York.

Siedlecki W & Sklansky J (1988) On automatic feature selection. International Journal of Pattern Recognition and Artificial Intelligence 22: 197–220.

Siegel S & Castellan N (1988) Nonparametric Statistics for the Behavioral Sciences: Second edition. New York: McGraw-Hill.

Silven O, Niskanen M & Kauppinen H (2003) Wood inspection with non-supervised clustering. Machine Vision and Applications 135-6: 275–285.

Smola A & Schölkopf B (1998) A tutorial on support vector regression. Technical Report NC2-TR-1998-030, Royal Holloway College, London, UK.

Somol P, Pudil P, Novovičová J & Paclik P (1999) Adaptive floating search methods in feature selection. Pattern Recognition Letters 20: 1,157–1,163.

Srivastava A (2004) Mixture density Mercer kernels: a method to learn kernels directly from data. Proc. of the Fourth SIAM International Conference on Data Mining.

Stearns S (1976) On selecting features or pattern classifiers. Proc. Third International Conference on Pattern Recognition, 71–75.

Tenenbaum J (1998) Mapping a manifold of perceptual observations. In: Jordan M, Kearns M & Solla S (eds) Advances in Neural Information Processing Systems, volume 10, 682–688. Cambridge, MA, MIT Press.

Tenenbaum J, de Silva V & Langford J (2000) A global geometric framework for nonlinear dimensionality reduction. Science 2905500: 2,319–2,323.

Thomaz C (2004) Maximum entropy covariance estimate for statistical pattern recognition. Ph.D. thesis, Department of Computing, Imperial College, London, UK.

Turtinen M, Pietikäinen M, Silven O, Mäenpää T & Niskanen M (2003) Paper characterization by texture using visualization-based training. The International Journal of Advanced Manufacturing Technology 2211-12: 890–898.

Vapnik V (1995) The Nature of Statistical Learning Theory. Springer, New York.
Wang W & Yang J (2005) Mining high-dimensional data. In: Maimon O & Rokach L (eds) Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers. Kluwer Academic Publishers.
Ward M (1994) Xmdvtool: integrating multiple methods for visualizing multivariate data. Proc. of IEEE Visualization, 326–333.
Webb A (1995) Multidimensional scaling by iterative majorization using radial basis functions. Pattern Recognition 285: 753–759.
Weiss Y (1999) Segmentation using eigenvectors: a unifying view. Proc. of the IEEE International Conference on Computer Vision, 975–982.
Wilkinson J (1965) Algebraic Eigenvalue Problem. Claredon Press, Oxford, England.
Wilson C & Garris M (1992) Handprinted character database 3. National Institute of Standards and Technology; Advanced Systems division. http://www.nist.gov/srd/nistsd19.htm.
Xing E, Ng A, Jordan M & Russell S (2003) Distance metric learning, with application to clustering with side-information. In: Becker S, Thrun S & Obermayer K (eds) Advances in Neural Information Processing, volume 15, 505–512. MIT Press, Cambridge, MA.
Xu L (1999) Data mining, unsupervised learning and Bayesian Ying-Yang theory. Proc. of the International Conference on Neural Networks, 2,520–2,525.
Yang J & Honavar V (1998) Feature subset selection using a genetic algorithm. IEEE Intelligent Systems 132: 44–49.
Zar J (1996) Biostatistical Analysis: Third edition. Prentice Hall, New Jersey.
Zhang Y, Zhang C & Zhang D (2004) Distance metric learning by knowledge embedding. Pattern Recognition 371: 161–163.
Zheng A (2000) Deconstructing motion. Technical report, EECS Department, UC Berkeley.

Appendix 1 Description of the data sets used in the experiments

– **S-curve, swiss-roll, c-curve**
S-curve, swiss-roll and c-curve are artificially generated data sets possessing all the properties of a well-sampled manifold that is ideal input for LLE algorithm. Each of the data sets is composed of 2,000 three-dimensional points forming two-dimensional flat surfaces, demonstrated in Figure 25.

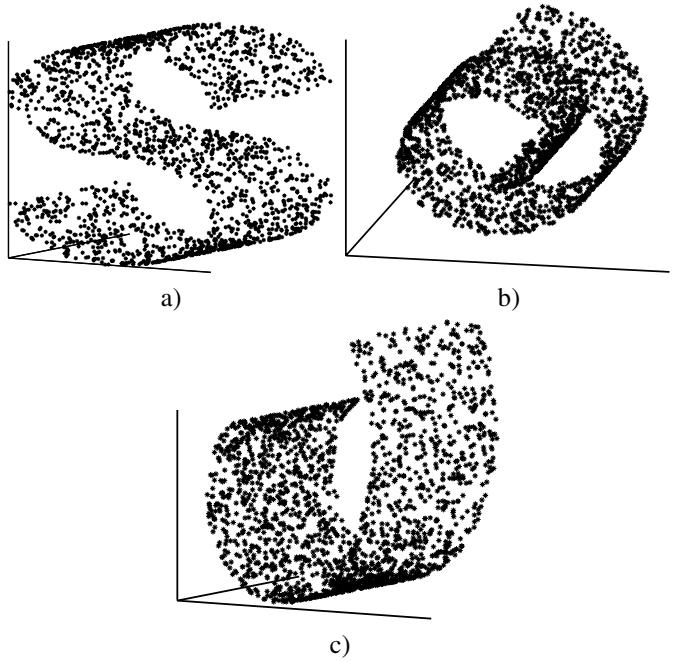


Fig. 25. **a) s-curve, b) swiss-roll, and c) c-curve data sets.**

– **UCI data sets**
Short descriptions of UCI[1] data sets are given. A more detailed description can be found in (Hettich *et al.* 1998).

  – **Iris data**: The data contains 150 samples composed of 4 measurements. The attribute information is represented by the following parameters (in cm): 1) sepal length, 2) sepal width, 3) petal length, and 4) petal width. There were 3 pattern classes (50 samples in each) corresponding to three different types of iris: Virginica, Setosa, and Versicolor. One class is linearly separable from the other two, while the latter two are not linearly separable from each other.
  – **Liver data**: The data contains 345 samples possessing 6 characteristics: 5 blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption and the number of alcoholic beverages drunk per day. The data is divided into two classes containing 145 and 200 samples.
  – **Diabetes data**: The data set contains the distribution for 70 sets of data recorded on diabetes patients (several weeks' to months' worth of glucose, insulin, and lifestyle data per patient). The data is represented by 768 samples divided into 2 classes (268 and 500) of dimensionality 8.
  – **Glass data**: The data represents 6 types of glass defined in terms of their oxide content (i.e. Na, Fe, K etc.). The total number of samples is 214, the dimensionality of which is equal to 9.
  – **Wine data**: This data is constructed from results obtained with chemical analysis of wine grown in the same region in Italy but derived from three different cultivars. The chemical analysis defined the quantities of 13 components found in each of three types of wine containing 59, 71 and 48 samples.
  – **Vehicle data**: The data contains four classes represented by four "Corgie" model vehicles: a double decker bus, a Cheverolet van, a Saab 9000 and an Opel Manta 400. This particular combination of vehicles was chosen with the expectation that the bus, van and either one of the cars would be readily distinguishable, but it would be more difficult to distinguish between the cars. 18 features for each of 846 samples were extracted from the silhouette of a car using the hierarchical image processing system, which extracts a combination of scale independent features utilizing both classical moment based measures, such as scaled variance, skewness and kurtosis about the major/minor axes, and heuristic measures, such as hollows, circularity, rectangularity and compactness.
  – **Ionosphere data**: The radar data was taken by a system in Goose Bay, Labrador. Each of the 351 data samples is described by 34-dimensional feature vector and belongs to one of 2 different classes of sizes 219 and 126.
  – **Splice data**: Splice junctions are points on a DNA sequence at which superfluous DNA is removed during the process of protein creation in higher organisms. The splice data set contains 3,188 samples with a dimensionality of 60, which are divided into 3 classes.
  – **Sonar data**: This file contains 208 signals corresponding to sonar returns from rocks. The data samples are divided into 2 classes (97 and 111). The dimensionality of the data is 60.

[1] University of California, Irvine.

– **Optdigits data**: The data contains 5,620 examples of handwritten digits from 0 to 9. Each 32×32 binary image is divided into non-overlapping blocks of size 4×4 pixels, and the amount object pixels are counted in each block. The resulting 8×8 image reshaped into the vector of size 1×64 is taken as a data input.

– **Chromosomes**
The data contains 30 gray-values sampled from chromosome banding profiles. The 2,520 data items are divided into 24 classes (De Ridder *et al.* 2003a).

– **Natural and structured textures**
The two textures data sets contain 12×12-pixel gray-value image patches of either natural (i.e., irregular) or structured (i.e., regular) Brodatz textures representing a collection of single high-resolution images (De Ridder 2001). The original 256×256 pixel images were rescaled to a size of 192×192 pixels using bicubic interpolation, and divided into 12×12 elements. For each image, the gray value range was rescaled to [0,1]. Note that there is no intra class illumination or orientation variation within the data class, since all samples are obtained from the same image. They are in 6 classes in natural and structured texture data sets, each of which contains 3,000 samples. Figure 26 demonstrates the texture images from which the data was obtained (Brodatz 1966).

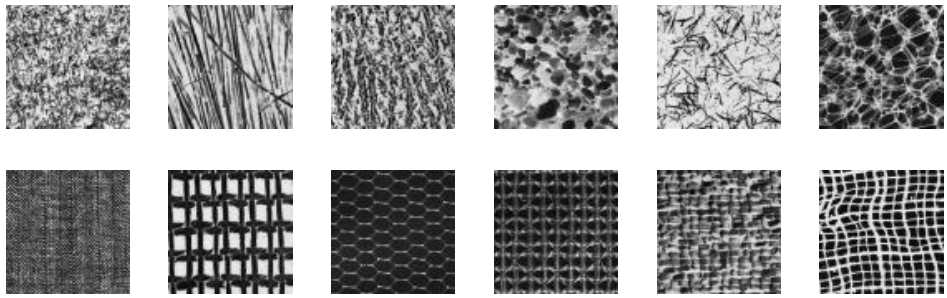


Fig. 26. **Natural and structured textures.**

– **NIST digits**
The handwritten characters data set used in the experiments was taken from the Special Database 3 of the U.S. National Institute of Standards and Technology (NIST) data set (Wilson & Garris 1992). 625 items for each of ten data classes containing handwritten digits from 0 to 9 were chosen from the original 2,800 samples and were preprocessed as in (De Ridder 2001). The dimensionality of the data space is 256, since the final images are 16×16 pixels. Several data samples are demonstrated in Figure 27.

– **Frey faces**
The Frey faces data set is represented by 1,965 images of Brendan's face, 20×28, taken from sequential frames of a small video. These images contain different facial expression, several examples are shown in Figure 28. The dimensionality of the data is 560, since the feature vector is organized by grayscale pixel values of the data images.

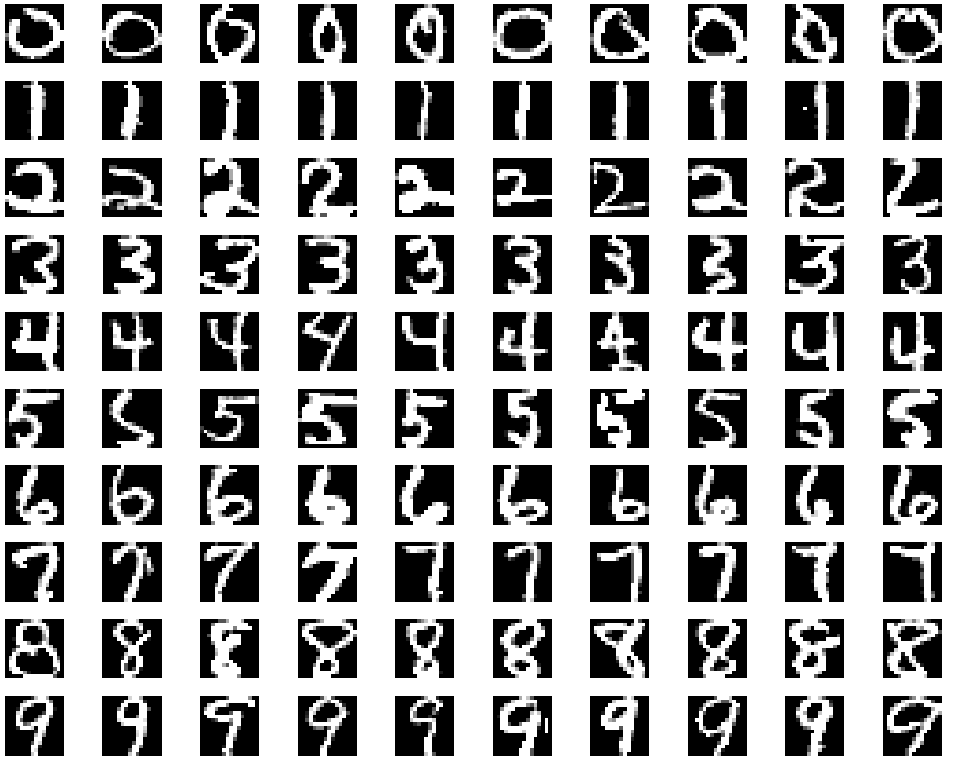


Fig. 27. **Examples of handwritten digit images from the NIST data set.**

Fig. 28. **Examples of Frey faces.**

– **Wood data**
The wood data set contains eight classes of wood surface images from the database of the University of Oulu (Poskanzer 1991). It includes 1,947 images of clean and defective surfaces, each of which is an RGB image of 16×16 pixels. A data sample

is described by values from different color channels concatenated into one feature vector of dimensionality 768. Figure 29 demonstrates a few samples from the data set in a grayscale level.

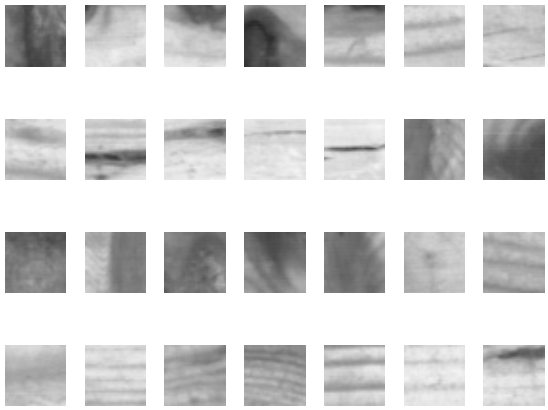


Fig. 29. **Examples of wood surface images.**

– **MNIST digits**
The modified NIST (MNIST) database of handwritten digits (LeCun 1998) consists of handwritten digits images from 0 to 9 (each of 28×28 pixels in size) together with their class labels. Examples of the MNIST images are given in Figure 30.
Though the MNIST images are grayscale as a result of the anti-aliasing normalization, the vast majority of pixels have just one of two values (either 0 or 255) so that the images can be considered as almost binary. The MNIST's training set is composed of 30,000 images from NIST's Special Database 1 and 30,000 images from NIST's Special Database 3 written by 250 persons. The MNIST's test set consists of 5,000 images from NIST's Special Database 1 and 5,000 images from NIST's Special Database 3.
– **Paper data**
The data set consists of 1,004 light through paper images forming four different quality classes (Figure 31). The size of the images are 760×570 pixels with 8 bits per pixel. The multiresolution local binary patterns are extracted from the images and used as features (Ojala *et al.* 2002); hence, the data dimensionality is 857.
– **COIL-20 data**
The Columbia Object Image Library (COIL-20) is a database of greyscale images of 20 objects (Nene *et al.* 1996). The objects were placed on a motorized turntable against a black background. The turntable was rotated through 360 degrees to vary object pose with respect to a fixed camera. Images of the objects were taken at pose intervals of 5 degrees, which corresponds to 72 images per object (Nene *et al.* 1996). The final data contains 1,440 object images of 128×128 pixels (Figure 32). The original resolution of each image was reduced to 32×32 pixels. The grayscale pixel values are used as features; hence, the data dimensionality is 1,024.

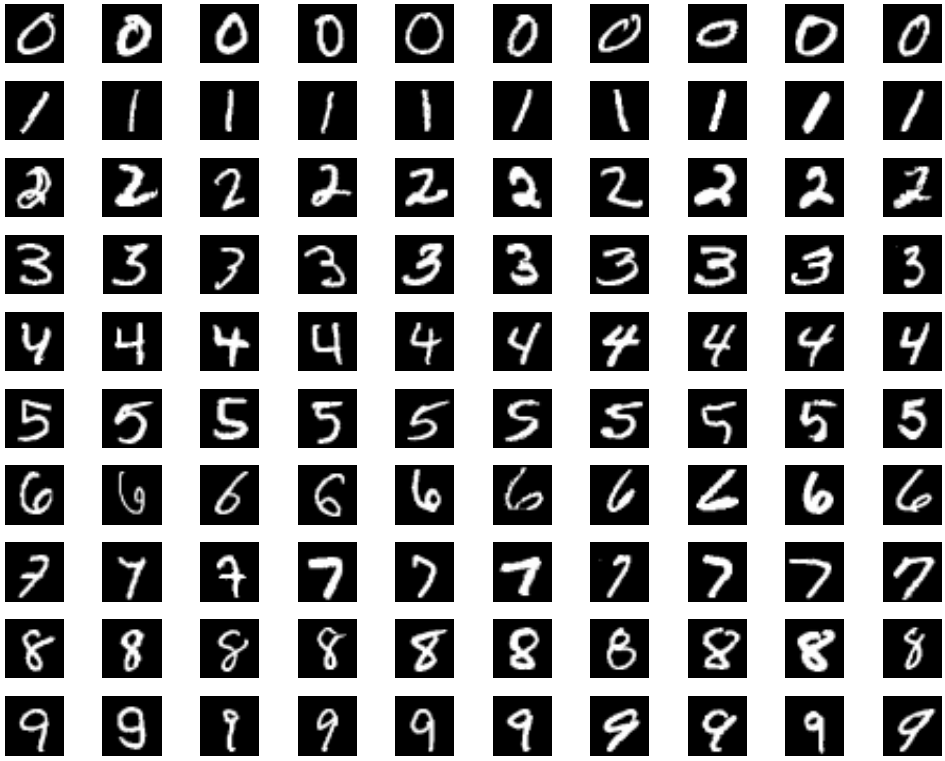


Fig. 30. **Examples of handwritten digit images from the MNIST data set.**

– **Olga's and Oleg's face data**
Olga's and Oleg's face data sets were taken with a Sony DFW-X700 digital camera under fixed illumination conditions. Each sequence consists of images of 70×90 pixels showing one person slowly rotating the head from left to right, while trying to fix the chin at the same level in order to obtain one degree of freedom: the angle of rotation. In spite of the fact that initial conditions for capturing these data sets were the same, Oleg's face data is uniformly distributed, while Olga's face 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 face data set to be non-uniform. The original RGB images were converted to grayscale without losing much useful information, and some of the grayscale data images are shown in Figures 33 and 34. There are 1,130 and 1,200 samples of dimensionality 6,300 in Oleg's and Olga's data sets, respectively.
– **Abdenour's face data**
The face data set is comprised of 500 120×100 pixels grayscale images. Each image was first transformed into a column vector ($D = 12{,}000$). All vectors were then concatenated to form the input matrix so that the original space was composed of vectors of intensity values. Typical samples from this data set are shown in Figure 35.

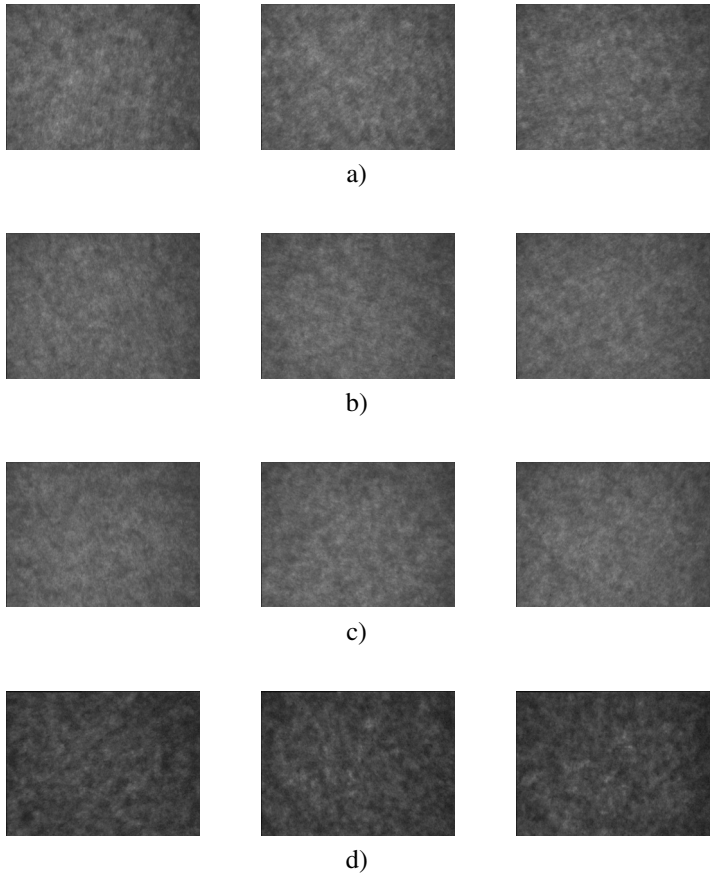


Fig. 31. **Examples of paper images representing four different qualities of paper (a,b,c,d).**

Like Oleg's data set, the face data is considered to be uniformly distributed, since the velocity of head rotation was invariable while capturing the images.

Fig. 32. Examples images from the COIL-20 data set.

Fig. 33. Representative grayscale face images from Olga's face data.

Fig. 34. Representative grayscale face images from Oleg's face data.

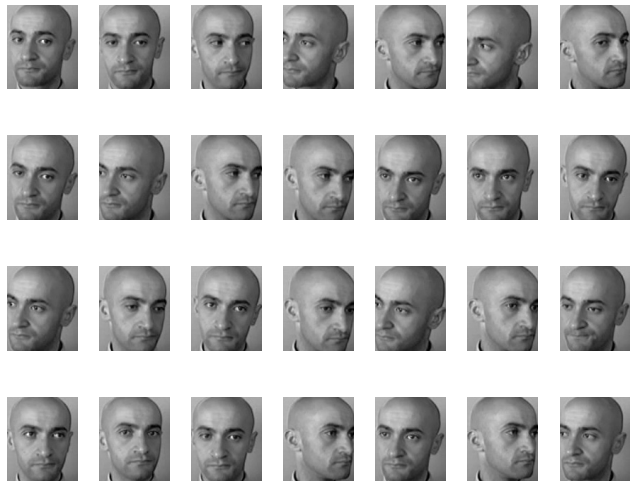


Fig. 35. **Representative grayscale face images from Abdenour's face data.**

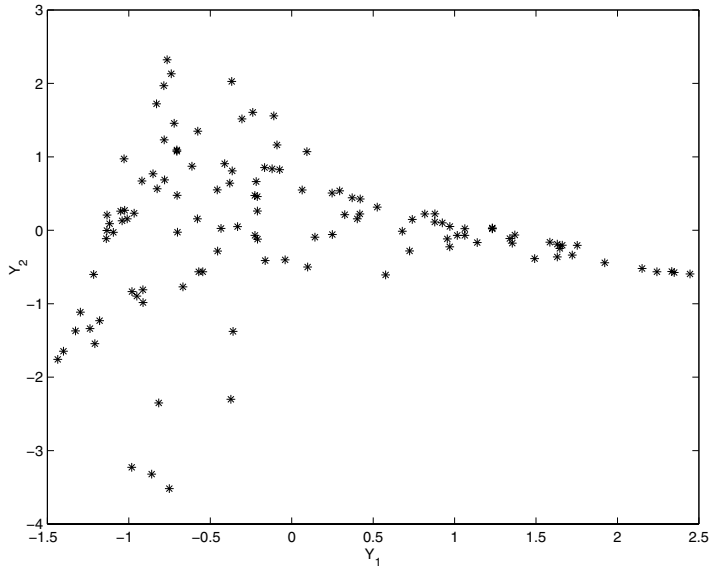
Appendix 2 The ill-conditioned eigenproblem

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

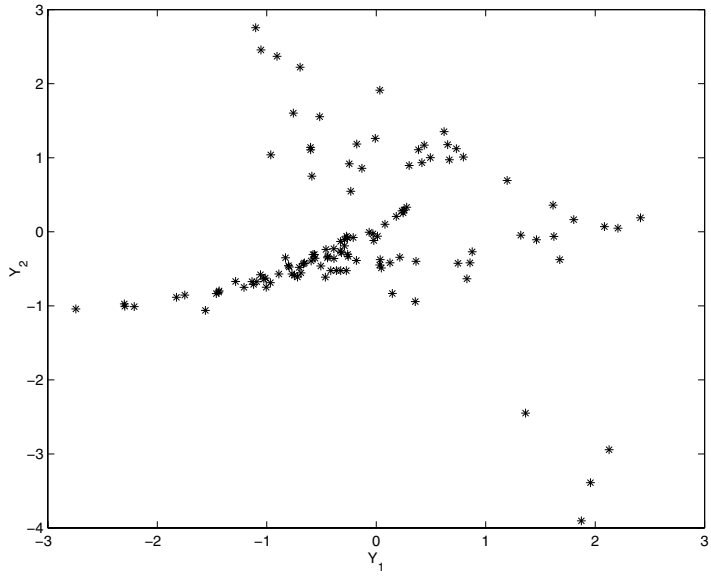
$$A_0 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & \varepsilon \\ 0 & \varepsilon & 1 \end{pmatrix} \text{ to } A_1 = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1+\varepsilon \end{pmatrix}$$

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.

Figure 36 demonstrates the consequence of ill-conditioning. First, the two bottom eigenvectors of a particular matrix were found. These eigenvectors form 2D space in Figure 36 (a). Then the elements of the matrix were changed by adding or subtracting very small values. Finally, two bottom eigenvectors of the modified matrix were computed. They are shown in Figure 36 (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 $-2.36 \cdot 10^{-33}$.



a)



b)

Fig. 36. 2D spaces formed by the two bottom eigenvectors of a) the initial matrix, and b) the modified matrix. The difference between norms of these matrices is equal to $-2.36 \cdot 10^{-33}$.

Appendix 3 Intrinsic dimensionality estimation by packing numbers

The goal of this intrinsic dimensionality estimation is to find the number of independent parameters needed to represent a data sample. Kégl has proposed an algorithm for calculating the intrinsic dimensionality of data by assuming that it depends on resolutions from which the data is considered (Kégl 2003). Figure 37 provides an intuitive example demonstrating the dependence. The algorithm calculates the intrinsic dimensionality of the data by discovering the manifold structure and requires two parameters, called scales, to be set.

The algorithm uses the capacity dimension from the concept of topological dimensionality to calculate the intrinsic dimensionality of the data[1]:

Definition 1. Given a metric space X with distance metric $d(\cdot,\cdot)$, the r-covering number $N(r)$ of a set $S \subset X$ is the minimum number of open balls $B(\boldsymbol{x}_0, r) = \{\boldsymbol{x} \in X \mid d(\boldsymbol{x}_0, \boldsymbol{x}) < r\}$ whose union is a covering of S. Then the capacity dimension of a subset S of a metric space X is

$$D_{cap} = -\lim_{r \to 0} \frac{\log N(r)}{\log r}. \tag{A3.26}$$

Since it is computationally difficult to find the covering numbers, even for a finite data set, D_{cap} is redefined by using packing numbers. Given a metric space X with distance metric $d(\cdot,\cdot)$, a set $V \subset X$ is said to be r-separated if $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq r$ for all distinct $\boldsymbol{x}_i, \boldsymbol{x}_j \in V$. The r-packing number $M(r)$ of a set $S \subset X$ is defined as the maximum cardinality of r-separated subsets of S. Therefore, the capacity dimension can be written as $D_{cap} = -\lim_{r \to 0} \frac{\log M(r)}{\log r}$. But in the case of a finite set of samples the zero limit ($r \to 0$) cannot be achieved, consequently the definition of scale-dependent capacity is introduced as follows.

Definition 2. The scale-dependent capacity dimension of a finite set $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is

$$\hat{D}_{cap}(r_1, r_2) = -\frac{\log M(r_2) - \log M(r_1)}{\log r_2 - \log r_1}. \tag{A3.27}$$

The scale parameters, (r_1, r_2), are chosen heuristically, but there are the following restrictions. First, if the difference between two scales ($\Delta_r = \mid r_1 - r_2 \mid$) is too small, then the

[1] Notations used in this appendix do not relate to the notations used in the main text of the thesis.

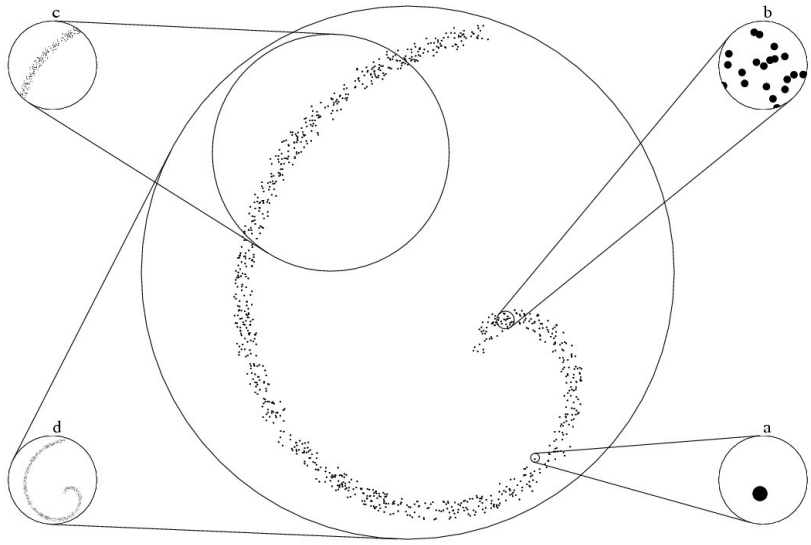
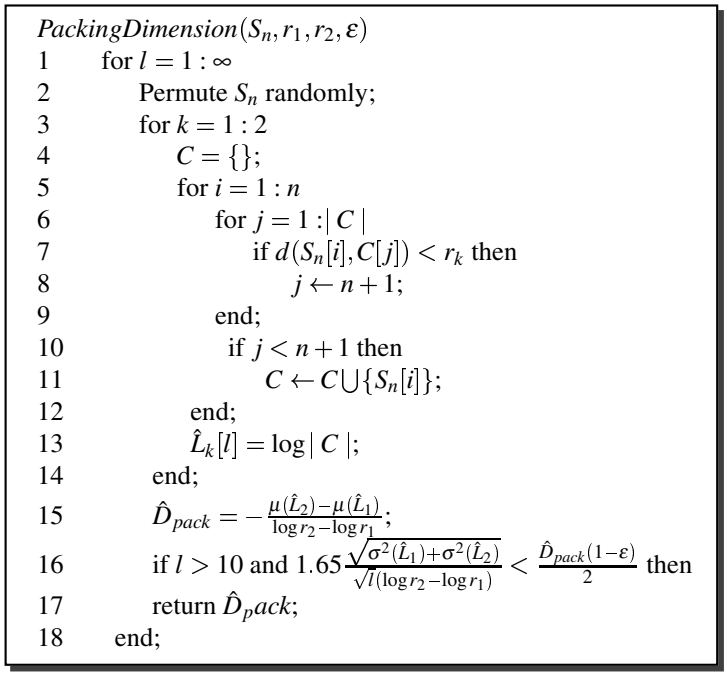


Fig. 37. Intrinsic dimension d at different resolutions. a) The data looks zero-dimensional at very low scale. b) At noise level, the intrinsic dimensionality is equal to 2. c) The dimensionality of the manifold is 1 according to the right scale in terms of noise and curvature. d) At very large scale the global dimension dominates, $d = 2$. (Courtesy of Baláz Kégl.)

variance of the estimate of the intrinsic dimensionality is large, thus, small Δ_r leads to a highly noisy final result. Second, if Δ_r is too large, then the final estimate represents the global dimensionality of the data. These effects are demonstrated in Figure 37.

Figure 38 describes the steps of the algorithm calculating the intrinsic dimensionality of the data. The algorithm is robust to the presence of noise and to the distribution of the data from which points are sampled. The latter property is the most desired one since the Gaussian distribution is not always the case.



PackingDimension$(S_n, r_1, r_2, \varepsilon)$
for $l = 1 : \infty$
  Permute S_n randomly;
  for $k = 1 : 2$
    $C = \{\}$;
    for $i = 1 : n$
      for $j = 1 :\mid C \mid$
        if $d(S_n[i], C[j]) < r_k$ then
          $j \leftarrow n + 1$;
      end;
      if $j < n + 1$ then
        $C \leftarrow C \bigcup \{S_n[i]\}$;
    end;
    $\hat{L}_k[l] = \log \mid C \mid$;
  end;
  $\hat{D}_{pack} = -\frac{\mu(\hat{L}_2) - \mu(\hat{L}_1)}{\log r_2 - \log r_1}$;
  if $l > 10$ and $1.65 \frac{\sqrt{\sigma^2(\hat{L}_1) + \sigma^2(\hat{L}_2)}}{\sqrt{l}(\log r_2 - \log r_1)} < \frac{\hat{D}_{pack}(1-\varepsilon)}{2}$ then
  return $\hat{D}_pack$;
end;

Fig. 38. The algorithm for intrinsic dimensionality estimation using packing numbers, where μ and σ^2 denote the mean and variance, correspondingly, and ε is the tolerance.

Appendix 4 Intrinsic mean

One way of seeing how the logmap may be estimated is to consider some results related to how the intrinsic mean is computed (Karcher 1977, Fletcher *et al.* 2004). Let $\{\boldsymbol{x}_i\}$ be N data points in a manifold $\mathscr{M}$ and seek the minimizer to the function:

$$f(\boldsymbol{x}) = \frac{1}{2N}\sum_{i=1}^{N} d^2(\boldsymbol{x},\boldsymbol{x}_i). \tag{A4.28}$$

It is then shown that under the appropriate assumptions of convexivity the gradient of f is (Karcher 1977):

$$\bigtriangledown f(\boldsymbol{x}) = -\frac{1}{N}\sum_{i=1}^{N} \log_{\boldsymbol{x}} \boldsymbol{x}_i. \tag{A4.29}$$

From this we directly see that for $N = 1$, we have

$$\log_{\boldsymbol{x}_p}(\boldsymbol{x}) = -\frac{1}{2}\bigtriangledown_{\boldsymbol{x}_p} d^2(\boldsymbol{x}_p,\boldsymbol{x}). \tag{A4.30}$$

which shows that the logmap of $\boldsymbol{x}$ calculated at the base point $\boldsymbol{x}_p$, is precisely $-1/2$ times the gradient of the squared distance function. The approach for calculating S-LogMap in this thesis is based on this result (Brun 2006).

Appendix 5 Global and local intrinsic dimensionality estimated by PCA

The global intrinsic dimensionality of a data set estimated by PCA is equal to the number of retained eigenvectors corresponding to the m largest eigenvalues, λ_i, of the sample covariance matrix (Hotelling 1933). In practice, m might be either specified by a user or calculated based on the amount of variance, v, that should be retained by projecting the data set:

$$v \leq \frac{\sum_{j=1}^{m} \lambda_j}{\sum_{i=1}^{D} \lambda_i}, \tag{A5.31}$$

where D is the original dimensionality of the given data set. Usually, v is set to 0.90 or 0.95.

Local intrinsic dimensionality is estimated by performing eigenanalysis on every local covariance matrix. For every data sample $\boldsymbol{x}_i$ the value of m_i is found by Eq. A5.31. To obtain overall local intrinsic dimensionality of the data m, one may use, for example, majority voting over the entire data set.

Book orders:
OULU UNIVERSITY PRESS
P.O. Box 8200, FI-90014
University of Oulu, Finland

Distributed by
OULU UNIVERSITY LIBRARY
P.O. Box 7500, FI-90014
University of Oulu, Finland

UNIVERSITY OF OULU P.O. Box 7500 FI-90014 UNIVERSITY OF OULU FINLAND

ACTA UNIVERSITATIS OULUENSIS

ISBN 951-42-8040-7 (Paperback)
ISBN 951-42-8041-5 (PDF)
ISSN 0355-3213 (Print)
ISSN 1796-2226 (Online)