An Introduction to Dimensionality Reduction Using Matlab

L.J.P. van der Maaten

Report MICC 07-07

Universiteit Maastricht
MICC/IKAT
P.O. Box 616
6200 MD Maastricht
The Netherlands

Maastricht, July 2007

An Introduction to Dimensionality Reduction Using Matlab

Laurens van der Maaten

MICC, Maastricht University
P.O. Box 616, NL-6200 MD Maastricht, The Netherlands
E-mail: `l.vandermaaten@micc.unimaas.nl`

Abstract

Dimensionality reduction is an important task in machine learning, for it facilitates classification, compression, and visualization of high-dimensional data by mitigating undesired properties of high-dimensional spaces. Over the last decade, a large number of new (nonlinear) techniques for dimensionality reduction have been proposed. Most of these techniques are based on the intuition that data lies on or near a complex low-dimensional manifold that is embedded in the high-dimensional space. New techniques for dimensionality reduction aim at identifying and extracting the manifold from the high-dimensional space. A systematic empirical evaluation of a large number of dimensionality reduction techniques has been presented in [86]. This work has led to the development of the Matlab Toolbox for Dimensionality Reduction, which contains implementations of 27 techniques for dimensionality reduction. In addition, the toolbox contains implementation of 6 intrinsic dimensionality estimators and functions for out-of-sample extension, data generation, and data prewhitening. The report describes the techniques that are implemented in the toolbox in detail. Furthermore, it presents a number of examples that illustrate the functionality of the toolbox, and provide insight in the capabilities of state-of-the-art techniques for dimensionality reduction.

Contents

1 Introduction

Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality. Ideally, the reduced representation has a dimensionality that corresponds to the intrinsic dimensionality of the data. The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data [31]. Dimensionality reduction is important in many domains, since it facilitates classification, visualization, and compression of high-dimensional data, by mitigating the curse of dimensionality and other undesired properties of high-dimensional spaces [46].
In recent years, a large number of new techniques for dimensionality reduction have been proposed [2, 7, 8, 14, 24, 36, 39, 40, 54, 70, 73, 72, 80, 78, 89, 90, 93, 92]. Many of these techniques have the ability to deal with complex nonlinear data manifolds and therefore constitute traditional techniques for dimensionality reduction, such as Principal Components Analysis (PCA) and Linear Discriminant Analysis (LDA) [25]. In particular for real-world data, dimensionality reduction techniques that are capable of dealing with complex nonlinear data manifolds may offer an advantage. Various studies have shown that nonlinear techniques outperform their linear counterparts on artificial tasks that are highly nonlinear. In order to make the recent advances in dimensionality reduction available to a broader community, we developed the Matlab Toolbox for Dimensionality Reduction, which is described in this report.
The Matlab Toolbox for Dimensionality Reduction is a toolbox with Matlab implementations of 27 techniques for dimensionality reduction, 6 techniques for intrinsic dimensionality reduction estimation, and additional functions for out-of-sample extension, data generation, and data prewhitening. The report provides a description of all techniques that are implemented in the toolbox, as well as a number of examples on how to use the toolbox. Note that this report is not intended to provide a comparative review of techniques for dimensionality reduction. A comparative review on state-of-the-art techniques for dimensionality reduction can be found in [86].
The Matlab Toolbox for Dimensionality Reduction includes all main techniques for dimensionality reduction, except self-organizing maps [51] and their probabilistic extension GTM [12], because we consider these techniques to be clustering techniques[1]. Neither does the toolbox include techniques for blind-source separation such as ICA [9]. Techniques for dimensionality reduction that are missing in the toolbox (but might be added later) include principal curves [17], kernel maps [76], FastMap [27], Geodesic Nullspace Analysis [14], and the technique for global alignment of linear models presented in [69].
The outline of the remainder of this report is as follows. In section 2, we give an introduction to dimensionality reduction. Section 3 discusses the techniques for dimensionality reduction that are implemented in the toolbox. Section 4 describes 6 techniques for intrinsic dimensionality estimation that are implemented in the toolbox. In Section 5, we provide examples of the use of the toolbox, along with some results on artificial datasets. Section 6 concludes the report. For installation instructions and information on possible pitfalls we refer to the appendices.

[1]A theoretical discussion of the link between clustering and dimensionality reduction can be found in [10].

2 Dimensionality reduction

The problem of dimensionality reduction can be defined as follows. Suppose we have a $n \times D$ matrix X consisting of n datavectors x_i with dimensionality D, and that this dataset has the intrinsic dimensionality d (where $d < D$, and often $d \ll D$). In geometric terms, intrinsic dimensionality means that the points in dataset X are lying on or near a manifold with dimensionality d that is embedded in the D-dimensional space. A dimensionality reduction technique transforms dataset X into a new dataset Y with dimensionality d, while retaining the geometry of the data as much as possible. In general, neither the geometry of the embedded manifold, nor the intrinsic dimensionality d of the dataset X are known. Therefore, dimensionality reduction is an ill-posed problem that can only be solved by assuming certain properties of the data (such as its intrinsic dimensionality).

We illustrate dimensionality reduction by an example. Consider a dataset X that consists of n images of an object that were depicted under n different orientations. The images have a size of 32×32 pixels, and thus form points in a 1,024-dimensional space. Since there is only one intrinsic dimension (viz., the orientation of the depicted object), the datapoints x_i lie on or near a curved 1-dimensional manifold. The curvedness of the data manifold follows from the observation that the image corresponding to an orientation of 0 degrees is similar to the image corresponding to an orientation of 360 degrees. The aim of dimensionality reduction techniques is to identify the 1-dimensional data manifold that is embedded in the 1,024-dimensional space and to embed it into a space of lower dimensionality, thereby retaining the geometry of the data manifold.

3 Techniques for dimensionality reduction

In section 2, we defined the problem of dimensionality reduction. In this section, we discuss 23 techniques for dimensionality reduction that are implemented in the toolbox[2]. The techniques presented in the section can be subdivided into five main groups: (1) traditional linear techniques, (2) global nonlinear techniques, (3) local nonlinear techniques, (4) extensions and variants of local nonlinear techniques, and (5) techniques based on the alignment of a collection of linear models. Note that our subdivision of techniques for dimensionality reduction is not the only imagineable subdivision, and that there exist many similarities between the various techniques.

The outline of the remainder of this section is as follows. In subsection 3.1, we discuss the two traditional linear techniques for dimensionality reduction, viz., PCA and LDA. In subsection 3.2, nine global nonlinear techniques for dimensionality reduction are discussed. Subsection 3.3 describes four local techniques for dimensionality reduction. For local nonlinear techniques for dimensionality reduction, several extensions have been presented, as well as linear versions of the local techniques. Two extensions for local nonlinear techniques and three linear variants are presented in 3.4. Three techniques that construct a low-dimensional data representation based on the alignment of a number of linear models are described in subsection 3.5.

Throughout the report, we denote a high-dimensional datapoint by x_i, where x_i is the ith row of the D-dimensional data matrix X. The low-dimensional counterpart of x_i is denoted by y_i, where y_i is the ith row of the d-dimensional data matrix Y.

3.1 Traditional linear techniques

Two linear techniques for dimensionality reduction are already known in statistics since the beginning of the 20^{th} century. These techniques are Principal Components Analysis (originally known as the Karhunen-Loève Transform) and Linear Discriminant Analysis (originally known as the Fisher mapping). Because of their importance in the field of statistics, we discuss these two techniques separately from novel linear techniques such as LPP, NPE, and LLTSA (these techniques are discussed in subsection 3.4). PCA is described in subsubsection 3.1.1 and LDA is discussed in subsubsection 3.1.2.

3.1.1 PCA

Principal Components Analysis (PCA) [43] constructs a low-dimensional representation of the data that describes as much of the variance in the data as possible. This is done by finding a linear basis of reduced dimensionality for the data, in which the amount of variance in the data is maximal.

In mathematical terms, PCA attempts to find a linear transformation T that maximizes $T^T \operatorname{cov}_{X-\bar{X}} T$, where $\operatorname{cov}_{X-\bar{X}}$ is the covariance matrix of the zero mean data X. It can be shown that this linear mapping is formed by the d principal eigenvectors (i.e., principal components) of the covariance matrix of the zero-mean data[3]. Hence, PCA solves the eigenproblem

$$\underset{X-\bar{X}}{\operatorname{cov}} v = \lambda v \tag{1}$$

The eigenproblem is solved for the d principal eigenvalues λ. The corresponding eigenvectors form the columns of the linear transformation matrix T. The low-dimensional data representations y_i of the datapoints x_i are computed by mapping them onto the linear basis T, i.e., $Y = (X - \bar{X})T$.

PCA has been successfully applied in a large number of domains such as face recognition [85], coin classification [44], and seismic series analysis [66]. The main drawback of PCA is that the size of the covariance matrix is proportional to the dimensionality of the datapoints. As a result, the computation of the eigenvectors might be infeasible for very high-dimensional data (under the assumption that $n > D$). Approximation methods, such as Simple PCA [65], deal with this problem by applying an iterative Hebbian approach in order to estimate the principal eigenvectors of the

[2]The reader should note that there are 27 techniques that are implemented in the toolbox. However, in the toolbox, there are 3 PCA implementations, 2 Isomap implementations, and 2 autoencoder implementations. As a result, only 23 techniques are discussed in the report.

[3]PCA maximizes $T^T \operatorname{cov}_{X-\bar{X}} T$ with respect to T, under the constraint that $|T| = 1$. This constraint can be enforced by introducing a Lagrange multiplier λ. Hence, an unconstrained maximization of $T^T \operatorname{cov}_{X-\bar{X}} T + \lambda(1 - T^T T)$ is performed. A stationary point of this quantity is to be found when $\operatorname{cov}_{X-\bar{X}} T = \lambda T$.

covariance matrix. Alternatively, PCA can also be rewritten in a probabilistic framework, allowing for performing PCA by means of an EM algorithm [83]. The reader should note that probabilistic PCA is closely related to factor analysis [3]. In the toolbox, a traditional PCA implementation is available, as well as implementations of Simple PCA and probabilistic PCA.

3.1.2 LDA

Linear Discriminant Analysis (LDA) [28] attempts to maximize the linear separability between datapoints belonging to different classes. In contrast to most other dimensionality reduction techniques, LDA is a supervised technique.
LDA finds a linear mapping M that maximizes the linear class separability in the low-dimensional representation of the data. The criteria that are used to formulate linear class separability in LDA are the within-class scatter S_w and the between-class scatter S_b, which are defined as

$$S_w = \sum_c p_c \operatorname*{cov}_{X^c - \bar{X}^c} \tag{2}$$

$$S_b = \sum_c \operatorname*{cov}_{\bar{X}^c} = \operatorname*{cov}_{X - \bar{X}} - S_w \tag{3}$$

where p_c is the class prior of class label c, $\text{cov}_{X^c - \bar{X}^c}$ is the covariance matrix of the zero mean datapoints x_i assigned to class $c \in C$ (where C is the set of possible classes), $\text{cov}_{\bar{X}^c}$ is the covariance matrix of the cluster means, and $\text{cov}_{X-\bar{X}}$ is the covariance matrix of the zero mean data X. LDA optimizes the ratio between the within-class scatter S_w and the between-class scatter S_b in the low-dimensional representation of the data, by finding a linear mapping M that maximizes the so-called Fisher criterion

$$\frac{T^T S_b T}{T^T S_w T} \tag{4}$$

This maximization can be performed by solving the generalized eigenproblem

$$S_b v = \lambda S_w v \tag{5}$$

for the d largest eigenvalues (under the requirement that $d < |C|$). The eigenvectors v form the columns of the linear transformation matrix T. The low-dimensional data representation Y of the datapoints in X can be computed by mapping them onto the linear basis T, i.e., $Y = (X - \bar{X})T$.
LDA has been successfully applied in a large number of classification tasks. Successful applications include speech recognition [34], mammography [15], and document classification [84].

3.2 Global nonlinear techniques

Global nonlinear techniques for dimensionality reduction are techniques that attempt to preserve global properties of the data (similar to PCA and LDA), but that are capable of constructing nonlinear transformations between the high-dimensional data representation X and its low-dimensional counterpart Y. The subsection presents nine global nonlinear techniques for dimensionality reduction: (1) MDS, (2) SPE, (3) Isomap, (4) FastMVU, (5) Kernel PCA, (6) GDA, (7) diffusion maps, (8) SNE, and (9) multilayer autoencoders. The techniques are discussed in subsubsection 3.2.1 to 3.2.9.

3.2.1 MDS

Multidimensional scaling (MDS) [19, 52] represents a collection of nonlinear techniques that maps the high-dimensional data representation to a low-dimensional representation while retaining the pairwise distances between the datapoints as much as possible. The quality of the mapping is expressed in the stress function, a measure of the error between the pairwise distances in the low-dimensional and high-dimensional representation of the data. Two important examples of stress functions (for metric MDS) are the raw stress function and the Sammon cost function. The raw stress function is defined by

$$\phi(Y) = \sum_{ij} (\|x_i - x_j\| - \|y_i - y_j\|)^2 \tag{6}$$

in which $\|x_i - x_j\|$ is the Euclidean distance between the high-dimensional datapoints x_i and x_j and $\|y_i - y_j\|$ is the Euclidean distance between the low-dimensional datapoints y_i and y_j. The Sammon cost function is given by

$$\phi(Y) = \frac{1}{\sum_{ij}\|x_i - x_j\|} \sum_{ij} \frac{(\|x_i - x_j\| - \|y_i - y_j\|)^2}{\|x_i - x_j\|} \tag{7}$$

The Sammon cost function differs from the raw stress function in that it puts more emphasis on retaining distances that were originally small. The minimization of the stress function can be performed using various methods, such as the eigendecomposition of a pairwise dissimilarity matrix, the conjugate gradient method, or a pseudo-Newton method [19].
MDS is widely used for the visualization of data, e.g., in fMRI analysis [77] and in molecular modelling [88]. The popularity of MDS has led to the proposal of variants such as SPE [2] and FastMap [27].

3.2.2 SPE

Stochastic Proximity Embedding (SPE) is an iterative algorithm that minimizes the MDS raw stress function. SPE differs from MDS in the efficient rule it employs to update the current estimate of the low-dimensional data representation. In addition, SPE can readily be applied in order to retain only distances in a neighborhood graph defined on the graph, leadin to behavior that is comparable to, e.g., Isomap (see subsubsection 3.2.3).
SPE minimizes the MDS raw stress function that was given in Equation 6. For convenience, we rewrite the raw stress function as

$$\phi(Y) = \sum_{ij}(d_{ij} - r_{ij})^2 \tag{8}$$

where r_{ij} is the proximity between the high-dimensional datapoints x_i and x_j (computed by $r_{ij} = \frac{\|x_i - x_j\|}{\max R}$), and d_{ij} is the Euclidean distance between their low-dimensional counterparts y_i and y_j in the current approximation of the embedded space. SPE can readily be performed in order to retain solely distances in a neighborhood graph G defined on the data, by setting d_{ij} and r_{ij} to 0 if $(i, j) \notin G$. Using this setting, SPE behaves comparable to techniques based on neighborhood graphs. Nonlinear dimensionality reduction techniques based on neighborhood graphs are discussed in more detail later.
SPE performs an iterative algorithm in order to minimize the raw stress function defined above. The initial positions of the points y_i are selected randomly in $[0, 1]$. An update of the embedding coordinates y_i is performed by randomly selecting s pairs of points (y_i, y_j). For each pair of points, the Euclidean distance in the low-dimensional data representation Y is computed. Subsequently, the coordinates of y_i and y_j are updated in order to decrease the difference between the distance in the original space r_{ij} and the distance in the embedded space d_{ij}. The updating is performed using the following update rules

$$y_i = y_i + \lambda \frac{r_{ij} - d_{ij}}{2d_{ij} + \epsilon}(y_i - y_j) \tag{9}$$

$$y_j = y_j + \lambda \frac{r_{ij} - d_{ij}}{2d_{ij} + \epsilon}(y_j - y_i) \tag{10}$$

where λ is a learning parameter that decreases with the number of iterations, and ϵ is a regularization parameter that prevents divisions by zero. The updating of the embedded coordinates is performed for a large number of iterations (e.g., 10^5 iterations). The high number of iterations is feasible because of the low computational costs of the update procedure.

3.2.3 Isomap

Multidimensional scaling has proven to be successful in many applications, but it suffers from the fact that it is based on Euclidean distances, and does not take into account the distribution of the neighboring datapoints. If the high-dimensional data lies on or near a curved manifold, such as in the Swiss roll dataset [79], MDS might consider two datapoints as near points, whereas their distance over the manifold is much larger than the typical interpoint distance. Isomap [79] is a technique that resolves this problem by attempting to preserve pairwise geodesic (or curvilinear)

distances between datapoints. Geodesic distance is the distance between two points measured over the manifold.
In Isomap [79], the geodesic distances between the datapoints x_i are computed by constructing a neighborhood graph G, in which every datapoint x_i is connected with its k nearest neighbors x_{i_j} in the dataset X. The shortest path between two points in the graph forms a good (over)estimate of the geodesic distance between these two points, and can easily be computed using Dijkstra's or Floyd's shortest-path algorithm [23, 29]. The geodesic distances between all datapoints in X are computed, thereby forming a pairwise geodesic distance matrix. The low-dimensional representations y_i of the datapoints x_i in the low-dimensional space Y are computed by applying multidimensional scaling (see subsection 3.2.1) on the resulting distance matrix.
An important weakness of the Isomap algorithm is its topological instability [6]. Isomap may construct erroneous connections in the neighborhood graph G. Such short-circuiting [55] can severely impair the performance of Isomap. Several approaches were proposed to overcome the problem of short-circuiting, e.g., by removing datapoints with large total flows in the shortest path-algorithm [18] or by removing nearest neighbors that violate local linearity of the neighborhood graph [71]. Furthermore, Isomap may suffer from holes in the manifold, a problem that can be dealt with by tearing manifolds with holes [55]. A third weakness of Isomap is that it can fail if the manifold is nonconvex [80]. Despite these weaknesses, Isomap was successfully applied on tasks such as wood inspection [64], visualization of biomedical data [57], and head pose estimation [68].

3.2.4 FastMVU

Similar to Isomap, Fast Maximum Variance Unfolding (FastMVU) defines a neighborhood graph on the data and retains pairwise distances in the resulting graph. FastMVU differs from Isomap in that explicitly attempts to 'unfold' data manifolds. It does so by maximizing the Euclidean distances between the datapoints, under the constraint that the distances in the neighborhood graph are left unchanged (i.e., under the constraint that the local geometry of the data manifold is not distorted). The resulting optimization problem can be solved efficiently using semidefinite programming (which is why FastMVU is also known as Semidefinite Embedding or SDE).
FastMVU starts with the construction of a neighborhood graph G, in which each datapoint x_i is connected to its k nearest neighbors x_{i_j}. Subsequently, FastMVU attempts to maximize the sum of the squared Euclidean distances between all datapoints, under the constraint that the distances inside the neighborhood graph G are preserved. In other words, FastMVU performs the following optimization problem

$$\begin{aligned}&\text{Maximize} \sum_{ij} \| y_i - y_j \|^2 \text{ subject to:}\\ &\quad 1)\ \| y_i - y_j \|^2 = \| x_i - x_j \|^2 \text{ for all } (i,j) \in G\end{aligned}$$

FastMVU reformulates the optimization problem as a semidefinite programming problem (SDP) [87] by defining a matrix K that is the inner product of the low-dimensional data representation Y. It can be shown that the above optimization is similar to the SDP

$$\begin{aligned}&\text{Maximize } \mathrm{trace}(K) \text{ subject to:}\\ &\quad 1)\ k_{ii} - 2k_{ij} + k_{jj} = \| x_i - x_j \|^2 \text{ for all } (i,j)\\ &\quad 2)\ \sum_{ij} k_{ij} = 0\\ &\quad 3)\ K \geq 0\end{aligned}$$

From the solution K of the SDP, the low-dimensional data representation Y can be obtained by computing $Y = K^{1/2}$.

3.2.5 Kernel PCA

Kernel PCA (KPCA) is the reformulation of traditional linear PCA in a high-dimensional space that is constructed using a kernel function [72]. In recent years, the reformulation of linear techniques using the 'kernel trick' has led to the proposal of successful techniques such as kernel ridge regression and Support Vector Machines [74]. Kernel PCA

computes the principal eigenvectors of the kernel matrix, rather than those of the covariance matrix. The reformulation of traditional PCA in kernel space is straightforward, since a kernel matrix is similar to the inproduct of the datapoints in the high-dimensional space that is constructed using the kernel function. The application of PCA in kernel space provides Kernel PCA the property of constructing nonlinear mappings.
Kernel PCA computes the kernel matrix K of the datapoints x_i. The entries in the kernel matrix are defined by

$$k_{ij} = \kappa(x_i, x_j) \tag{11}$$

where κ is a kernel function [74]. Subsequently, the kernel matrix K is centered using the following modification of the entries

$$k_{ij} = k_{ij} - \frac{1}{n}\sum_l k_{il} - \frac{1}{n}\sum_l k_{jl} + \frac{1}{n^2}\sum_{lm} k_{lm} \tag{12}$$

The centering operation corresponds to subtracting the mean of the features in traditional PCA. It makes sure that the features in the high-dimensional space defined by the kernel function are zero-mean. Subsequently, the principal d eigenvectors v_i of the centered kernel matrix are computed. It can be shown that the eigenvectors of the covariance matrix α_i (in the high-dimensional space constructed by κ) are scaled versions of the eigenvectors of the kernel matrix v_i

$$\alpha_i = \frac{1}{\sqrt{\lambda_i}} v_i \tag{13}$$

In order to obtain the low-dimensional data representation, the data is projected onto the eigenvectors of the covariance matrix. The result of the projection (i.e., the low-dimensional data representation Y) is given by

$$Y = \left\{ \sum_j \alpha_1 \kappa(x_j, x), \sum_j \alpha_2 \kappa(x_j, x), ..., \sum_j \alpha_d \kappa(x_j, x) \right\} \tag{14}$$

where κ is the kernel function that was also used in the computation of the kernel matrix. Since Kernel PCA is a kernel-based method, the mapping performed by Kernel PCA highly relies on the choice of the kernel function κ. Possible choices for the kernel function include the linear kernel (making Kernel PCA equal to traditional PCA), the polynomial kernel, and the Gaussian kernel [74].
Kernel PCA has been successfully applied to, e.g., face recognition [49], speech recognition [58], and novelty detection [41]. An important drawback of Kernel PCA is that the size of the kernel matrix is square with the number of instances in the dataset. An approach to resolve this drawback is proposed in [82].

3.2.6 GDA

In the same way as Kernel PCA, Generalized Discriminant Analysis (or Kernel LDA) is the reformulation of LDA in the high-dimensional space constructed using a kernel function [7]. Similar to LDA, GDA attempts to maximize the Fisher criterion. The difference between GDA and LDA is that GDA maximizes the Fisher criterion in the high-dimensional space that is constructed using the kernel function. In this way, GDA allows for the construction of nonlinear mappings that maximize the class separability in the data.
GDA starts by performing Kernel PCA on the data in order to remove noise from the data. Hence, GDA computes the kernel matrix K of the datapoints x_i using Equation 11. The resulting kernel matrix K is centered by modifying its entries using Equation 12. The centering equation is the kernel equivalent of subtracting the mean of the features (as is done in traditional LDA). Subsequently, an eigenanalysis of the centered kernel matrix K is performed. The n eigenvectors of the kernel matrix K are stored in a matrix P that satisfies

$$K = P\Lambda P^T \tag{15}$$

In general, many eigenvectors p_i correspond to small eigenvalues λ_i. In order to remove noise from the data, the eigenvectors p_i that correspond to an eigenvalue $\lambda_i < \epsilon$ (for a small value of ϵ) are removed from matrix P. The smoothed kernel K can be recomputed using Equation 15. Subsequently, GDA maximizes the Fisher criterion (see

Equation 4) in the high-dimensional space defined by the kernel function κ. It can be shown[4] that this is equivalent to maximizing the Rayleigh quotient

$$\phi(W) = \frac{\alpha^T KWK\alpha}{\alpha^T KK\alpha} \tag{16}$$

where W is an $n \times n$ blockdiagonal matrix

$$W = (W_c)_{c=1,2,\ldots,l} \tag{17}$$

In the equation, W_c is an $n_c \times n_c$ matrix of which the entries are $\frac{1}{n_c}$ (where n_c indicates the number of instances of class c) and l indicates the number of classes. Hence, the solution that maximizes the between-class scatter and minimizes the within-class scatter is given by the eigenvectors corresponding to the principal eigenvalues of KWK. In fact, the smoothed kernel K does not have to be computed explicitly, since the eigenanalysis of KWK is equal to the eigenanalysis of $P^T WP$. Hence, GDA computes the d principal eigenvectors of the matrix $P^T WP$ and stores these in a matrix V. The eigenvectors v_i are normalized by computing

$$\alpha_i = \frac{v_i}{\sqrt{v_i^T K v_i}} \tag{18}$$

In order to obtain the low-dimensional representation of the data, the data is projected onto the normalized eigenvectors α_i in the high-dimensional space defined by the kernel function κ. Hence, the low-dimensional representation of the data is given by Equation 14.

3.2.7 Diffusion maps

The diffusion maps (DM) framework [54, 61] originates from the field of dynamical systems. Diffusion maps are based on defining a Markov random walk on the graph of the data. By performing the random walk for a number of timesteps, a measure for the proximity of the datapoints is obtained. Using this measure, the so-called diffusion distance is defined. In the low-dimensional representation of the data, the pairwise diffusion distances are retained as well as possible.
In the diffusion maps framework, a graph of the data is constructed first. The weights of the edges in the graph are computed using the Gaussian kernel function, leading to a matrix W with entries

$$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \tag{19}$$

where σ indicates the variance of the Gaussian. Subsequently, normalization of the matrix W is performed in such a way that its rows add up to 1. In this way, a matrix $P^{(1)}$ is formed with entries

$$p_{ij}^{(1)} = \frac{w_{ij}}{\sum_k w_{ik}} \tag{20}$$

Since diffusion maps originate from dynamical systems theory, the resulting matrix $P^{(1)}$ is considered a Markov matrix that defines the forward transition probability matrix of a dynamical process. Hence, the matrix $P^{(1)}$ represents the probability of a transition from one datapoint to another datapoint in a single timestep. The forward probability matrix for t timesteps $P^{(t)}$ is given by $(P^{(1)})^t$. Using the random walk forward probabilities $p_{ij}^{(t)}$, the diffusion distance is defined by

$$D^{(t)}(x_i, x_j) = \sum_k \frac{(p_{ik}^{(t)} - p_{jk}^{(t)})^2}{\psi(x_k)^{(0)}} \tag{21}$$

In the equation, $\psi(x_i)^{(0)}$ is a term that attributes more weight to parts of the graph with high density. It is defined by $\psi(x_i)^{(0)} = \frac{m_i}{\sum_j m_j}$, where m_i is the degree of node x_i defined by $m_i = \sum_j p_{ij}$. From Equation 21, it can be observed that pairs of datapoints with a high forward transition probability have a small diffusion distance. The key idea behind the diffusion distance is that it is based on many paths through the graph. This makes the diffusion distance more

[4]For the proof, we refer to [7].

robust to noise than, e.g., the geodesic distance. In the low-dimensional representation of the data Y, diffusion maps attempt to retain the diffusion distances. Using spectral theory on the random walk, it can be shown[5] that the low-dimensional representation Y that retains the diffusion distances is formed by the d nontrivial principal eigenvectors of the eigenproblem

$$P^{(t)}Y = \lambda Y \tag{22}$$

Because the graph is fully connected, the largest eigenvalue is trivial (viz. $\lambda_1 = 1$), and its eigenvector v_1 is thus discarded. The low-dimensional representation Y is given by the next d principal eigenvectors. In the low-dimensional representation, the eigenvectors are normalized by their corresponding eigenvalues. Hence, the low-dimensional data representation is given by

$$Y = \{\lambda_2 v_2, \lambda_3 v_3, ..., \lambda_{d+1} v_{d+1}\} \tag{23}$$

3.2.8 SNE

Stochastic Neighbor Embedding (SNE) is an iterative technique that (similar to MDS) attempts to retain the pairwise distances between the datapoints in the low-dimensional representation of the data [39]. SNE differs from MDS by the distance measure that is used, and by the cost function that it minimizes. In SNE, similarities of nearby points contribute more to the cost function. This leads to a low-dimensional data representation that preserves mainly local properties of the manifold. The locality of its behavior strongly depends on the user-defined value for the free parameter σ.

In SNE, the probability p_{ij} that datapoints x_i and x_j are generated by the same Gaussian is computed for all combinations of datapoints and stored in a matrix P. The probabilities p_{ij} are computed using the Gaussian kernel function (see Equation 19). Subsequently, the coordinates of the low-dimensional representations y_i are set to random values (close to zero). The probabilities q_{ij} that datapoints y_i and y_j are generated by the same Gaussian are computed and stored in a matrix Q using the Gaussian kernel as well. In a perfect low-dimensional representation of the data, P and Q are equal. Hence, SNE minimizes the difference between the probability distributions P and Q. Kullback-Leibler divergences are a natural distance measure to measure the difference between two probability distributions. SNE attempts to minimize the following sum of Kullback-Leibler divergences

$$\phi(Y) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{24}$$

The minimization of $\phi(Y)$ can be performed in various ways. In the original algorithm [39], minimization of $\phi(Y)$ is performed using a gradient descent method. To avoid problems with local minima, an amount of Gaussian jitter (that is reduced over time) is added in every iteration. An alternative approach to SNE is based on a trust-region algorithm [62].

3.2.9 Multilayer autoencoders

Multilayer encoders are feed-forward neural networks with an odd number of hidden layers [21, 40]. The middle hidden layer has d nodes, and the input and the output layer have D nodes. An example of an autoencoder is shown schematically in Figure 1. The network is trained to minimize the mean squared error between the input and the output of the network (ideally, the input and the output are equal). Training the neural network on the datapoints x_i leads to a network in which the middle hidden layer gives a d-dimensional representation of the datapoints that preserves as much information in X as possible. The low-dimensional representations y_i can be obtained by extracting the node values in the middle hidden layer, when datapoint x_i is used as input. If linear activation functions are used in the neural network, an autoencoder is very similar to PCA [53]. In order to allow the autoencoder to learn a nonlinear mapping between the high-dimensional and low-dimensional data representation, sigmoid activation functions are generally used.

Multilayer autoencoders usually have a high number of connections. Therefore, backpropagation approaches converge slowly and are likely to get stuck in local minima. In [40], this drawback is overcome by performing a pretraining

[5]See [54] for the derivation.

using Restricted Boltzmann Machines (RBMs) [38]. RBMs are neural networks of which the units are binary and stochastic, and in which connections between hidden units are not allowed. RBMs can successfully be used to train neural networks with many hidden layers using a learning approach based on simulated annealing [50]. Once the pretraining using RBMs is performed, finetuning of the network weights is performed using backpropagation. As an alternative approach, genetic algorithms [42, 67] can be used to train an autoencoder.

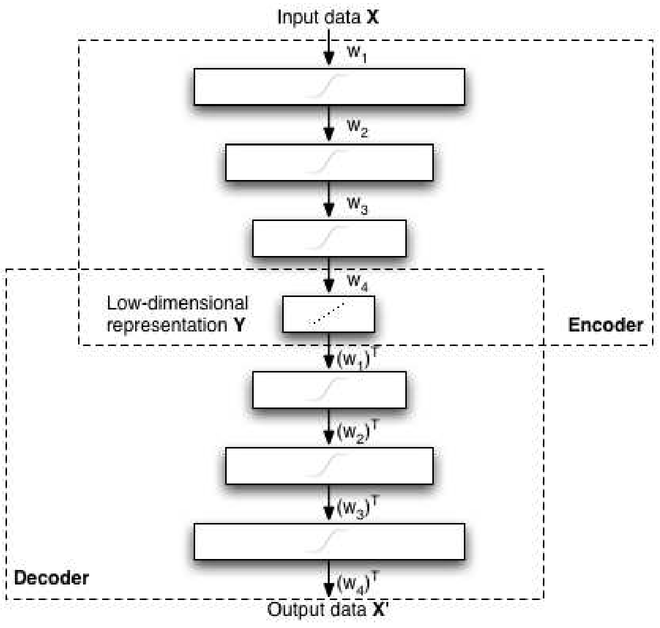

Figure 1: Schematic structure of an autoencoder.

3.3 Local nonlinear techniques

Subsection 3.2 presented eight techniques for dimensionality reduction that attempt to retain global properties of the data. In contrast, local nonlinear techniques for dimensionality reduction are based on solely preserving properties of small neighborhoods around the datapoints. This subsection presents four local nonlinear techniques for dimensionality reduction: (1) LLE, (2) Laplacian Eigenmaps, (3) Hessian LLE, and (4) LTSA in subsubsection 3.3.1 to 3.3.4. The key idea behind these techniques is that by preservation of local properties of the data, the global layout of the data manifold is retained as well.
It can be shown that local techniques for dimensionality reduction can be viewed upon as definitions of specific local kernel functions for Kernel PCA [10, 35]. Therefore, these techniques can be rewritten within the Kernel PCA framework. In this paper, we do not elaborate further on the theoretical connection between local methods for dimensionality reduction and Kernel PCA.

3.3.1 LLE

Local Linear Embedding (LLE) [70] is a local technique for dimensionality reduction that is similar to Isomap in that it constructs a graph representation of the datapoints. In contrast to Isomap, it attempts to preserve solely local properties of the data, making LLE less sensitive to short-circuiting than Isomap. Furthermore, the preservation of local properties allows for successful embedding of nonconvex manifolds. In LLE, the local properties of the data manifold are constructed by writing the datapoints as a linear combination of their nearest neighbors. In the low-dimensional representation of the data, LLE attempts to retain the reconstruction weights in the linear combinations as well as possible.

LLE describes the local properties of the manifold around a datapoint x_i by writing the datapoint as a linear combination W_i (the so-called reconstruction weights) of its k nearest neighbors x_{i_j}. Hence, LLE fits a hyperplane through the datapoint x_i and its nearest neighbors, thereby assuming that the manifold is locally linear. The local linearity assumption implies that the reconstruction weights W_i of the datapoints x_i are invariant to translation, rotation, and rescaling. Because of the invariance to these transformations, any linear mapping of the hyperplane to a space of lower dimensionality preserves the reconstruction weights in the space of lower dimensionality. In other words, if the low-dimensional data representation preserves the local geometry of the manifold, the reconstruction weights W_i that reconstruct datapoint x_i from its neighbors in the high-dimensional data representation also reconstruct datapoint y_i from its neighbors in the low-dimensional data representation. As a consequence, finding the d-dimensional data representation Y amounts to minimizing the cost function

$$\phi(Y) = \sum_i (y_i - \sum_{j=1}^{k} w_{ij} y_{i_j})^2 \tag{25}$$

It can be shown[6] that the coordinates of the low-dimensional representations y_i that minimize this cost function can be found by computing the eigenvectors corresponding to the smallest d nonzero eigenvalues of the inproduct of $(I - W)$. In this formula, I is the $n \times n$ identity matrix.
LLE has been successfully applied for, e.g., superresolution [16] and sound source localization [26]. However, there also exist experimental studies that report weak performance of LLE. In [57], LLE was reported to fail in the visualization of even simple synthetic biomedical datasets. In [45], it is claimed that LLE performs worse than Isomap in the derivation of perceptual-motor actions. A possible explanation lies in the difficulties that LLE has when confronted with manifolds that contains holes [70].

3.3.2 Laplacian Eigenmaps

Similar to LLE, Laplacian Eigenmaps find a low-dimensional data representation by preserving local properties of the manifold [8]. In Laplacian Eigenmaps, the local properties are based on the pairwise distances between near neighbors. Laplacian Eigenmaps compute a low-dimensional representation of the data in which the distances between a datapoint and its k nearest neighbors are minimized. This is done in a weighted manner, i.e., the distance in the low-dimensional data representation between a datapoint and its first nearest neighbor contributes more to the cost function than the distance between the datapoint and its second nearest neighbor. Using spectral graph theory, the minimization of the cost function is defined as an eigenproblem.
The Laplacian Eigenmap algorithm first constructs a neighborhood graph G in which every datapoint x_i is connected to its k nearest neighbors. For all points x_i and x_j in graph G that are connected by an edge, the weight of the edge is computed using the Gaussian kernel function (see Equation 19), leading to a sparse adjacency matrix W. In the computation of the low-dimensional representations y_i, the cost function that is minimized is given by

$$\phi(Y) = \sum_{ij} (y_i - y_j)^2 w_{ij} \tag{26}$$

In the cost function, large weights w_{ij} correspond to small distances between the datapoints x_i and x_j. Hence, the difference between their low-dimensional representations y_i and y_j highly contributes to the cost function. As a consequence, nearby points in the high-dimensional space are brought closer together in the low-dimensional representation. The computation of the degree matrix M and the graph Laplacian L of the graph W allows for formulating the minimization problem as an eigenproblem [4]. The degree matrix M of W is a diagonal matrix, whose entries are the row sums of W (i.e., $m_{ii} = \sum_j w_{ij}$). The graph Laplacian L is computed by $L = M - W$. It can be shown that the following holds[7]

$$\phi(Y) = \sum_{ij} (y_i - y_j)^2 w_{ij} = 2Y^T L Y \tag{27}$$

[6]$\phi(Y) = (Y - WY)^2 = Y^T(I - W)^T(I - W)Y$ is the function that has to be minimized. Hence, the eigenvectors of $(I - W)^T(I - W)$ corresponding to the smallest nonzero eigenvalues form the solution that minimizes $\phi(Y)$.

[7]Note that $\phi(Y) = \sum_{ij}(y_i - y_j)^2 w_{ij} = \sum_{ij}(y_i^2 + y_j^2 - 2y_i y_j)w_{ij} = \sum_i y_i^2 m_{ii} + \sum_j y_j^2 m_{jj} - 2\sum_{ij} y_i y_j w_{ij} = 2Y^T MY - 2Y^T WY = 2Y^T LY$

Hence, minimizing $\phi(Y)$ is proportional to minimizing $Y^T LY$. The low-dimensional data representation Y can thus be found by solving the generalized eigenvector problem

$$Lv = \lambda Mv \tag{28}$$

for the d smallest nonzero eigenvalues. The d eigenvectors v_i corresponding to the smallest nonzero eigenvalues form the low-dimensional data representation Y.
Laplacian Eigenmaps have been successfully applied to, e.g., clustering [63, 75, 91] and face recognition [37].

3.3.3 Hessian LLE

Hessian LLE (HLLE) [24] is a variant of LLE that minimizes the 'curviness' of the high-dimensional manifold when embedding it into a low-dimensional space, under the constraint that the low-dimensional data representation is locally isometric. This is done by an eigenanalysis of a matrix $\mathcal{H}$ that describes the curviness of the manifold around the datapoints. The curviness of the manifold is measured by means of the local Hessian at every datapoint. The local Hessian is represented in the local tangent space at the datapoint, in order to obtain a representation of the local Hessian that is invariant to differences in the positions of the datapoints. It can be shown[8] that the coordinates of the low-dimensional representation can be found by performing an eigenanalysis of $\mathcal{H}$.
Hessian LLE starts with identifying the k nearest neighbors for each datapoint x_i using Euclidean distance. In the neighborhood, local linearity of the manifold is assumed. Hence, a basis for the local tangent space at point x_i can be found by applying PCA on its k nearest neighbors x_{i_j}. In other words, for every datapoint x_i, a basis for the local tangent space at point x_i is determined by computing the d principal eigenvectors $M = \{m_1, m_2, ..., m_d\}$ of the covariance matrix $\mathrm{cov}_{x_{i_j} - \bar{x}_{i_j}}$. Note that the above requires that $k \geq d$. Subsequently, an estimator for the Hessian of the manifold at point x_i in local tangent space coordinates is computed. In order to do this, a matrix Z_i is formed that contains (in the columns) all cross products of M up to the dth order (including a column with ones). The matrix Z_i is orthonormalized by applying Gram-Schmidt orthonormalization [1] on the matrix Z_i. The estimation of the tangent Hessian H_i is now given by the transpose of the last $\frac{d(d+1)}{2}$ columns of the matrix Z_i. Using the Hessian estimators in local tangent coordinates, a matrix $\mathcal{H}$ is constructed with entries

$$\mathcal{H}_{lm} = \sum_i \sum_j ((H_i)_{jl} \times (H_i)_{jm}) \tag{29}$$

The matrix $\mathcal{H}$ represents information on the curviness of the high-dimensional data manifold. An eigenanalysis of $\mathcal{H}$ is performed in order to find the low-dimensional data representation that minimizes the curviness of the manifold. The eigenvectors corresponding to the d smallest nonzero eigenvalues of $\mathcal{H}$ are selected and form the matrix Y, which contains the low-dimensional representation of the data.

3.3.4 LTSA

Similar to Hessian LLE, Local Tangent Space Analysis (LTSA) is a technique that describes local properties of the high-dimensional data using the local tangent space of each datapoint [93]. LTSA is based on the observation that, if local linearity of the manifold is assumed, there exists a linear mapping from a high-dimensional datapoint to its local tangent space, and that there exists a linear mapping from the corresponding low-dimensional datapoint to the same local tangent space [93]. LTSA attempts to align these linear mappings in such a way, that they construct the local tangent space of the manifold from the low-dimensional representation. In other words, LTSA simultaneously searches for the coordinates of the low-dimensional data representations, and for the linear mappings of the low-dimensional datapoints to the local tangent space of the high-dimensional data.
Similar to Hessian LLE, LTSA starts with computing bases for the local tangent spaces at the datapoints x_i. This is done by applying PCA on the k datapoints x_{i_j} that are neighbors of datapoint x_i. This results in a mapping M_i from the neighborhood of x_i to the local tangent space Θ_i. A property of the local tangent space Θ_i is that there exists a

[8]The derivation is too extensive for this paper, but can be found in [24].

linear mapping L_i from the local tangent space coordinates θ_{i_j} to the low-dimensional representations y_{i_j}. Using this property of the local tangent space, LTSA performs the following minimization

$$\min_{Y_i, L_i} \sum_i \| Y_i J_k - L_i \Theta_i \|^2 \tag{30}$$

where J_k is the centering matrix of size k [74]. It can be shown[9] that the solution of the minimization is formed by the eigenvectors of an alignment matrix B, that correspond to the d smallest nonzero eigenvalues of B. The entries of the alignment matrix B are obtained by iterative summation (for all matrices V_i and starting from $b_{ij} = 0$ for $\forall ij$)

$$B_{N_i N_i} = B_{N_i N_i} + J_k (I - V_i V_i^T) J_k \tag{31}$$

where N_i is a selection matrix that contains the indices of the nearest neighbors of datapoint x_i. Subsequently, the low-dimensional representation Y is obtained by computation of the eigenvectors corresponding to the d smallest nonzero eigenvectors of the symmetric matrix $\frac{1}{2}(B + B^T)$.
In [81], a successful application of LTSA to microarray data is reported.

3.4 Extensions and variants of local nonlinear techniques

In subsection 3.3, we discussed four local nonlinear techniques for dimensionality reduction. The capability of these techniques to successfully identify complex data manifolds has led to the proposal of several extensions and variants. In this subsection, we discuss two extensions of local nonlinear techniques for dimensionality reduction, as well as three linear approximations to local nonlinear techniques. The two extensions are discussed in subsubsection 3.4.1 and 3.4.2. Subsection 3.4.3 to 3.4.5 discuss the three linear approximations to local nonlinear dimensionality reduction techniques.

3.4.1 Conformal Eigenmaps

Conformal Eigenmaps (CCA) are based on the observation that local nonlinear techniques for dimensionality reduction do not employ information on the geometry of the data manifold that is contained in discarded eigenvectors that correspond to relatively small eigenvalues [73]. Conformal Eigenmaps initially perform LLE (or alternatively, another local nonlinear technique for dimensionality reduction) to reduce the high-dimensional data to a dataset of dimensionality d_t, where $d < d_t < D$. Conformal Eigenmaps use the resulting intermediate solution in order to construct a d-dimensional embedding that is maximally angle-preserving (i.e., conformal).
A conformal mapping is a transformation that is preserves the angles between neighboring datapoints when reducing the dimensionality of the data. Consider a triangle (x_h, x_i, x_j) in the high-dimensional data representation X and its low-dimensional counterpart (y_h, y_i, y_j). The mapping between these two triangles is conformal iff

$$\frac{\| y_h - y_i \|^2}{\| x_h - x_i \|^2} = \frac{\| y_i - y_j \|^2}{\| x_i - x_j \|^2} = \frac{\| y_h - y_j \|^2}{\| x_h - x_j \|^2} \tag{32}$$

Based on this observation, Conformal Eigenmaps define a measure $\mathcal{C}_h$ that evaluates the conformality of the triangles at datapoint x_h in the neighborhood graph by

$$\mathcal{C}_h = \sum_{ij} \eta_{hi} \eta_{ij} (\| y_i - y_j \|^2 - s_h \| x_i - x_j \|^2)^2 \tag{33}$$

where η_{ij} is a variable that is 1 if x_i and x_j are connected in the neighborhood graph, and s_h is a variable that corrects for scalings in the transformation from the high-dimensional data representation X to its low-dimensional counterpart Y. Summing over $\mathcal{C}_h$ yields a measure $\mathcal{C}(Y)$ that measures the conformality of a low-dimensional data representation Y with respect to the original data X. Conformal Eigenmaps seek for a linear mapping M from the d_t-dimensional data representation Z obtained from LLE to the low-dimensional data representation Y that maximizes

[9]The proof is too extensive for this paper, but can be found in [93].

the conformality measure $\mathcal{C}(Y)$. The maximization of the conformality measure $\mathcal{C}(Y)$ can be performed by solving the following semidefinite program[10]

$$\begin{aligned}
&\text{Maximize } t \text{ subject to:}\\
&\quad 1)\ P \geq 0\\
&\quad 2)\ \operatorname{trace} P = 1\\
&\quad 3)\ \begin{pmatrix} I & SP \\ (SP)^T & t \end{pmatrix} \geq 0
\end{aligned}$$

where I represents the identity matrix, S is a matrix containing the scaling coefficients s_h, and P represents a matrix that is given by $P = M^T M$. Using the solution P of the SDP, the linear transformation M can be obtained by computing $M = P^{1/2}$. The low-dimensional data representation Y is obtained by mapping the intermediate solution Z (that was contructed by LLE) onto M, i.e., by computing $Y = ZM$.

3.4.2 MVU

Maximum Variance Unfolding (MVU) is very similar to FastMVU in that both techniques solve the same optimization problem. However, in contrast to FastMVU, MVU starts from an intermediate solution that is obtained from LLE (similar to Conformal Eigenmaps). The idea behind MVU is that local nonlinear techniques for dimensionality reduction aim to preserve local properties of the data, but not necessarily aim to completely 'unfold' a data manifold. MVU performs the unfolding based on the low-dimensional data representation computed by LLE, thereby improving the low-dimensional representation.
MVU starts with performing LLE (although another local nonlinear method could be used as well) in order to reduce the dimensionality of the data to d_t, where $d < d_t < D$. Subsequently, the FastMVU procedure is applied on the obtained data representation Z of dimensionality d_t in order to obtain the final low-dimensional representation Y. The intuition behind this approach is that local nonlinear techniques for dimensionality reduction preserve local properties of a data manifold, but not explicitly unfold the data manifold. The advantage of MVU over FastMVU is that the semidefinite program that MVU solves is much smaller than the program that is solved by FastMVU.

3.4.3 LPP

In contrast to traditional linear techniques such as PCA, local nonlinear techniques for dimensionality reduction (see subsection 3.3) are capable of the successful identification of complex data manifolds such as the Swiss roll. This capability is due to the cost functions that are minimized by local nonlinear dimensionality reduction techniques, which aim at preserving local properties of the data manifold. However, in many learning settings, the use of a linear technique for dimensionality reduction is desired, e.g., when an accurate and fast out-of-sample extension is necessary, when data has to be transformed back into its original space, or when one wants to visualize the transformation that was constructed by the dimensionality reduction technique. Linearity Preserving Projection (LPP) is a technique that aims at combining the benefits of linear techniques and local nonlinear techniques for dimensionality reduction by finding a linear mapping that minimizes the cost function of Laplacian Eigenmaps [36].
Similar to Laplacian Eigenmaps, LPP starts with the construction of a nearest neighbor graph in which every datapoint x_i is connected to its k nearest neighbors x_{i_j}. The weights of the edges in the graph are computed using Equation 19. Subsequently, LPP solves the generalized eigenproblem

$$(X - \bar{X})^T L (X - \bar{X}) v = \lambda (X - \bar{X})^T M (X - \bar{X}) v \tag{34}$$

in which L is the graph Laplacian, and M the degree matrix of the graph, as described in subsubsection 3.3.2. It can be shown that the eigenvectors v_i corresponding to the d smallest nonzero eigenvalues form the columns of the linear mapping T that minimizes the Laplacian Eigenmap cost function (see Equation 26). The low-dimensional data representation Y is thus given by $Y = (X - \bar{X})T$.

[10]For the derivation of the SDP, we refer to [73].

3.4.4 NPE

Similar to LPP, Neigborhood Preserving Embedding (NPE) minimizes the cost function of a local nonlinear technique for dimensionality reduction under the constraint that the mapping from the high-dimensional to the low-dimensional data representation is linear. NPE is the linear approximation to LLE (see subsubsection 3.3.1).
NPE defines a neighborhood graph on the dataset X, and subsequently computes the reconstruction weights W_i as in LLE. The cost function of LLE (see Equation 25) is optimized by solving the following generalized eigenproblem for the d smallest nonzero eigenvalues

$$(X - \bar{X})^T (I - W)^T (I - W)(X - \bar{X})v = \lambda (X - \bar{X})^T (X - \bar{X})v \tag{35}$$

where I represents the $n \times n$ identity matrix. The low-dimensional data representation is computed by mapping X onto the obtained mapping T, i.e., by computing $Y = (X - \bar{X})T$.

3.4.5 LLTSA

Using a similar approach as LPP and NPE, LLTSA is a linear technique for dimensionality reduction that minimizes the cost function of LTSA. Summarizing, LLTSA defines a neighborhood graph on the data and estimates the local tangent space Θ_i at every datapoint x_i. Subsequently, it forms the alignment matrix B by performing the summation in Equation 31. The LTSA cost function (see Equation 30) is minimized in a linear manner by solving the generalized eigenproblem

$$(X - \bar{X})^T B (X - \bar{X})v = \lambda (X - \bar{X})^T (X - \bar{X})v \tag{36}$$

for the d smallest nonzero eigenvalues. The d eigenvectors v_i form the columns of the linear mapping T, which can be used to obtain the low-dimensional representation Y by computing $Y = (X - \bar{X})T$.

3.5 Global alignment of linear models

In the previous subsections, we discussed techniques that compute a low-dimensional data representation by preserving global or local properties of the data. Techniques that perform global alignment of (local) linear models combine these two types: they compute a collection of linear models and perform an alignment of these linear models. In this subsection, three techniques are discussed that globally align a collection of linear models: (1) LLC, (2) manifold charting, and (3) CFA. The techniques are discussed separately in subsubsection 3.5.1 to 3.5.3.

3.5.1 LLC

Locally Linear Coordination (LLC) [78] computes a number of locally linear models and subsequently performs a global alignment of the linear models. This process consists of two steps: (1) computing a mixture of local linear models on the data by means of an Expectation Maximization (EM) algorithm and (2) aligning the local linear models in order to obtain the low-dimensional data representation by finding a linear mapping from the data models that minimizes the LLE cost function.
LLC first constructs a mixture of m factor analyzers using the EM-algorithm [22, 33, 47]. Alternatively, a mixture of probabilistic PCA models could be used [82]. The mixture of linear models outputs m data representations z_{ij} and corresponding responsibilities r_{ij} (where $j \in \{1, ..., m\}$) for every datapoint x_i. The responsibility r_{ij} describes to what extent datapoint x_i corresponds to the linear model z_{ij}, and satisfies $\sum_j r_{ij} = 1$. Using the linear models and the corresponding responsibilities, responsibility-weighted data representations $u_{ij} = r_{ij}[z_{ij}\ 1]$ are computed. The 1 is added to the data representations z_{ij} in order to allow translations of the data representations z_{ij} in the linear mapping L that is computed later. The responsibility-weighted data representations u_{ij} are stored in a $n \times mD$ block matrix U. The alignment of the local models is performed based on U and on a matrix M that is given by $M = (I - W)^T (I - W)$. Herein, the matrix W contains the reconstruction weights computed by LLE (see subsubsection 3.3.1), and I denotes the $n \times n$ identity matrix. LLC aligns the local models by solving the generalized eigenproblem

$$Av = \lambda Bv \tag{37}$$

for the d smallest nonzero eigenvalues[11]. In the equation, A is the inproduct of $M^T U$ and B is the inproduct of U. The d eigenvectors v_i form a matrix L, that can be shown to define a linear mapping from the responsibility-weighted data representation U to the underlying low-dimensional data representation Y that minimizes the LLE cost function. The low-dimensional data representation is thus obtained by computing $Y = UL$.

3.5.2 Manifold charting

Similar to LLC, manifold charting constructs a low-dimensional data representation by aligning a mixture of factor analyzers (MFA) model [13]. In constrast to LLC, manifold charting does not minimize a cost function that corresponds to another dimensionality reduction technique (such as the LLE cost function). Manifold charting minimizes a convex cost function that measures the amount of disagreement between the linear models on the global coordinates of the datapoints. The minimization of this cost function can be performed by solving a generalized eigenproblem.
Manifold charting first performs the EM algorithm to learn a mixture of factor analyzers, in order to obtain m low-dimensional data representations z_{ij} and corresponding responsibilities r_{ij} (where $j \in \{1, ..., m\}$) for all datapoints x_i. Manifold charting finds a linear mapping from the data representations z_{ij} to the global coordinates y_i that minimizes the cost function

$$\phi(Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \parallel y_i - y_{ik} \parallel^2 \tag{38}$$

where $y_i = \sum_{k=1}^{m} r_{ik} y_{ik}$. The intuition behind the cost function is that whenever there are two linear models to which a datapoint has a high responsibility, these linear models should agree on the final coordinate of the datapoint. The cost function can be rewritten in the form

$$\phi(Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} r_{ij} r_{ik} \parallel y_{ij} - y_{ik} \parallel^2 \tag{39}$$

which allows the cost function to be rewritten in the form of a Rayleigh quotient. The Rayleight quotient can be constructed by the definition of a block-diagonal matrix D with m blocks by

$$D = \begin{pmatrix} D_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & D_m \end{pmatrix} \tag{40}$$

where D_j is the sum of the weighted covariances of the data representations z_{ij}. Hence, D_j is given by

$$D_j = \sum_{i=1}^{n} r_{ij} \underset{[z_{ij} 1]}{\text{cov}} \tag{41}$$

In the equation, the 1 is added to the linear models z_{ij} in order to facilitate translations in the construction of y_i from the data representations z_{ij}. Using the definition of the matrix D and the $n \times mD$ block-diagonal matrix U with entries $u_{ij} = r_{ij}[z_{ij}\ 1]$, the manifold charting cost function can be rewritten as

$$\phi(Y) = L^T (D - U^T U) L \tag{42}$$

where L represents the linear mapping on the matrix Z that can be used to compute the final low-dimensional data representation Y. The linear mapping L can thus be computed by solving the generalized eigenproblem

$$(D - U^T U) v = \lambda U^T U v \tag{43}$$

for the d smallest nonzero eigenvalues. The d eigenvectors v_i form the columns of the linear combination L from $[Z\ 1]$ to Y.

[11]The derivation of this eigenproblem can be found in [78].

3.5.3 CFA

Similar to LLC and manifold charting, Coordinated Factor Analysis (CFA) constructs a low-dimensional data representation by globally aligning a mixture of factor analyzers [89]. However, CFA combines the construction of the mixture model and the alignment into one stage, whereas LLC and manifold charting first construct the mixture model, and subsequently align the separate linear models. The main advantage of the approach taken by CFA is that it allows for changes in the linear models if in order to facilitate a better global alignment of the linear models. CFA is implemented by means of an EM algorithm that maximizes the normal MFA log-likelihood function, minus a term that measures to what extent the mixture $p(y|x_i)$ resembles a Gaussian. The penalty term takes the form of a Kullback-Leibler divergence.
The penalized log-likelihood function that is maximized in CFA is given by

$$\mathcal{L} = \sum_{i=1}^{n} \left(\log p(x_i) - \mathcal{D}(q_i(y) \parallel p(y|x_i)) \right) \tag{44}$$

where $\log p(x_i)$ represents the log-likelihood of the MFA model, $\mathcal{D}(\cdot \parallel \cdot)$ represents a Kullback-Leibler divergence, and q_i represents a Gaussian $q_i = \mathcal{N}(y; y_i, \Sigma_i)$. The penalty term measures to what extent the mixture $p(y|x_i)$ resembles a Gaussian. In other words, the penalty term reflects the notion of agreement in the learning algorithm on the low-dimensional data representation y. The EM algorithm that maximizes the penalized log-likelihood function in Equation 44 is tedious, but all update rules are given by closed form equations. The update rules that implement the EM algorithm can be found in [89]. Although CFA allows for changes in the MFA model in case this is beneficial for the alignment of the linear model, its performance may suffer from the presence of many local maxima in the penalized log-likelihood function.

4 Techniques for intrinsic dimensionality estimation

In section 3, we presented 23 techniques for dimensionality reduction. In the entire section, we assumed that the target dimensionality of the low-dimensional data representation was specified by the user. Ideally, the target dimensionality is set equal to the intrinsic dimensionality of the dataset. Intrinsic dimensionality is the minimum number of parameters that is necessary in order to account for all information in the data. Several techniques have been proposed in order to estimate the intrinsic dimensionality of a dataset X. Techniques for intrinsic dimensionality estimation can be used in order to circumvent the problem of selecting a proper target dimensionality (e.g., using hold-out testing) to reduce the dataset X to. In this section, we discuss six techniques for intrinsic dimensionality estimation. For convenience, we denote the intrinsic dimensionality of the dataset X by d and its estimation by $\hat{d}$ (not to be confused with the notation d for target dimensionality that was employed in the previous section).
Techniques for intrinsic dimensionality estimation can be subdivided into two main groups: (1) estimators based on the analysis of local properties of the data and (2) estimators based on the analysis of global properties of the data. In this section, we present six techniques for intrinsic dimensionality estimation. Three techniques based on the analysis of local data properties are discussed in subsection 4.1. Subsection 4.2 presents three techniques for intrinsic dimensionality estimation that are based on the analysis of global properties of the data.

4.1 Local estimators

Local intrinsic dimensionality estimators are based on the observation that the number of datapoints covered by a hypersphere around a datapoint with radius r grows proportional to r^d, where d is the intrinsic dimensionality of the data manifold around that datapoint. As a result, the intrinsic dimensionality d can be estimated by measuring the number of datapoints covered by a hypersphere with a growing radius r.
This subsection describes three local estimators for intrinsic dimensionality: (1) the correlation dimension estimator, (2) the nearest neighbor dimension estimator, and (3) the maximum likelihood estimator. The estimators are discussed separately in subsubsection 4.1.1 to 4.1.3.

4.1.1 Correlation dimension estimator

The correlation dimension estimator uses the the intuition that the number of datapoints in a hypersphere with radius r is proportional to r^d by computing the relative amount of datapoints that lie within a hypersphere with radius r. The relative amount of datapoints that lie within a hypersphere with radius r is given by

$$C(r) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} c \quad \text{where } c = \begin{cases} 1, & \text{if } \parallel x_i - x_j \parallel \leq r \\ 0, & \text{if } \parallel x_i - x_j \parallel > r \end{cases} \tag{45}$$

Because the value $C(r)$ is proportional to r^d, we can use $C(r)$ to estimate the intrinsic dimensionality d of the data. The intrinsic dimensionality d is given by the limit

$$d = \lim_{r \to 0} \frac{\log C(r)}{\log r} \tag{46}$$

Since the limit in the above equation cannot be solved for explicitly, its value is estimated by computing $C(r)$ for two values of r. The estimation of the intrinsic dimensionality of the data is then given by the ratio

$$\hat{d} = \frac{\log(C(r_2) - C(r_1))}{\log(r_2 - r_1)} \tag{47}$$

4.1.2 Nearest neighbor estimator

Similar to the correlation dimension estimator, the nearest neighbor estimator is based on the the evaluation of the number of neighboring datapoints that are covered by a hypersphere with radius r. In contrast to the correlation dimension estimator, the nearest neighbor estimator does not explicitly count the number of datapoints inside a hypersphere with radius r, but it computes the minimum radius r of the hypersphere that is necessary to cover k nearest neighbors. Mathematically, the nearest neighbor estimator computes

$$C(k) = \frac{1}{n} \sum_i T_k(x_i) \tag{48}$$

where $T_k(x_i)$ represents the radius of the smallest hypersphere with center x_i that covers k neighboring datapoints. Similar to the correlation dimension estimator, the nearest neighbor estimator estimates the intrinsic dimensionality by

$$\hat{d} = \frac{\log(C(k_2) - C(k_1))}{\log(k_2 - k_1)} \tag{49}$$

4.1.3 Maximum likelihood estimator

Similar to the correlation dimension and the nearest neighbor dimension estimator, the maximum likelihood estimator for intrinsic dimensionality [56] estimates the number of datapoints covered by a hypersphere with a growing radius r. In contrast to the former two techniques, the maximum likelihood estimator does so by modelling the number of datapoints inside the hypersphere as a Poisson process. In the Poisson process, the rate of the process $\lambda(t)$ at intrinsic dimensionality d is expressed as

$$\lambda(t) = \frac{f(x)\pi^{d/2} d t^{d-1}}{\Gamma(d/2 + 1)} \tag{50}$$

in which $f(x)$ is the sampling density and $\Gamma(\cdot)$ is the gamma function. Based on the Poisson process, it can be shown that the maximum likelihood estimation of the intrinsic dimensionality d around datapoint x_i given k nearest neighbors is given by[12]

$$\hat{d}_k(x_i) = \left(\frac{1}{k-1} \sum_{j=1}^{k-1} \log \frac{T_k(x_i)}{T_j(x_i)} \right)^{-1} \tag{51}$$

in which $T_k(x_i)$ represents the radius of the smallest hypersphere with center x_i that covers k neighboring datapoints. In the original paper [56], the estimation of the intrinsic dimensionality of the entire dataset X is obtained by averaging over the n local estimates $\hat{d}_k(x_i)$. However, there exist arguments indicating that a maximum likelihood estimator for intrinsic dimensionality should not average over all estimations, but over their inverses [59].

4.2 Global estimators

Whereas local estimators for intrinsic dimensionality average over local estimates of intrinsic dimensionality, global estimators consider the data as a whole when estimating the intrinsic dimensionality. This subsection describes three global intrinsic dimensionality estimators: (1) the eigenvalue-based estimator, (2) the packing number estimator, and (3) the geodesic minimum spanning tree estimator. The three global intrinsic dimensionality estimators are descibed separately in subsubsection 4.2.1 to 4.2.3.

4.2.1 Eigenvalue-based estimator

The eigenvalue-based intrinsic dimensionality estimator performs PCA (see subsubsection 3.1.1) on the high-dimensional dataset X and evaluates the eigenvalues corresponding to the principal components[13] [32]. The values of the eigenvalues provide insight in the amount of variance that is described by their corresponding eigenvectors[14]. After normalization, the eigenvalues can be plotted in order to provide insight in the intrinsic dimensionality of the data. In

[12]The derivation of the maximum likelihood estimator can be found in [56].

[13]Please note that the eigenvalues obtained from Kernel PCA can be evaluated in exactly the same way.

[14]This follows directly from the theory of the Rayleigh quotient. A function of the form $\frac{X^T A X}{X^T X}$ is maximized by the eigenvector v_1 of A that corresponds to the largest eigenvector λ_1 of A. The value of the function $\frac{X^T A X}{X^T X}$ at the maximum given by λ_1 itself. Note that PCA maximizes a Rayleigh quotient where $A = \text{cov}_{X-\bar{X}}$.

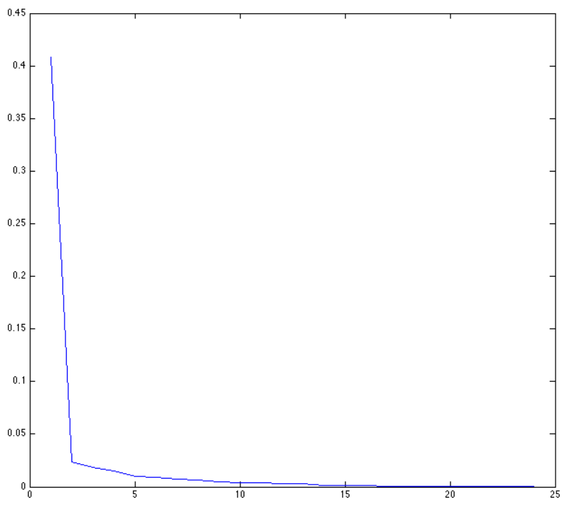

Figure 2: Example of an eigenvalue plot.

an eigenvalue plot, the first d eigenvalues are typically high (where d is the intrinsic dimensionality of the dataset), whereas the remaining eigenvalues are typically small (since they only model noise in the data). An example of a typical eigenvalue plot is shown in Figure 2. The estimation of the intrinsic dimensionality d is performed by counting the number of normalized eigenvalues that is higher than a small threshold value ϵ.

4.2.2 Packing numbers estimator

The packing numbers intrinsic dimensionality estimator [48] is based on the intuition that the r-covering number $N(r)$ is proportional to r^{-d}. The r-covering number $N(r)$ is the number of hyperspheres with radius r that is necessary to cover all datapoints x_i in the dataset X. Because $N(r)$ is proportional to r^{-d}, the intrinsic dimensionality of a dataset X is given by

$$d = -\lim_{r \to 0} \frac{\log N(r)}{\log r} \tag{52}$$

In general, finding the r-covering number $N(r)$ of a dataset X is a problem that is computationally infeasible. The packing numbers estimator circumvents this problem by using the r-packing number $M(r)$ instead of the r-covering number $N(r)$. The r-packing number $M(r)$ is defined as the maximum size of an r-separated subset of X. In other words, the r-packing number $M(r)$ is the maximum number of datapoints in X that can be covered by a single hypersphere with radius r. For datasets of reasonable size, finding the r-packing number $M(r)$ is computationally feasible. The intrinsic dimensionality of a dataset X can be found by evaluating the limit

$$d = -\lim_{r \to 0} \frac{\log M(r)}{\log r} \tag{53}$$

Because the limit cannot be evaluated explicitly, the intrinsic dimensionality of the data is estimated (similar to the correlation dimension estimator) by

$$\hat{d} = -\frac{\log(M(r_2) - M(r_1))}{\log(r_2 - r_1)} \tag{54}$$

4.2.3 GMST estimator

The geodesic minimum spanning tree (GMST) estimator is based on the observation that the length function of a geodesic minimum spanning tree is strongly dependent on the intrinsic dimensionality d. The GMST is the minimum spanning tree of the neighborhood graph defined on the dataset X. The length function of the GMST is the sum of the Euclidean distances corresponding to all edges in the geodesic minimum spanning tree. The intuition behind the dependency between the length function of the GMST and the intrinsic dimensionality d is similar to the intuition behind local intrinsic dimensionality estimators.

Similar to Isomap, the GMST estimator constructs a neighborhood graph G on the dataset X, in which every datapoint x_i is connected with its k nearest neighbors x_{i_j}. The geodesic minimum spanning tree T is defined as the minimal graph over X, which has length

$$L(X) = \min_{T \in \mathcal{T}} \sum_{e \in T} g_e \tag{55}$$

where $\mathcal{T}$ is the set of all subtrees of graph G, e is an edge in tree T, and g_e is the Euclidean distance corresponding to the edge e. In the GMST estimator, a number of subsets $A \subset X$ of the dataset X are constructed with various sizes m, and the lengths $L(A)$ of the GMSTs of the subsets A are computed. Theoretically, the ratio $\frac{\log L(A)}{\log m}$ is linear, and can thus be estimated by a function of the form $y = ax + b$. The variables a and b are estimated by means of least squares. It can be shown that the estimated value of a provides an estimate for the intrinsic dimensionality through $\hat{d} = \frac{1}{1-a}$.

5 Examples

In the previous sections, we described 23 techniques for dimensionality reduction and 6 techniques for intrinsic dimensionality estimation. All these techniques are implemented in the Matlab Toolbox for Dimensionality Reduction. In this section, we explain how the implementations in the toolbox can be used. Throughout this section, we assume the dataset is contained in an $n \times D$ matrix X. If class labels are available for the datapoints in X, these are contained in an $n \times 1$ numeric vector labels.
If you encounter any errors while excuting the example code in this section, please check whether your system meets the prerequisites for using the Matlab Toolbox for Dimensionality Reduction, and whether you installed the toolbox properly. A list of prerequisites can be found in appendix A. Installation instructions are to be found in appendix B.
In subsection 5.1, examples of the use of intrinsic dimensionality estimators are provided. Subsection 5.2 describes examples of performing dimensionality reduction on a dataset X. In subsection 5.3, we discuss additional features of the toolbox.

5.1 Intrinsic dimensionality estimation

In the Matlab Toolbox for Dimensionality Reduction, the 6 intrinsic dimensionality estimators described in section 4 are available through the function intrinsic_dim. Consider the following example:

```
[X, labels] = generate_data('swiss', 2000, 0.05);
scatter3(X(:,1), X(:,2), X(:,3), 5, labels); drawnow
d = intrinsic_dim(X, 'MLE');
disp(['Intrinsic dimensionality of data: ' num2str(d)]);
```

The code above constructs a Swiss roll dataset of 2,000 datapoints with a small amount of noise around the manifold. The dataset is plotted in a 3D scatter plot. Subsequently, the intrinsic dimensionality of the constructed dataset is estimated by means of the maximum likelihood estimator. The result of the code above should be similar to:

```
Intrinsic dimensionality of data: 1.978
```

Note that the true intrinsic dimensionality of the Swiss roll dataset is 2, and the maximum likelihood estimation of the intrinsic dimensionality is thus fairly accurate.
As can be observed from the example above, the syntax of the intrinsic_dim function is

```
d = intrinsic_dim(X, method)
```

where method can have the following values:

- 'CorrDim'
- 'NearNbDim'
- 'MLE'
- 'EigValue'
- 'PackingNumbers'
- 'GMST'

The default value for method is 'MLE'. Below, we present an example reveals differences between the estimators for intrinsic dimensionality:

```
[X, labels] = generate_data('twinpeaks', 2000, 0.05);
scatter3(X(:,1), X(:,2), X(:,3), 5, labels); drawnow
d1 = intrinsic_dim(X);
disp(['MLE estimation: ' num2str(d1)]);
d2 = intrinsic_dim(X, 'CorrDim');
disp(['Correlation dim. estimation: ' num2str(d2)]);
d3 = intrinsic_dim(X, 'NearNbDim');
disp(['NN dim. estimation: ' num2str(d3)]);
d4 = intrinsic_dim(X, 'EigValue');
disp(['Eigenvalue estimation: ' num2str(d4)]);
```

```
d5 = intrinsic_dim(X, 'PackingNumbers');
disp(['Packing numbers estimation: ' num2str(d5)]);
d6 = intrinsic_dim(X, 'GMST');
disp(['GMST estimation: ' num2str(d6)]);
mu = mean([d1 d2 d3 d4 d5 d6]);
disp(['Mean estimation: ' num2str(mu)]);
```

The code above will generate a twin peaks manifold of 2,000 datapoints with some noise around the manifold, and run the six intrinsic dimensionality estimators on the resulting dataset. The result of the code above should be similar to:

```
MLE estimation: 2.1846
Correlation dim. estimation: 2.093
NN dim. estimation: 3.2165
Eigenvalue estimation: 2
Packing numbers estimation: 1.5292
GMST estimation: 2.098
Mean estimation: 2.1869
```

From the example above, we observe that there are differences in the estimations made by various intrinsic dimensionality estimators. For real-world datasets, these differences may be larger than for the artificial dataset we used in the example.

5.2 Dimensionality reduction

In the previous subsection, we provided examples of intrinsic dimensionality estimation using the Matlab Toolbox for Dimensionality Reduction. Based on the intrinsic dimensionality estimations (or based on knowledge on the underlying distribution of the data), it is possible to select a proper value for the target dimensionality d. Once the target dimensionality d is chosen, we can select a technique in order to reduce the dimensionality of the dataset `X` to d dimensions by means of the `compute_mapping` function. Consider the following example:

```
[X, labels] = generate_data('twinpeaks', 2000, 0.05);
scatter3(X(:,1), X(:,2), X(:,3), 5, labels); drawnow
d = round(intrinsic_dim(X));
Y = compute_mapping(X, 'PCA', d);
figure, scatter(Y(:,1), Y(:,2), 5, labels), drawnow
```

The code above generates a twinpeaks manifold of 2,000 datapoints, estimates its intrinsic dimensionality d (which is somewhere near 2, see subsection 5.1), and reduces the dataset to d dimensions by means of Principal Components Analysis. The resulting data representation is shown in Figure 3.
The syntax of the `compute_mapping` function is similar for all dimensionality reduction techniques, although the number of parameters that can be specified differs per technique. The syntax of the `compute_mapping` function is

```
[Y, M] = compute_mapping(X, method, d, pars, eig)
```

where `method` indicates the technique that is used, `d` represents the target dimensionality d, `pars` represents one or more parameters, and `eig` indicates the type of iterative eigensolver that is used in spectral techniques that perform an eigenanalysis of a sparse matrix. The `compute_mapping` returns the low-dimensional data representation in `Y`. Additional information on the computed mapping is contained in the `M` structure. The `compute_mapping` supports the following values for `method`:

- `'PCA'` or `'SimplePCA'` or `'ProbPCA'`
- `'LDA'`
- `'MDS'`
- `'SPE'`
- `'Isomap'` or `'LandmarkIsomap'`
- `'FastMVU'`
- `'KernelPCA'`

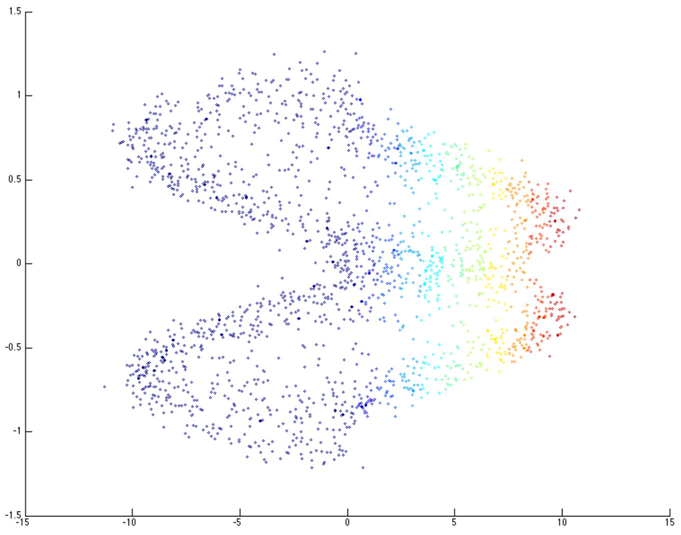

Figure 3: PCA applied on the twinpeaks dataset.

- `'GDA'`
- `'DiffusionMaps'`
- `'AutoencoderRBM'` or `'AutoencoderEA'`
- `'LLE'`
- `'Laplacian'`
- `'HessianLLE'`
- `'LTSA'`
- `'Conformal'`
- `'MVU'`
- `'LPP'`
- `'NPE'`
- `'LLTSA'`
- `'LLC'`
- `'ManifoldChart'`
- `'CFA'`

The default value for `method` is `'PCA'`. The default value for the target dimensionality `d` is 2. Note that the dimensionality of `Y` is not necessarily equal to `d`. For various techniques, the rank of a matrix that is spectrally analyzed may be lower than `d`, leading to a data representation `Y` of lower dimensionality than the original setting of `d`. If this occurs, the toolbox will always show a warning. In addition, running the Isomap algorithm might reduce the number of datapoints in the low-dimensional data representation Y, because Isomap only embeds the largest connected component in the neighborhood graph. The indices of the datapoints that are retained in the low-dimensional data representation `Y` can be obtained from `M.conn_comp`.

Some of the techniques in the toolbox are very closely related, which is why we briefly discuss their differences. The `'PCA'` setting performs the traditional PCA procedure (as described in subsubsection 3.1.1), whereas the `'SimplePCA'` applies a Hebbian learning technique [65] to estimate the eigenvectors of the covariance matrix. The `'ProbPCA'` performs probabilistic PCA by means of an EM algorithm [83]. The setting `'LandmarkIsomap'` selects a number of landmark points from the dataset X, performs Isomap on these datapoints, and fits the remaining points into the

Technique	*Parameters*
PCA	none
SPCA	none
PPCA	$i = 200$
LDA	none
MDS	none
SPE	$type = \{$['Global'], 'Local'$\}$
Isomap	$k = 12$
Landmark Isomap	$k = 12$
FastMVU	$k = 12$
Kernel PCA	$\kappa(\cdot, \cdot)$
GDA	$\kappa(\cdot, \cdot)$
Diffusion maps	$\sigma = 1 \quad t = 1$
SNE	$\sigma = 1$
Autoencoders	none
LLE	$k = 12$
Laplacian Eigenmaps	$k = 12 \quad \sigma = 1$
Hessian LLE	$k = 12$
LTSA	$k = 12$
Conformal Eigenmaps	$k = 12$
MVU	$k = 12$
LPP	$k = 12$
NPE	$k = 12 \quad \sigma = 1$
LLTSA	$k = 12$
LLC	$k = 12 \quad m = 20 \quad i = 200$
Manifold charting	$m = 40 \quad i = 200$
CFA	$m = 40 \quad i = 200$

Table 1: Parameters and their default values.

low-dimensional data representation [20]. In contrast, the setting `'Isomap'` performs the original Isomap algorithm (as described in subsubsection 3.2.3), which might be computationally infeasible for large problems. The setting `'AutoencoderRBM'` trains an autoencoder by means of a pretraining with an RBM and backpropagation (see [40] for details), whereas the setting `'AutoencoderEA'` trains an autoencoder by means of an evolutionary algorithm.
A large number of techniques for dimensionality reduction has one or more free parameters, which can be set when applying the technique. A list of parameters and their default values is provided in Table 1. The parameters should be provided in the same order as listed in the table. If there are two parameters that need to be set, it is possible to set only the first one. However, if the user wants to specify the second parameter, the first parameter needs to be specified as well. In all techniques, the parameter `k` may have a integer value representing the number of neighbors to use in the neighborhood graph, but it may also be set to `'adaptive'`. The `'adaptive'` setting constructs a neighborhood graph using the technique for adaptive neighborhood selection described in [60].
For the kernel methods Kernel PCA and GDA, the kernel function $\kappa(\cdot, \cdot)$ can be defined through the `compute_mapping` function. An overview of the possible kernel functions is provided in Table 2. For datasets up to 3,000 datapoints, Kernel PCA explicitly computes the kernel matrix K and stores it in to memory. Because of memory constraints, for larger problems, the eigendecomposition of the kernel matrix is performed using an iterative eigensolver that does not explicitly store the kernel matrix in memory. As a result, it is not possible to use your own kernel matrices in the Kernel PCA implementation.

The last parameter of the `compute_mapping` function is the `eig` parameter. The `eig` parameter applies only to techniques for dimensionality reduction that perform a spectral analysis of a sparse matrix. Possible values are `'Matlab'` and `'JDQR'` (the default value is `'Matlab'`). The `'Matlab'` setting performs the spectral analysis using Arnoldi methods [5], whereas the setting `'JDQR'` employs Jacobi-Davidsson methods [30] for the spectral analysis. For large datasets (where $n > 10,000$), we advice the use of Jacobi-Davidsson methods for spectral analysis of sparse matrices. The parameter `eig` should be the last parameter that is given to the `compute_mapping` function.

Kernel	*Parameters*	*Function*
Linear	'linear'	$k_{ij} = x_i^T x_j$
Polynomial	'poly', a, b	$k_{ij} = (x_i^T x_j + a)^b$
Gaussian	'gauss', σ	$k_{ij} = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$

Table 2: Kernel functions in the toolbox.

Below, we provide a second example of the use of the toolbox:

```
[X, labels] = generate_data('swiss', 2000, 0.05);
d = round(intrinsic_dim(X));
Y1 = compute_mapping(X, 'LLE');
Y2 = compute_mapping(X, 'LLE', d, 7);
Y3 = compute_mapping(X, 'Laplacian', d, 7, 'JDQR');
Y4 = compute_mapping(X, 'LTSA', d, 7);
Y5 = compute_mapping(X, 'CCA', d, 'Matlab');
subplot(3, 2, 1), scatter3(X(:,1), X(:,2), X(:,3), 5, labels);
subplot(3, 2, 2), scatter(Y1(:,1), Y1(:,2), 5, labels);
subplot(3, 2, 3), scatter(Y2(:,1), Y2(:,2), 5, labels);
subplot(3, 2, 4), scatter(Y3(:,1), Y3(:,2), 5, labels);
subplot(3, 2, 5), scatter(Y4(:,1), Y4(:,2), 5, labels);
subplot(3, 2, 6), scatter(Y5(:,1), Y5(:,2), 5, labels);
```

The code above generates a Swiss roll dataset of 2,000 datapoints with some noise around the data manifold, estimates the intrinsic dimensionality of the resulting dataset, performs five techniques for dimensionality reduction on the dataset using various settings, and plots the results. The resulting plot should look like the plot in Figure 4.

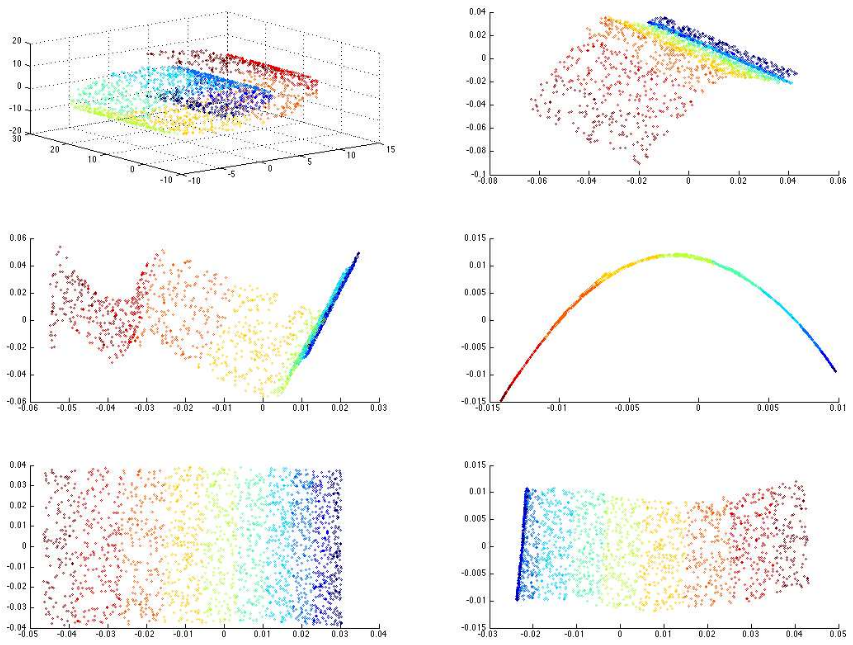

Figure 4: Dimensionality reduction on the Swiss roll dataset.

5.3 Additional functions

In addition to the functions for intrinsic dimensionality estimation and dimensionality reduction, the toolbox contains functions for data generation, data prewhitening, and out-of-sample extension. The generation of a number of artificial dataset can be performed using the `generate_data` function, which is discussed in subsubsection 5.3.1. For data prewhitening, the function `prewhiten` can be used (see subsubsection 5.3.2). Out-of-sample extension can be performed by means of the `out_of_sample` and the `out_of_sample_est` functions. The two functions for out-of-sample extension are described in more detail in subsubsection 5.3.3.

5.3.1 Data generation

The `generate_data` is a function that can be used to generate five different artificial datasets: (1) the Swiss roll dataset, (2) the twinpeaks dataset, (3) the helix dataset, (4) the 3D clusters dataset, and (5) the intersecting dataset. All artificial datasets have dimensionality 3, but their intrinsic dimensionalities vary from 1 to 3. The syntax of the `generate_data` function is

```
[X, labels] = generate_data(name, n, noise)
```

Herein, `name` is the name of the dataset. The possible values are `'swiss'`, `'twinpeaks'`, `'helix'`, `'3d_clusters'`, and `'intersect'`. The default value is `'swiss'`. The parameter `n` specifies the number of datapoints that is sampled from the underlying manifold (the default value is 1,000). The parameter `noise` determines the amount of noise around the data manifold (the default value is 0.05).
Below, we provide an example of the use of the `generate_data` function:

```
[X, labels] = generate_data('helix', 2000, 0.05);
subplot(1, 3, 1), scatter3(X(:,1), X(:,2), X(:,3), 5, labels);
[X, labels] = generate_data('intersect', 2000, 0.5);
subplot(1, 3, 2), scatter3(X(:,1), X(:,2), X(:,3), 5, labels);
[X, labels] = generate_data('3d_clusters');
subplot(1, 3, 3), scatter3(X(:,1), X(:,2), X(:,3), 5, labels);
```

The plot resulting from the code above should look like Figure 5.

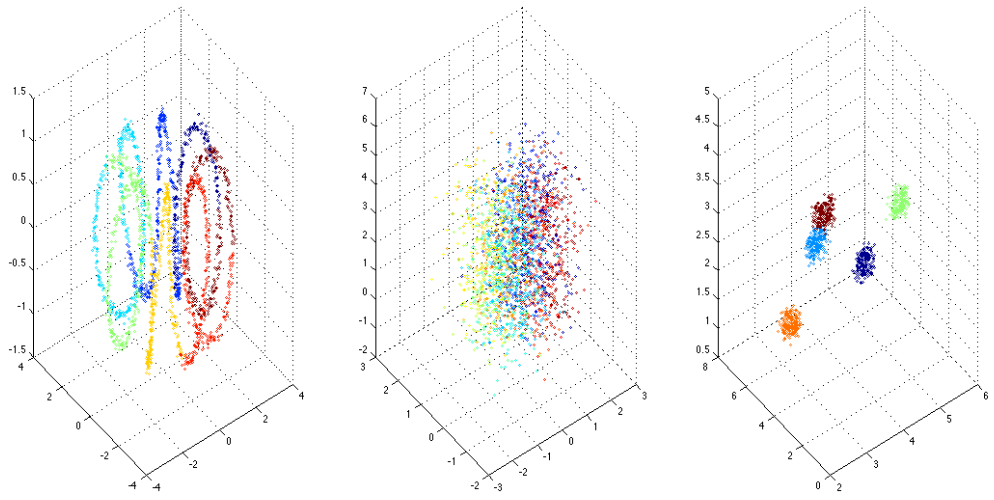

Figure 5: Examples of artificial datasets.

5.3.2 Prewhitening

In section 2, we argued that data lies on a manifold of low dimensionality that is embedded in a high-dimensional space. In practice, the data is likely to contain noise, i.e., the data lies near a manifold of low dimensionality. The performance of techniques for dimensionality reduction can often be improved by the removal of noise around the data manifold. Prewhitening is a straightforward way to remove noise from the high-dimensional data. Prewhitening performs PCA (see subsection 3.1.1) on the high-dimensional dataset, thereby retaining a large portion of the variance in the data (typically around 95% of the variance in the data is retained).
Using the Matlab Toolbox for Dimensionality Reduction, prewhitening of data can be performed by means of the `prewhiten` function. The syntax of the `prewhiten` function is

```
X = prewhiten(X)
```

The function is parameterless. Because of the simplicity of the `prewhiten` function, we dot not present example code in which the function is used.

5.3.3 Out-of-sample extension

Until now, we have assumed that dimensionality reduction is applied in learning settings in which both training and test data are available. In, e.g., classification systems, test data often becomes available after the training process is completed. In order to avoid retraining every time new test data comes available, the construction of a method for out-of-sample extension is desirable. Out-of-sample extension transforms new high-dimensional datapoints to the low-dimensional space.
For linear techniques for dimensionality reduction, out-of-sample extension is straightforward, since the linear mapping M defines the transformation from the high-dimensional data representation to its low-dimensional counterpart. For linear dimensionality reduction techniques, out-of-sample extension of a new datapoint x_{n+1} is thus performed by computing

$$y_{n+1} = (x_{n+1} - \bar{X})M \tag{56}$$

For Kernel PCA, a similar procedure can be applied, although the out-of-sample extension for Kernel PCA requires some additional kernel computations. For autoencoders, out-of-sample extension can readily be performed, because the trained neural network defines the transformation from the high-dimensional data representation to its counterpart of lower dimensionality.
For other dimensionality reduction techniques, out-of-sample extension can only be performed using estimation methods [11, 81]. Because of its general applicability, we briefly discuss the out-of-sample extension method in [81]. The method starts by idenitifying the nearest neigbor x_i of the new datapoint x_{n+1}. Subsequently, the affine transformation matrix T that maps the nearest neighbor x_i to its low-dimensional counterpart y_i is computed. The transformation matrix is given by

$$T = (y_i - \overline{y}_i)(x_i - \overline{x}_i)^+ \tag{57}$$

where $(\cdot)^+$ represents the matrix pseudoinverse. Because x_{n+1} is near to x_i, the transformation matrix T can be applied to the new datapoint x_{n+1} as well. The coordinates of the new low-dimensional datapoint y_{n+1} are thus given by

$$y_{n+1} = \overline{y}_i + T(x_{n+1} - \overline{x}_i) \tag{58}$$

The advantage of the method described above is its general applicability. Its main disadvantage is that it suffers from estimation errors, especially when the sampling rate of the data is low with respect to the curvature of the data manifold. Using the Matlab Toolbox for Dimensionality Reduction, out-of-sample extension can be performed by means of the `out_of_sample` and `out_of_sample_est` functions.

The out_of_sample function performs an exact out-of-sample extension on new datapoints, and is therefore only applicable to the techniques 'PCA', 'SimplePCA', 'ProbPCA', 'LDA', 'LPP', 'NPE', 'LLTSA', 'KernelPCA', 'AutoencoderRBM', and 'AutoencoderEA'. The syntax of the out_of_sample function is

```
Ynew = out_of_sample(Xnew, mapping)
```

in which Xnew represents the set of new datapoints and mapping is a structure containing information on the performed dimensionality reduction (and can be obtained from the compute_mapping function). The low-dimensional counterparts of Xnew are returned in Ynew.
Below, we provide an example of the use of the out_of_sample function:

```
[X1, lab1] = generate_data('twinpeaks', 1000, 0.05);
[X2, lab2] = generate_data('twinpeaks', 1000, 0.05);
[Y1, mapping] = compute_mapping(X1, 'PCA');
Y2 = out_of_sample(X2, mapping);
Y = [Y1; Y2];
labels = [lab1; lab2];
scatter(Y(:,1), Y(:,2), 5, labels);
```

The code above constructs two twinpeaks datasets of 1,000 points, applies PCA on the first dataset, and performs exact out-of-sample extension on the second dataset. The resulting plot looks similar to Figure 3.
The out_of_sample_est fuction performs out-of-sample extension of new data by means of the general estimation method that was described above. The syntax of the out_of_sample_est function is

```
Ynew = out_of_sample_est(Xnew, X, Y)
```

in which Xnew represents the set of new datapoints, X is the original dataset, and Y is the low-dimensional counterpart of X that was obtained from the compute_mapping function. The low-dimensional counterpart of Xnew obtained using the out-of-sample extimation procedure described above is returned in Ynew. Below, we provide an example of the use of the out_of_sample_est function:

```
[X1, labels1] = generate_data('helix', 1000, 0.05);
[X2, labels2] = generate_data('helix', 1000, 0.05);
[Y1, mapping] = compute_mapping(X1, 'Laplacian');
Y2 = out_of_sample_est(X2, X1, Y1);
Y = [Y1; Y2];
labels = [labels1; labels2];
scatter(Y(:,1), Y(:,2), 5, labels);
```

The code above constructs two helix datasets of 1,000 points, applies Laplacian Eigenmaps on the first dataset, and performs out-of-sample extension on the second dataset using the estimation method described above. The result of the code above is shown in Figure 6. In the plot, the estimation errors in the out-of-sample extension are clearly visible.

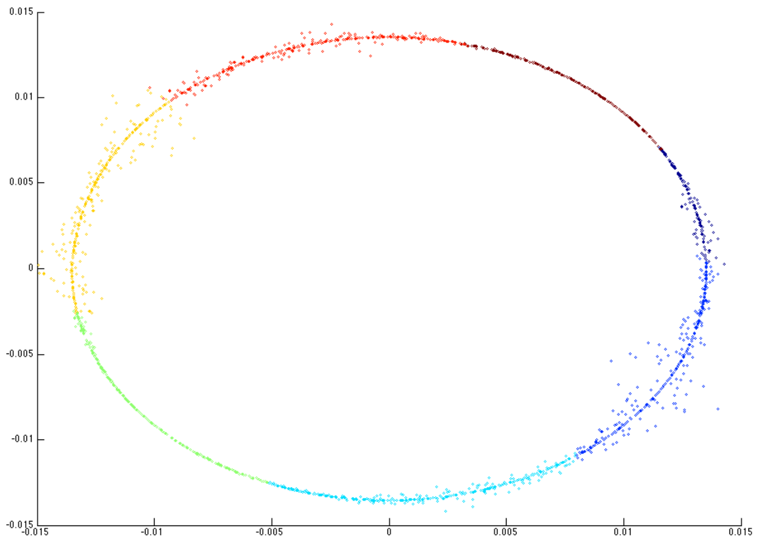

Figure 6: Out-of-sample extension using an estimation method.

6 Conclusion

The report presented a description of the techniques that are implemented in the Matlab Toolbox for Dimensionality Reduction. In addition, we provided a number of code examples that illustrate the functionality of the toolbox. The report did not present discuss properties of the techniques such as computational complexities, neither did it present a theoretical or empirical comparison of the techniques. A study that presents such a comparative review can be found in [86].

Acknowledgements

The work is supported by NWO/CATCH, project RICH, under grant 640.002.401. I would like to thank Eric Postma, Stijn Vanderlooy, and Guido de Croon for their comments on early versions of parts of this report.

References

[1] G. Afken. *Gram-Schmidt Orthogonalization*. Academic Press, Orlando, FL, USA, 1985.

[2] D.K. Agrafiotis. Stochastic proximity embedding. *Journal of Computational Chemistry*, 24(10):1215–1221, 2003.

[3] T.W. Anderson. Asymptotic theory for principal component analysis. *Annals of Mathematical Statistics*, 34:122–148, 1963.

[4] W.N. Anderson and T.D. Morley. Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra*, 18:141–145, 1985.

[5] W.E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–25, 1951.

[6] M. Balasubramanian and E.L. Schwartz. The Isomap algorithm and topological stability. *Science*, 295(5552):7, 2002.

[7] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[8] M. Belkin and P. Niyogi. Laplacian Eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14, pages 585–591, Cambridge, MA, USA, 2002. The MIT Press.

[9] A.J. Bell and T.J. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[10] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and Kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

[11] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, volume 16, Cambridge, MA, USA, 2004. The MIT Press.

[12] C.M. Bishop, M. Svensen, and C. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[13] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems*, volume 15, pages 985–992, Cambridge, MA, USA, 2002. The MIT Press.

[14] M. Brand. From subspaces to submanifolds. In *Proceedings of the* 15^{th} *British Machine Vision Conference*, London, UK, 2004. British Machine Vision Association.

[15] H.-P. Chan. Computer-aided classification of mammographic masses and normal tissue: linear discriminant analysis in texture feature space. *Phys. Med. Biol.*, 40:857–876, 1995.

[16] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 275–282, 2004.

[17] K.-Y. Chang and J. Ghosh. Principal curves for nonlinear feature extraction and classification. In *Applications of Artificial Neural Networks in Image Processing III*, pages 120–129, Bellingham, WA, USA, 1998. SPIE.

[18] H. Choi and S. Choi. Robust kernel Isomap. *Pattern Recognition*, 40(3):853–862, 2007.

[19] T. Cox and M. Cox. *Multidimensional scaling*. Chapman & Hall, London, UK, 1994.

[20] V. de Silva and J.B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 721–728, Cambridge, MA, USA, 2003. The MIT Press.

[21] D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 5, pages 580–587, San Mateo, CA, USA, 1993. Morgan Kaufmann.

[22] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[23] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[24] D.L. Donoho and C. Grimes. Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.

[25] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley Interscience Inc., 2001.

[26] R. Duraiswami and V.C. Raykar. The manifolds of spatial hearing. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 285–288, 2005.

[27] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174, New York, NY, USA, 1995. ACM Press.

[28] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[29] R.W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[30] D.R. Fokkema, G.L.G. Sleijpen, and H.A. van der Vorst. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing*, 20(1):94–125, 1999.

[31] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[32] K. Fukunaga and D.R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, C-20:176–183, 1971.

[33] Z. Ghahramani and G.E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.

[34] R. Haeb-Umbach and H. Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1992*, volume 1, pages 13–16, 1992.

[35] J. Ham, D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. Technical Report TR-110, Max Planck Institute for Biological Cybernetics, Germany, 2003.

[36] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems*, volume 16, page 37, Cambridge, MA, USA, 2004. The MIT Press.

[37] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

[38] G.E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[39] G.E. Hinton and S.T. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840, Cambridge, MA, USA, 2002. The MIT Press.

[40] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[41] H. Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3):863–874, 2007.

[42] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[43] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[44] R. Huber, H. Ramoser, K. Mayer, H. Penz, and M. Rubik. Classification of coins using an eigenspace approach. *Pattern Recognition Letters*, 26(1):61–75, 2005.

[45] O.C. Jenkins and M.J. Mataric. Deriving action and behavior primitives from human motion data. In *International Conference on Intelligent Robots and Systems*, volume 3, pages 2551–2556, 2002.

[46] L.O. Jimenez and D.A. Landgrebe. Supervised classification in high-dimensional space: geometrical,statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man and Cybernetics*, 28(1):39–54, 1997.

[47] N. Kambhatla and T.K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.

[48] B. Kegl. Intrinsic dimension estimation based on packing numbers. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840, Cambridge, MA, USA, 2002. The MIT Press.

[49] K.I. Kim, K. Jung, and H.J. Kim. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, 9(2):40–42, 2002.

[50] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[51] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag, Berlin, Germany, 1988.

[52] J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[53] S.Y. Kung, K.I. Diamantaras, and J.S. Taur. Adaptive Principal component EXtraction (APEX) and applications. *IEEE Transactions on Signal Processing*, 42(5):1202–1217, 1994.

[54] S. Lafon and A.B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

[55] J.A. Lee and M. Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, 67:29–53, 2005.

[56] E. Levina and P.J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems*, volume 17, Cambridge, MA, USA, 2004. The MIT Press.

[57] I.S. Lim, P.H. Ciechomski, S. Sarni, and D. Thalmann. Planar arrangement of high-dimensional biomedical data sets by Isomap coordinates. In *Proceedings of the* 16^{th} *IEEE Symposium on Computer-Based Medical Systems*, pages 50–55, 2003.

[58] A. Lima, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura. On the use of Kernel PCA for feature extraction in speech recognition. *IEICE Transactions on Information Systems*, E87-D(12):2802–2811, 2004.

[59] D.J.C. MacKay and Z. Ghahramani. Comments on 'maximum likelihood estimation of intrinsic dimension'.

[60] N. Mekuz and J.K. Tsotsos. Parameterless Isomap with adaptive neighborhood selection. In *Proceedings of the* 28^{th} *DAGM Symposium*, pages 364–373, Berlin, Germany, 2006. Springer.

[61] B. Nadler, S. Lafon, R.R. Coifman, and I.G. Kevrekidis. Diffusion maps, spectral clustering and the reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis: Special Issue on Diffusion Maps and Wavelets*, 21:113–127, 2006.

[62] K. Nam, H. Je, and S. Choi. Fast Stochastic Neighbor Embedding: A trust-region algorithm. In *Proceedings of the IEEE International Joint Conference on Neural Networks 2004*, volume 1, pages 123–128, Budapest, Hungary, 2004.

[63] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, pages 849–856, Cambridge, MA, USA, 2001. The MIT Press.

[64] M. Niskanen and O. Silvén. Comparison of dimensionality reduction methods for wood surface inspection. In *Proceedings of the* 6^{th} *International Conference on Quality Control by Artificial Vision*, pages 178–188, Gatlinburg, TN, USA, 2003. International Society for Optical Engineering.

[65] M. Partridge and R. Calvo. Fast dimensionality reduction and Simple PCA. *Intelligent Data Analysis*, 2(3):292–298, 1997.

[66] A.M. Posadas, F. Vidal, F. de Miguel, G. Alguacil, J. Pena, J.M. Ibanez, and J. Morales. Spatial-temporal analysis of a seismic series using the principal components method. *Journal of Geophysical Research*, 98(B2):1923–1932, 1993.

[67] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4:164–171, 2000.

[68] B. Raytchev, I. Yoda, and K. Sakaue. Head pose estimation by nonlinear manifold learning. In *Proceedings of the* 17^{th} *ICPR*, pages 462–466, 2004.

[69] S.T. Roweis, L. Saul, and G. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems*, volume 14, pages 889–896, Cambridge, MA, USA, 2001. The MIT Press.

[70] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[71] A. Saxena, A. Gupta, and A. Mukerjee. Non-linear dimensionality reduction by locally linear isomaps. *Lecture Notes in Computer Science*, 3316:1038–1043, 2004.

[72] B. Schölkopf, A.J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[73] F. Sha and L.K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *Proceedings of the* 22^{nd} *International Conference on Machine Learning*, pages 785–792, 2005.

[74] J. Shawe-Taylor and N. Christianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

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

[76] J.A.K. Suykens. Data visualization and dimensionality reduction using kernel maps with a reference point. Technical Report 07-22, ESAT-SISTA, K.U. Leuven, 2007.

[77] G.A. Tagaris, W. Richter, S.G. Kim, G. Pellizzer, P. Andersen, K. Ugurbil, and A.P. Georgopoulos. Functional magnetic resonance imaging of mental rotation and memory scanning: a multidimensional scaling analysis of brain activation patterns. *Brain Res.*, 26(2-3):106–12, 1998.

[78] Y.W. Teh and S.T. Roweis. Automatic alignment of hidden representations. In *Advances in Neural Information Processing Systems*, volume 15, pages 841–848, Cambridge, MA, USA, 2002. The MIT Press.

[79] J.B. Tenenbaum. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems*, volume 10, pages 682–688, Cambridge, MA, USA, 1998. The MIT Press.

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

[81] L. Teng, H. Li, X. Fu, W. Chen, and I.-F. Shen. Dimension reduction of microarray data based on local tangent space alignment. In *Proceedings of the* 4^{th} *IEEE International Conference on Cognitive Informatics*, pages 154–159, 2005.

[82] M.E. Tipping. Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems*, volume 13, pages 633–639, Cambridge, MA, USA, 2000. The MIT Press.

[83] M.E. Tipping and C.M. Bishop. Probabilistic prinicipal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, 1997.

[84] K. Torkkola. Linear discriminant analysis in document classification. In *IEEE ICDM-2001 Workshop on Text Mining*, pages 800–806, 2001.

[85] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Proceedings of the Computer Vision and Pattern Recognition 1991*, pages 586–591, 1991.

[86] L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. Dimensionality reduction: A comparative review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (submitted)*, 2007.

[87] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[88] M.S. Venkatarajan and W. Braun. New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physicalchemical properties. *Journal of Molecular Modeling*, 7(12):445–453, 2004.

[89] J. Verbeek. Learning nonlinear image manifolds by global alignment of local linear models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1236–1250, 2006.

[90] K.Q. Weinberger, B.D. Packer, and L.K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceedings of the* 10^{th} *International Workshop on AI and Statistics*, Barbados, WI, 2005. Society for Artificial Intelligence and Statistics.

[91] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 975–982, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[92] T. Zhang, J. Yang, D. Zhao, and X. Ge. Linear local tangent space alignment and application to face recognition. *Neurocomputing*, 70:1547–1533, 2007.

[93] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26(1):313–338, 2004.

A Prerequisites

The prerequisites for the installation of the Matlab Toolbox for Dimensionality Reduction are:

- A working installation of Matlab 7.0 (R14) or a newer version of Matlab.
- A working installation of the Statistics Toolbox.
- On platforms for which no binaries are provided (e.g., PowerPC Mac and Solaris): a proper setup of `mex`. Type `mex -setup` in Matlab to perform the setup of `mex`.

B Installation instructions

Installation of the Matlab Toolbox for Dimensionality Reduction is fairly straightforward. A step-by-step installation guide is provided below.

- Decompress the file `drtoolbox.tar.gz`. Under Windows, the decompression can be performed using a tool such as WinRar. Under Linux and Mac OS X, the decompression can be performed by opening a new Terminal window, navigating to the location of the `drtoolbox.tar.gz` file, and executing the command `tar -xvf drtoolbox.tar.gz`.
- Decompression of the file `drtoolbox.tar.gz` results in a new directory with the name `drtoolbox`. Although you can put this directory in any location you like, we advice you to move the `drtoolbox` directory to the standard Matlab toolbox directory, which is located in `$MATLAB_ROOT/toolbox`.
- In order to be able to use the Matlab Toolbox for Dimensionality Reduction successfully, Matlab should be notified about its presence. You do this by starting Matlab, and navigating to `File -> Set Path...` Now click the button `Add with Subfolders...` and navigate to the location of the `drtoolbox` directory. Click `OK` and subsequently, save your changes to the Matlab path by clicking the `Save` button. The toolbox is now ready for use.
- The toolbox employs several functions that were implemented in C or C++ in order to speed up the techniques. Precompiled versions for all main platforms are included in the toolbox, but a compiled version for your platform might be either lacking or not have been optimized for your platform. In order to recompile all C and C++ files on your platform, please execute the following code

  ```
  cd([matlabroot '/toolbox/drtoolbox']);
  mexall
  ```

  During the run of the `mexall` function, Matlab might display one or more warnings, which can be ignored. If no errors are displayed during the run of `mexall`, all C and C++ files in the toolbox were successfully compiled.

Technical Reports in Computer Science

The following reports have appeared in this series:

CS 89-01	S.T. Dekker H.J. van den Herik I.S. Herschberg	Complexity Starts at Five (14 pages).
CS 89-02	H.J. van den Herik I.S. Herschberg N. Nakad	A Six-Men-Endgame Database: KRP(a2)KbBP(a3) (18 pages).
CS 89-03	I.S. Herschberg H.J. van den Herik P.N.A. Schoo	Verifying and Codifying Strategies in a Chess Endgame (11 pages).
CS 89-04	J.W.H.M. Uiterwijk H.J. van den Herik L.V. Allis	A Knowledge-Based Approach to Connect-Four The Game is Over: White to Move Wins! (21 pages).
CS 89-05	P.J. Braspenning	De Modellering van Complexe Objecten de route-planning voor INtelligent CAse (11 pages).
CS 90-01	M. van der Meulen	Conspiracy-Number Search (12 pages).
CS 90-02	J. Henseler P.J. Braspenning	Training Complex Multi-Layer Neural Networks (5 pages).
CS 90-03	L.V. Allis M. van der Meulen H.J. van den Herik	$\alpha\beta$ Conspiracy-Number Search (18 pages).
CS 90-04	L.V. Allis H.J. van den Herik I.S. Herschberg	Which Games Will Survive? (9 pages).
CS 90-05	M. van der Meulen L.V. Allis H.J. van den Herik	Lithidion: an Awari-playing Program (23 pages).
CS 90-06	P.J. Braspenning J.W.H.M. Uiterwijk H. Bakker L.C.J. van Leeuwen	The Development of an Object-Oriented Representation System for Complex Objects in Analysis and Design: the INCA-project (61 pages).
CS 91-01	L.V. Allis M. van der Meulen H.J. van den Herik	Proof-Number Search (33 pages).
CS 91-02	J.H.J. Lenting P.J. Braspenning	Planning with Artificial Intelligence (20 pages).
CS 91-03	J.H.J. Lenting P.J. Braspenning	Issues of Representation in AI Planning (31 pages).
CS 91-04	J.H.J. Lenting P.J. Braspenning	Issues of Control in AI Planning (33 pages).
CS 91-05	J. Henseler P.J. Braspenning	Membrain: A Cellular Neural Network based on a Vibrating Membrance (16 pages).
CS 91-06	A. Weijters G. Hoppenbrouwers J. Thole R. van Nieuwenhoven	Spraaksynthese met behulp van Artificiële Neurale Netwerken (41 pages).
CS 91-07	H. Velthuijsen P.J. Braspenning	A Formalized Concept of the Blackboard Architecture (45 pages).
CS 92-01	J. Hage M. van der Meulen	MASC User Interface Guidelines (29 pages).
CS 92-02	M. van der Meulen J. Hage	The HUMF Library (18 pages).
CS 92-03	L.V. Allis H.J. van den Herik	Go-Moku Opgelost door Nieuwe Zoektechnieken (15 pages).

CS 92-04	L.V. Allis	A Knowledge-based Approach of Connect-Four (93 pages). (second edition)
CS 92-05	L.V. Allis H.J. van den Herik M.M. Wind	64. No Threats (14 pages).
CS 92-07	J.H.J. Lenting P.J. Braspenning	The Answer To The Frame Problem: Forty-Two (24 pages).
CS 93-01	D.M. Breuker L.V. Allis H.J. van den Herik	Syllabification Using Expert Rules (16 pages).
CS 93-02	L.V. Allis H.J. van den Herik M.P.H. Huntjens	Go-Moku and Threat-Space Search (11 pages).
CS 93-03	H. Iida J.W.H.M. Uiterwijk H.J. van den Herik	Opponent-Model Search (10 pages).
CS 94-01	G. van Liempd H.J. van den Herik	Predicting the behavior of search algorithms on CSP problems (24 pages).
CS 94-02	P.J. Braspenning A. Lodder J.P. Dekker	Multiple Scattering Theory in a Generalized Fashion A new approach to Artificial Neural Networks (28 pages).
CS 94-03	G.A.W. Vreeswijk	IACAS: an interactive argumentation system (25 pages).
CS 94-04	A.D.M. Wan P.J. Braspenning G.A.W. Vreeswijk	Limits to Ground Control in Autonomous Spacecraft (15 pages).
CS 95-01	G.A.W. Vreeswijk	Open Protocols in Multi-Agent Systems (31 pages).
CS 95-02	G.A.W. Vreeswijk	Formalizing Nomic: working on a theory of communication with modifiable rules of procedure (15 pages).
CS 95-03	G.A.W. Vreeswijk	Self-government in multi-agent systems: experiments and thought-experiments (12 pages).
CS 95-04	G. van Liempd H.J. van den Herik	Predicting the behavior of search algorithms on CSP problems (continued) (33 pages).
CS 95-05	G.A.W. Vreeswijk	Interpolation of Benchmark Problems in Defeasible Reasoning (19 pages).
CS 95-06	G.A.W. Vreeswijk	Several experiments in elementary self-modifying protocol games, such as Nomic (47 pages).
CS 95-07	G.A.W. Vreeswijk	Self-modifying Protocol Games: A Preparatory Case Study (research notes) (40 pages).
CS 95-08	G.A.W. Vreeswijk	Emergent Political Structures in Abstract Forms of Rule-making and Legislation (37 pages).
CS 95-09	G.A.W. Vreeswijk	Representation of Formal Dispute with a Standing Order (22 pages).
CS 95-10	A.J.M.M. Weijters	The BP-SOM Architecture and Learning Rule (11 pages).
CS 95-11	E.O. Postma H.J. van den Herik P.T.W. Hudson	SCAN: A Scalable Model of Attentional Selection (44 pages).
CS 96-01	A.J.M.M. Weijters A.P.J. van den Bosch H.J. van den Herik	Applying Ockham's Razor to Back-propagation (18 pages).
CS 96-02	R.W. van der Pol	A device for query composition (39 pages).
CS 96-03	K.M. Sim P.J. Braspenning	A Framework of Characterizing Multi-agent Systems: Cognition, Communication and Social Theory (12 pages).
CS 96-04	E.N. Smirnov P.J. Braspenning	Disjunctive Version Space Approach to Concept Learning (34 pages).

CS 97-01	A.D.M. Wan P.J. Braspenning	State Identification in Dynamic Goal-State Learning Tasks (6 pages).
CS 97-02	D.M. Breuker H.J. van den Herik L.V. Allis J.W.H.M. Uiterwijk	A Solution to the GHI Problem for Best-First Search (18 pages).
CS 97-03	H.H.L.M. Donkers	Beslissen bij Onzekerheid (95 pages).
CS 97-04	H.H.L.M. Donkers	Markov Decision Networks (M.Sc. Thesis) (89 pages).
CS 97-05	I.D. Craig	VESUVIUS: Modular Reflection and Messages (10 pages).
CS 98-01	P.J. Braspenning K.M. Sim	Characterization and Design of Multi-agent Systems: Theories and Testbeds (19 pages).
CS 98-02	H.H.L.M. Donkers J.W.H.M. Uiterwijk H.J. van den Herik	Introduction to Markov Decision Networks (15 pages).
CS 98-03	H.J. van den Herik P.J. Braspenning I.M.C. Lemmens	Kennistechnologie en Kunstmatige Intelligentie (26 pages)
CS 98-04	H.H.L.M. Donkers J.W.H.M. Uiterwijk H.J. van den Herik	Time Complexity of the Incremental-Pruning Algorithm for Markov Decision Networks (13 pages).
CS 98-05	D.M. Breuker J.W.H.M. Uiterwijk H.J. van den Herik	Solving Domineering (16 pages).
CS 98-06	J.W.H.M. Uiterwijk H.J. van den Herik	The Advantage of the Initiative (12 pages).
CS 98-07	H.H.L.M. Donkers H.J. van den Herik J.W.H.M. Uiterwijk	Techniques for decision making in the LWI-domain (INDEKS-project, phase 3) (14 pages).
CS 99-01	I.M.C Lemmens P.J. Braspenning	A Formal Analysis of Smithsonian Computational Reflection (11 pages).
CS 99-02	I.M.C Lemmens P.J. Braspenning	Computational Reflection: Different Theories compared (23 pages)
CS 99-03	N. Roos	On the relation between information and subjective probability
CS 99-04	D.M. Breuker J.W.H.M Uiterwijk H.J. van den Herik	The pn^2 –search algorithm (16 pages)
CS 99-05	T. Carati G.I. Holle J. Hoonhout L. A. Plugge	Effectiviteitsonderzoek naar de kennisoverdracht van I&E Milieu (80 pages)
CS 99-06	A. Bossi S. Etalle S. Rossi	Properties of Input-Consuming Derivations (18 pages)
CS 99-07	S. Etalle (Ed.)	Benelog99 – Proceedings of the Eleventh Benelux Workshop on Logic Programming
CS 99-08	S. Etalle (Ed.) J.-G. Smaus (Ed.)	Verification of Logic Programs – Proceedings of the 1999 Workshop on Verification of Logic Programs
CS 99-09	A. Lukito A.J. van Zanten	Vertex Partitions of Hypercubes into Snakes (12 pages)
CS 99-10	J. Lenting	On the inadequacy of MAS notions of rationality for multi-agent technology (19 pages)
CS 99-11	J. Lenting	Drawbacks of Pareto optimality, strategy proofness, and incentive compatibility in the context of MAS mechanism design (12 pages)
CS 00-01	A. Bossi, S. Etalle, S. Rossi	Semantics of Input-Consuming Programs (19 pages)
CS 00-02	S. Etalle and J. Mountjoy	The (Lazy) Functional Side of Logic Programming (22 pages)

CS 00-03	J.W.H.M. Uiterwijk	The Fifth Computer Olympiad. Computer-Games Workshop Proceedings
CS 00-04	A.J. van Zanten, A. Lukito	Covers of Hypercubes with Symmetric Snakes (15 pages)
CS 00-05	H.H.L.M. Donkers, J.W.H.M. Uiterwijk, H. Iida and H.J. van den Herik	Beta-Pruning and Opponent-Model Search (14 pages)
CS 01-01	A.J. van Zanten	Iterative Solutions for the Tower of Hanoi Problem with Four Pegs (21 pages)
CS 01-02	I.G. Sprinkhuizen-Kuyper	Artificial Evolution of Box-pushing Behaviour (26 pages)
CS 01-03	J.P.G.M. Verbeek	Euro-Info (51 pages)
CS 01-04	J.W.H.M. Uiterwijk	The CMG Sixth Computer Olympiad. Computer-Games Workshop Proceedings
CS 01-05	H.J.van den Herik, J.W.H.M. Uiterwijk J. van Rijswijck	Games Solved: Now and in the Future (44 pages)
CS 02-05	N. Roos, P.H.M. Spronck E.N. Smirnov	ProANITA April 2005. Eindrapport (51 pages)
MICC 06-01	A.J. van Zanten L. Haryanto	Covers and Near-Covers of the Hypercube Q16 by Symmetric Snakes (59 pages)
MICC 06-02	G. de Croon, I.G., Sprinkhuizen-Kuyper, E.O. Postma	Comparing Active Vision Models (28 pages)
MICC 06-03	A.J. van Zanten L. Haryanto	A 2^{m-1}-cover of Snakes for Qn, $2^{m-1} < n \leq 2^m$ (50 pages)
MICC 07-01	S. de Jong, K. Tuyls, N. Roos, K. Verbeeck	A Descriptive Model of Priority-Based Fairness (14 pages)
MICC 07-02	S. Vanderlooy, I.G. Sprinkhuizen-Kuyper	An Overview of Algorithmic Randomness and its Application to Reliable Instance Classification (21 pages)
MICC 07-03	S. Vanderlooy, L. van der Maaten, I.G. Sprinkhuizen-Kuyper	Off-Line Learning with Transductive Confidence Machines: an Empirical Evaluation (36 pages)
MICC 07-04	S. de Jong, M. Ponsen, K. Tuyls, K. Verbeeck	Proceedings ALAMAS 2007 Conference (183 pages)
MICC 07-05	N. Roos, C. Witeveen	Diagnosis of Plan Structure Violations (15 pages)
MICC 07-06	H.J. van den Herik, J.W.H.M. Uiterwijk, M. Winands and M. Schadd	Proceedings Computer Games Workshop 2007 (243 pages)

Copies of these reports may be obtained from the secretarial office of MICC/IKAT, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: herik@micc.unimaas.nl