Matrix Learning for Topographic Neural Maps

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften

vorgelegt von

Banchar Arnonkijpanich

genehmigt von der

Fakultät für Mathematik/Informatik und Maschinenbau
der Technischen Universität Clausthal

Tag der mündlichen Prüfung
28. May 2010

Die Arbeit wurde angefertigt am Institut für Informatik an der Technischen Universität Clausthal.

Dekan der Fakultät:	Prof. Dr. J. Dix
Berichterstatterin	Prof. Dr. B. Hammer
Mitberichterstatter	Prof. Dr. T. Villmann
	Prof. Dr. S. Hartmann

CLAUSTHAL UNIVERSITY OF TECHNOLOGY

Matrix Learning for Topographic Neural Maps

A dissertation submitted in satisfaction
of the requirements for the degree
Doctor of Natural Science in Computer Science

by

Banchar Arnonkijpanich

2010

To John Forbes Nash Jr.

and Jesus Christ

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my supervisor, Prof. Dr. Barbara Hammer, for her valuable suggestions, continuous guidance and encouragement through the time of my studies at TU-Clausthal. My sincere thanks to my colleague, Dr. Alexander Hasenfuss, for his suggestions, help, and fellowship.

Many thanks to Institute of Computer Science at TU-Clausthal for workspace and instruments. This thesis would not have been materialized without scholarship from Khon Kaen University and Centre of Excellence in Mathematics of Thailand.[†]

Finally, I would like to dedicate this thesis to John Forbes Nash Jr., who gave me the inspiration of having a life, and Jesus Christ, who kept me alive on His mercy and grace.

[†]This research is partially supported by the Centre of Excellence in Mathematics, the Commission on Higher Education, Thailand.

VITA

1979	Born, Bangkok, Thailand.
2001	B.Sc. (Mathematics) , Khon Kaen University, Thailand.
2003	M.Sc. (Computational Science), Chulalongkorn University, Thailand.
2006–2007	Pre-Doctoral Student, Interdisciplinary Center for Scientific Computing, University of Heidelberg, Germany.
2008–2010	Doctoral Student, Department of Computer Science, Clausthal University of Technology, Germany.

ABSTRACT OF THE DISSERTATION

Matrix Learning for Topographic Neural Maps

by

Banchar Arnonkijpanich

Doctor of Natural Science in Computer Science

Clausthal University of Technology, 2010

Since electronic data sets are increasing rapidly with respect to both, size of the data sets and data resolution, i.e. dimensionality, adequate data inspection and data visualization have become central issues of data mining. In consequence, a multitude of data organization schemes such as clustering or topographic mapping have emerged, two of the most popular methods being the self-organizing map (SOM) and neural gas (NG). SOM as proposed by Kohonen and NG by Martinetz et al. and generalizations thereof such as the generative topographic map constitute popular algorithms to represent data by means of prototypes arranged on a (hopefully) topology representing map. However, most standard methods rely on the Euclidean metric, hence the resulting clusters are isotropic and they cannot account for local distortions or correlations of data. A more general metric with K-means and fuzzy-K-means has been proposed in [29, 12]. This general metric is given by a full adaptive matrix such that ellipsoidal clusters are accounted for. However, algorithms based on K-means are very sensitive to the initialization of the prototype distribution, which can lead to convergence to local minima.

In this thesis, we improve this algorithm by combining a general metric with SOM and NG such that the neighborhood based topology preservation in SOM and NG also reduces the influence of initialization. Thereby, our approach relies on a natural ex-

tension of the standard cost functions of NG and SOM (in the form of Heskes) and is conceptually intuitive. We give a strong mathematical foundation. We derive batch optimization learning rules for prototype and matrix adaptation based on the generalized cost functions and we prove convergence of the algorithm. Thereby, it can be seen that matrix learning implicitly performs local principal component analysis (PCA) and the local eigenvectors correspond to the main axes of the ellipsoidal clusters. Thus, the proposal also provides a cost function associated to alternative proposals in the literature which combine SOM or NG with local PCA models.

Since local PCA is a combination of vector quantization (VQ) and PCA, our approach can be directly applied to classification and clustering. The efficiency of the proposed method is evaluated by benchmark datasets for classification task. Image compression is used as an example of the (un)abilities of clustering for transform coding. Another focus of this work is to demonstrate the applicability of this implicit local PCA to low-dimensional data embedding for data inspection and data visualization. The proposed technique is based on matrix learning for neural gas and manifold charting. This provides an explicit forward mapping from a given high dimensional data space to low dimensionality and its inverse mapping is also estimated. Popular existing manifold learning methods such as Isomap and LLE lack such mapping. We demonstrate the usefulness of the forward mapping of the proposed model in several applications to manifold visualization and structure representation of images while an approximate inverse mapping is used in problems of dynamic texture synthesis as an interesting application of manifold learning.

CHAPTER 1

Introduction

Nowadays, scientific data analysis plays an important role in several disciplines: pattern recognition, digital signal/image processing, computer vision, and others. Scientific data analysis consists of a collection of methods based on machine learning and statistics to deal with the raw data and information obtained through observations, measurements, surveys or experiments about a phenomenon of interest. The purposes of data analysis are to extract as much information as possible as adequate input for the next process, to comprehend the overall meaning of the data, to explore regularities and relations in patterns, and to discover a representation of the data analysis which is easily perceptible. Parallel streams of data analysis from different perspectives can be combined to provide a synergy which leads to a complex data explanation system. In this thesis, we consider the process of analyzing data from different perspectives: clustering, classification, compression, visualization, and reconstruction.

Motivation

Data classfication and clustering aims at grouping observations that are similar together and assigning these observations to classes on the basis of measurements made on the observations. Several models have been proposed in order to perform classification and clustering: single layer networks [89], the back-propagation algorithm for learning multi-layer-perceptrons (MLP) [90], or competitive learning for self-organizing

maps [91, 92]. These methods often use the manhattan distance, Euclidean distance, and dot product as the measure of the similarity between two data points. Although these distance metrics are simple and convenient to implement, they are not appropriate if correlations or scalings have to be dealt with. The problem is particularly pronounced for unsupervised metric based approaches such as clustering since the metric is usually not adapted to further information in this case.

In this work, we emphasize on clustering and prototype based classification in which prototypes models are used instead of models derived from the connection weights between neurons as MLP. In this framework, we integrate an adaptive distance metric including learning a full matrix. This metric provides correlations between dimensions and can be adjusted gradually from isotropic to ellipsoidal distance which depends on the data distribution. The metric based on the full matrix is added to topographic neural map algorithms such as the self-organizing map (SOM) and neural gas (NG) and subsequently referred to as matrix learning for topographic neural maps or shortly matrix clustering. Interestingly, it can be seen by using batch optimization on matrix clustering that the prototypes converge to the center of clusters and a full matrix attached to each prototype implicitly yields local principal components (local PCA). Thus, the method can be considered as a blending of vector quantization (VQ) and principal component analysis (PCA).

Unlike existing proposals in the literature, our approach derives the learning rules from a general cost function such that a sound mathematical background is given and convergence can be proved. We demonstrate the usefulness of matrix clustering on a number of benchmarks. First, we test the method for standard clustering and classification tasks. Then, we apply the method for image compression. Local PCA is suitable for the task of data compression since it can remove spatial redundancy using VQ based representation and can encode the origianal data onto lower-dimensional

compact version using linear PCA. Therefore, image compression can be seen as a standard test in order to compare the efficiency between our approach and existing local PCA methods.

One advantage of using local PCA is that each data point has its own local coordinates by projection onto the low dimensional space induced by the local linear transforms. However these local representations might not lead to a comprehensible visualization of the overall data since an understandable visualization should be defined on a single coordinate space. Global coordination models such as locally linear coordination (LLC) or manifold charting (MC) can be employed to glue the local representations into a global coordinate system. Actually, the task of data visualization can be executed by manifold embedding based methods such as Isomap and LLE as well. Nevertheless, both Isomap and LLE have some drawbacks such as lacking an explicit mapping between the original data space and the low dimensional space. But both, LLC and MC, can defeat this limitation and are also combined with matrix clustering.

We demonstrate that matrix learning can succesfully be used as the first step in this manifold learning procedure. The resulting embedding is particularly intuitive and tractable, such that it also provides an approximate inverse of the embedding. This fact will be used in this thesis in the context of dynamic texture generation, a nontrivial task from computer graphics, to demonstrate the suitability of the proposed manifold learning procedure.

Contributions

The work of this thesis proposes a general mathematical framework for matrix learning based on a cost function and it makes the topographic neural map methods with matrix learning scheme applicable to several applications, e.g. image compression, manifold

visualization and dynamic texture synthesis, and provides the framework for implicit local PCA and distance metric learning. Especially, the major contributions include:

- Deriving batch learning rules for matrix learning from a general cost function and linking this approach to local PCA methods.
- Proving convergence of the learning algorithm based on the mathematical foundation by means of a cost function.
- Applying our approach, implicit local PCA, to problems of classification, clustering, and image compression.
- Deriving local coordinates and responsibilities from matrix clustering and combining it with manifold charting in order to perform data visualization and global manifold learning.
- Testing manifold charting based on matrix clustering with artificial and image data sets to study the internal representation in low dimensional space.
- Extending global manifold learning to an approximate inverse mapping and demonstrating the usefulness of the embedding for dynamic texture generation, as an example.
- Parts of the presented work have been published / submitted to international journals / conferences:

    1. B. Arnonkijpanich, B. Hammer, A. Hasenfuss, C. Lursinsap, Matrix Learning for Topograhpic Neural Maps, In: V. Kurkova, R. Neruda, J. Koutnik (eds.), ICANN'2008, pp. 572-582, Springer, 2008.
    2. B. Arnonkijpanich, A. Hasenfuss, B. Hammer, Local Matrix Adaptation in Clustering and Applications for Manifold Visualization, Neural Networks, accepted.

3. B. Arnonkijpanich, B. Hammer, Global Coordination based on Matrix Neural Gas for Dynamic Texture Synthesis, In: N. El Gayar, F. Schwenker (eds.), ANNPR'2010, pp. 84-95, Springer, 2010.
4. B. Arnonkijpanich, A. Hasenfuss, B. Hammer, Local Matrix Adaptation in Topographic Neural Maps, Neurocomputing, under revision.

Thesis structure

This thesis is divided into six chapters. In the first chapter, we identify the basic problem and scope of work. Chapter 2 introduces topographic neural maps with batch and online learning. Chapter 3 describes an extension of neural maps to local PCA. In this chapter, matrix clustering with batch optimization is introduced. The mathematical proofs of convergence and complexity are presented. A restriction to low rank matrices is discovered to achieve lower complexity and better numerical stability. Chapter 4 shows the experimental results on benchmark data evaluated by two different measures: cluster coherence and classification accuracy. Several applications are demonstrated on problems of classification, clustering, and image compression. Likewise, evaluation criteria and the compression ratio used in image compression are discussed. Chapter 5 contains the integration of matrix clustering into two applications, i.e. manifold visualization and dynamic texture synthesis. This chapter gives an extension of matrix neural gas (MNG) to mixtures of local linear models with responsibilities, and introduces MNG as the basis for manifold charting. The visualization results on artificial and real life data sets are shown, in which 1-NN classification and trustworthiness are used as evaluation measures. Furthermore, visible comparisons on image data reveal the structure representation of images. In the latter part of this chapter, we address the problem of dynamic texture synthesis and provide a comparison between MNG and a probabilistic model together with charting and traversing

technique. Eventually, in chapter 6, this thesis concludes with a summary of the contributions and a discussion of possible future directions for this work.

CHAPTER 2

Topographic neural maps

A neural map is an important neural model in unsupervised vector quantization. It conceptually works by dividing a large set of vectors into groups, and each group is then represented by its centroid or prototype. Neural map algorithms are based on the competitive learning paradigm, and originally used in several applications, e.g. data representation and compression, visualization, clustering, and prototyped based classification. Some of the most popular methods of neural maps are K-means, self-organizing map (SOM), and neural gas (NG). Among these techniques, K-Means is the most simple one but it is very sensitive to the initialization of the prototype distribution, which can lead to convergence to local minima. Therefore, K-means should be improved, e.g. by combining it with topology-preserving algorithms such as SOM and NG. Such neighborhood based topology preservation in SOM and NG also reduces the influence of initialization.

Topographic neural map algorithms: SOM and NG

The self-organizing map (SOM) as proposed by Kohonen constitutes one of the most popular data inspection and visualization tools due to its intuitive, robust, and flexible behavior with numerous applications ranging from web and text mining up to telecommunications and robotics [20]. A SOM consists of two major modules: the input space and the lateral lattice space. At input space, typical prototype vectors are

used to approximate a distribution of input vectors by means of a clustering process in the training data. On the low-dimensional projection space, a fixed low dimensional connectivity between the neurons is arranged on a regular lattice structure. A prototype vector or reference vector is associated to each neuron. This feature is used to connect both modules together and also to visualize data. The SOM generalizes standard vector quantization by integrating this priorly fixed prototype topology, this way achieving a method which is much more insensitive to initial conditions and which also extracts topological information about the given data. The main goal of SOM is to transform patterns in input space which displays arbitrary dimensionality onto one- or two-dimensional array of neurons, such that topological ordering and neighborhood preservation takes place. Because of this topological ordering, it assures that vectors closely located in input space are assigned to adjacent neurons on the topological map.

Neural gas (NG) as introduced by Martinetz transfers ideas of SOM towards an optimum neighborhood structure such that a representation by prototypes and topographic mapping of data can be achieved without any constraints on the topology given by a prior lattice structure [25]. In addition, NG is categorized as unsupervised version of a vector quantization technique with soft competition between the neurons like SOM. NG consists of a set of units called neurons. Each unit has an associated reference vector indicating the position of the prototype. These prototypes are considered as centers of receptive fields in input space. The learning algorithm of NG is quite similar to the one of SOM except for the neighborhood function. In the SOM model, the neigborhood structure is imposed on a lateral lattice space. For the neural gas model, in contrast to SOM, no topology of a fixed dimensionality is defined on the network. Both methods have in common that they extract typical prototypes from the data which are arranged according to the local data characteristics. Thereby, neighborhood integration accounts for the fact that the models are widely insensitive to initial conditions. A variety of alternative prototype based clustering methods have been pro-

posed including standard vector quantization e.g. by means of the popular k-means algorithm, probabilistic extensions such as mixture of Gaussians, or fuzzy variants of prototype-based clustering [3, 15, 41, 51, 52].

Batch and online learning

Popular training algorithms of K-means, SOM and NG can be classified into two major schemes, batch and online. The batch scheme of neural maps can be divided into two steps: vectors assignment and centroids relocation. The first step, vectors assignment, starts once the centroids have been placed in the input space. In this step the algorithm iterates over all vectors in the dataset and assigns each vector to the closest centroid, and vectors that are assigned to a particular centroid form a cluster. The next step, centroids relocation, moves the centroids to the positions in the input space that correspond to the average position of each cluster. The online version can be divided into the same two steps: assignment and relocation. However, both schemes differ in their execution of the two steps. While the batch version iterates over all vectors of the whole dataset before the centroids are relocated, the online version moves a centroid at every step of the iteration, that is, each vector of the dataset pulls the nearest centroid and its neighbors with a certain distance towards itself. In general, this distance is 1% of the distance between the vector and the centroid. The centroids appear to be gliding rather than jumping towards the cluster centres of the dataset. This process is repeated until some termination criterion is reached.

2.1 Formulas of topographic neural maps

Here we introduce the concepts of NG, k-means, and SOM formally. Assume data points $\vec{x} \in \mathbb{R}^m$ are distributed according to a probability distribution P. The goal of

prototype based clustering is to represent data by means of prototype vectors $\vec{w}^i \in \mathbb{R}^m$, $i = 1, \ldots, n$, such that they represent the distribution as accurately as possible. Thereby, a data point $\vec{x} \in \mathbb{R}^m$ is represented by the winning prototype $\vec{w}^{I(\vec{x})}$ which is the prototype located closest to the data point, i.e.

$$I(\vec{x}) = \mathrm{argmin}_i\{d(\vec{x}, \vec{w}^i)\} \tag{2.1}$$

measured according to the squared Euclidean distance

$$d(\vec{x}, \vec{w}^i) = (\vec{x} - \vec{w}^i)^t(\vec{x} - \vec{w}^i)\,. \tag{2.2}$$

The mathematical objective of vector quantization is to minimize the quantization error

$$E_{\mathrm{VQ}}(\vec{w}) = \frac{1}{2}\sum_{i=1}^{n}\int \delta_{i,I(\vec{x})} \cdot d(\vec{x}, \vec{w}^i)\, P(d\vec{x}) \tag{2.3}$$

where $\delta_{i,j}$ denotes the Kronecker delta symbol. Given a finite set of training examples $\vec{x}^1, \ldots, \vec{x}^p$, the popular k-means algorithm optimizes the corresponding discrete cost function

$$E^{disc}_{\mathrm{VQ}}(\vec{w}) = \frac{1}{2} \cdot \sum_{i=1}^{n}\sum_{j=1}^{p} \delta_{i,I(\vec{x}^j)} \cdot d(\vec{x}^j, \vec{w}^i) \tag{2.4}$$

by means of an iterative optimization scheme, at each step optimizing first data assignments and then prototype locations by means of the updates

$$k_{ij} := \delta_{i,I(\vec{x}^j)}, \quad \vec{w}^i := \sum_j k_{ij}\vec{x}^j \Big/ \sum_j k_{ij} \tag{2.5}$$

Since the cost function (2.3) is usually highly multimodal, k-means clustering gets easily stuck in local optima and multiple restarts are necessary to achieve a good solution. There are several proposals to get around this problem, such as e.g. deterministic annealing or alternative continuous optimization schemes [52, 53].

NG and SOM offer further alternatives which integrate neighborhood cooperation of the prototypes. This way, the sensitivity to initial conditions which is given for k-means can be diminished [54], i.e. the models are robust with respect to initialization.

This is one goal of SOM and NG. In addition, neighborhood cooperation accounts for a topological arrangement of prototypes such that browsing and, in the case of SOM with low-dimensional lattice structure, direct visualization of data become possible. The cost function of NG is given by

$$E_{\mathrm{NG}}(\vec{w}) = \frac{1}{2}\sum_{i=1}^{n}\int h_\sigma(k_i(\vec{x}))\cdot d(\vec{x},\vec{w}^i)\,P(d\vec{x}) \tag{2.6}$$

where $k_i(\vec{x}) \in \{0,\ldots,n-1\}$ constitutes the location of prototype i in a permutation of prototypes obtained as follows: the prototypes are arranged according to their distance from vector $\vec{x}$. If the distances are mutually disjoint, it is given by

$$k_i(\vec{x}) = |\{\vec{w}^j \mid d(\vec{x},\vec{w}^j) < d(\vec{x},\vec{w}^i)\}| \tag{2.7}$$

Thereby, $|\{\cdot\}|$ refers to the size of the set of vectors. If distances coincide, ties are broken deterministically. $h_\sigma(t) = \exp(-t/\sigma)$ is a Gaussian shaped curve with neighborhood range $\sigma > 0$. Obviously, for vanishing neighborhood $\sigma \to 0$, the quantization error (2.3) is recovered. For a given finite data set as above, the NG cost function (2.6) can be translated into the corresponding discrete cost function

$$E_{\mathrm{NG}}^{disc}(\vec{w}) = \frac{1}{2}\cdot\sum_{i=1}^{n}\sum_{j=1}^{p} h_\sigma(k_i(\vec{x}^j))\cdot d(\vec{x}^j,\vec{w}^i) \tag{2.8}$$

which can be optimized according to an online or batch scheme. NG is usually optimized by means of a stochastic gradient descent method, i.e. the online version. Online training initializes the neuron weights $\vec{w}^i$ at random and then adapts iteratively all neurons according to a presented pattern $\vec{x}^j$ by the following rules

$$k_{ij} := k_i(\vec{x}^j), \quad \vec{w}^i := \vec{w}^i + \eta\cdot h_\sigma(k_{ij})\cdot[\vec{x}^j - \vec{w}^i] \tag{2.9}$$

where η is the learning rate which decrease with increasing time. In a batch scheme, (2.8) can be optimized in analogy to k-means (2.5) using the update rules

$$k_{ij} := k_i(\vec{x}^j), \quad \vec{w}^i := \sum_j h_\sigma(k_{ij})\vec{x}^j \Big/ \sum_j h_\sigma(k_{ij}) \tag{2.10}$$

as pointed out in the approach [7]. During training, either online or batch, the neighborhood range σ is annealed close to 0 such that the quantization error is approximated in final steps. In intermediate steps, a neighborhood structure of the prototypes is determined by the ranks according to the given training data. This choice, on the one hand, accounts for a high robustness of the algorithm with respect to local minima of the quantization error, on the other hand, NG can be extended to extract an optimum topological structure from the data by means of Hebbian updates (for online NG) [26]. For batch NG, the data topology can be extracted after training, by connecting all prototypes which constitute the first and second winner for at least one data point, as shown in [26]. In both cases, the topology is correct if data and prototypes are sufficiently dense, as specified in [26]. Due to its simple adaptation rule, the independence of a prior lattice, and the independence of initialization because of the integrated neighborhood cooperation, NG offers a simple and highly effective algorithm for data clustering.

SOM uses the adaptation strength $h_\sigma(nd(I(\vec{x}^j), i))$ instead of $h_\sigma(k_i(\vec{x}^j))$, where nd is the distance of neurons $I(\vec{x}^j)$ and i on a priorly chosen, often two-dimensional neighborhood structure of the neurons. A low-dimensional lattice offers the possibility to visualize data easily. However, if the primary goal is clustering, a fixed topology puts restrictions on the map and topology preservation can be violated. The original SOM does not possess a cost function in the continuous case and its mathematical investigation is difficult [5, 57]. A slight change of the winner notation can be associated to a cost function as pointed out by Heskes and colleagues [55, 16]. (Earlier proposals of cost functions for discrete SOM variants or self-organization can be found in [56, 58]; however, we will use the form as proposed by Heskes in this contribution.) We substitute the winner by

$$I^*(\vec{x}) = \mathrm{argmin}_i \sum_{l=1}^{n} h_\sigma(nd(i,l)) d(\vec{x}, \vec{w}^l)\,, \tag{2.11}$$

i.e. the prototype whose lattice-neighbors are closest to $\vec{x}$ as measured by the average distance over the local lattice neighborhood of the prototype. This way, SOM optimizes the cost

$$E_{\mathrm{SOM}}(\vec{w}) = \frac{1}{2}\sum_{i=1}^{n}\int \delta_{i,I^*(\vec{x})} \cdot \sum_{l=1}^{n} h_\sigma(nd(i,l)) \cdot d(\vec{x},\vec{w}^l)\, P(d\vec{x}) \tag{2.12}$$

as pointed out by Heskes [16]. Like NG, SOM can be optimized in online and batch schemes. For a given finite set of training data as beforehand, batch optimization is characterized by the iterative updates

$$k_{ij} := \delta_{i,I^*(\vec{x}^j)}, \quad \vec{w}^i := \sum_{j,l} k_{lj} h_\sigma(nd(l,i))\vec{x}^j \Big/ \sum_{j,l} k_{lj} h_\sigma(nd(l,i)) \tag{2.13}$$

of assignments and prototype vectors. For online optimization which is usually performed through a stochastic gradient descent method, each prototype vector can be iteratively adapted according to a randomly chosen input pattern $\vec{x}^j$ using an update rule

$$\vec{w}^i := \vec{w}^i + \eta \cdot h_\sigma(nd(I(\vec{x}^j),i)) \cdot [\vec{x}^j - \vec{w}^i] \tag{2.14}$$

where the learning rate η is a decreasing function of time. Thereby, the winner $I(\vec{x}^j)$ is used in original SOM, whereas a derivation from the cost function as proposed by Heskes leads to $I^*(\vec{x}^j)$. The neigborhood strength is thereby annealed $\sigma \to 0$ (or a small nonvanishing value, respectively, depending on the application area) during training. This yields to a very fast adaptation towards a topology preserving map (for nonvanishing σ) such that browsing and, in the case of a low dimensional lattice, data visualization become possible. Topological mismatches can occur due to two reasons, a mismatch of data and lattice topology or a too fast adaptation, as discussed e.g. in [59, 11, 55]. In particular for batch SOM, topological defects can occur due to the fast convergence depending on the initialization [11]. For batch NG, the effect seems to be less pronounced [7].

It has been proved in [7] that batch SOM and batch NG converge to a local optimum of the corresponding cost functions in a finite number of steps. Both methods offer clustering schemes which require only few parameters for the adaptation scheme: the number of training epochs, the neighborhood annealing scheme, the number of neurons, i.e. clusters, and the lattice topology. This way, robust methods are obtained which do not need a time consuming optimization of the learning rate.

CHAPTER 3

Local matrix adaptation in topographic neural maps

3.1 Extension of topographic neural maps

Topographic neural maps with non-standard metrics

Clustering methods such as SOM, NG, and k-means rely on the Euclidean metric such that isotropic cluster shapes with spherical class boundaries are found by the models. In consequence, neither different scaling nor correlations of the data dimensions are taken into account, and the methods provide good compression schemes only in the case of essentially isotropic clusters of the data space. To deal with non-isotropic clusters, a large number of prototypes is needed to represent locally non-isotropic data faithfully.

A number of researches use alternative of the single Euclidean metric which better suit the respective situation. The Minkowski metrics as proposed in [10] is one of example which can be transformed into the Manhattan, Euclidean or even Chebychev metric. Several methods extend SOM and NG towards more general distance measures, such as variants which allow to process general dissimilarity data given by a data matrix of pairwise distances only [7, 13, 47, 48, 49, 50]. These methods, however, use a fixed priorly chosen metric. For the Euclidean setting, a number of approaches which adapt the distance calculation to the data at hand have been proposed such that more general cluster shapes can be accounted for and models with few prototypes can

already faithfully represent the data. The methods [32, 35] extend the setting towards a scenario where auxiliary information (such as label information) is available and they adapt the metric such that the aspects which are relevant for this supervised labeling of data are emphasized in the visualization. These settings require additional knowledge for the adaptation of the metric parameters.

Extension of neural maps to local PCA

Methods which are solely based on unsupervised training data for the neural map include the popular adaptive subspace SOM which extracts invariant features in terms of invariant local subspaces attached to the prototypes [21]. A similar idea is proposed in the approaches [23, 28, 37, 1, 19, 42, 43, 27, 46, 8, 40] where prototypes are enriched by vectors corresponding to the local main principal directions of the data. These methods combine principal component analysis (PCA) techniques such as Oja's rule and extensions thereof with a vector quantization scheme such as SOM, possibly using an enriched metric. Thereby, the neighborhood structure of SOM or NG is partially included into the local PCA procedures such that a faster convergence of the results can be achieved, see e.g. [28, 42]. Further, some of the approaches point out the efficient and intuitive visualization capability of the methods, such as [43].

Such extensions of neural maps to local PCA demonstrate that a change of data representation, which corresponds to a change of the metric, can impressively improve the clustering quality. However, most approaches rely on heuristics only, or the learning rules are derived separately for clustering and PCA, respectively. Partially, the methods also formally establish stability of parts of the learning rules such as [28]. To our knowledge, however, none of these approaches derives the learning rules of vector quantization and determination of the principal directions from a uniform cost function and shows convergence of the corresponding learning algorithm. Exceptions

are given by statistical models such as [38, 44, 43] which derive learning rules of class centers and principal directions from uniform statistical approaches which model PCA by means of a latent space model or which are based on mixture models and metric adaptation, respectively. Note that adaptations of statistical models which work particularly well in high dimensions have also been proposed, such as [45]. Learning rules can be derived by means of the expectation maximization algorithm – however, these models do not directly extend the standard SOM or NG update rules, respectively, rather, they use a statistical approach. Further, partially, the methods are sensitive to initialization of the parameters.

The goal of our work is to clarify the connection of standard cost functions of NG and SOM, metric learning by means of matrix adaptation, and local PCA models. More specifically, we will argue in our work that local PCA learning and vector quantization can be interpreted as matrix learning. The latter can be derived from a cost function which constitutes a simple extension of the standard cost function of NG towards an adaptive metric. As a consequence, convergence of a batch optimization scheme can be shown. This batch optimization method leads to tools which are very similar to alternative methods in the literature such as [28]. Therefore, this derivation also gives some theoretical background for efficient methods as already proposed in the literature.

There exist several approaches in the literature which enrich the metric used for vector quantization towards a general adaptive form. One very elegant possibility represents the metric by a full matrix which can take an adaptive weighting of the dimensions as well as correlations of dimensions into account. This scheme has been integrated into standard k-means algorithm and fuzzy extensions thereof as well as supervised prototype-based learning schemes such as learning vector quantization [29, 12, 36] and its algorithmic outline can be seen as a batch variant of the method proposed in [28]. Interestingly, for k-means and fuzzy-k-means, matrix adaptation

corresponds to an estimation of the local Mahalanobis distance of data. Hence the local principal directions are implicitly determined in these approaches and clusters are given by ellipsoids aligned along the local principal directions of data. To our knowledge, no such extensions of neural vector quantization schemes such as NG and SOM towards general adaptive local matrices exist.

In this approach, we extend the cost function of NG and SOM towards general adaptive matrices which characterize the metric. This way, prototype based representations result which rely on local data representations which can take scaling and correlations into account. Thus, the points with equal distance from a prototype have ellipsoidal rather than spherical shape, such that SOM and NG can represent data faithfully using less neurons than the standard Euclidean versions. We derive update rules of the parameters based on a uniform underlying cost function of NG and SOM in the variant as proposed by Heskes [16], relying on batch optimization schemes. The resulting update rules for the matrix correspond to a generalized Mahalanobis distance, hence the method can be linked to local PCA methods. Thus, a link of a formal cost function which resembles the one of NG and SOM, respectively, matrix learning, and local PCA results. We show convergence of the update rules, and we demonstrate their behavior in a variety of benchmark examples and an application to image compression. These experiments mainly serve the purpose to demonstrate that the derived batch rules lead to reasonable data representations which correspond to local PCA models. We demonstrate that initialization of the prototypes has only a minor effect on the outcome of the algorithm. Further, an application to image compression demonstrates that local PCA models can be used as a submodule in interesting problems. All experiments, however, mainly serve as a demonstration of the principled behavior of batch optimization for local PCA models, they are not intended as real-life applications since no adaptation of the model to high dimensionality has been done.

The direct batch optimization scheme uses a generalization of the Mahalanobis distance, i.e. it relies on the inverse of the data covariance matrix. This is often singular or near singular for high dimensional data, such that numerical problems result for the direct batch approach. Further, matrix inversion requires cubic complexity with respect to data dimensionality, i.e. it is not very efficient for high dimensional data. In [28], an elegant solution for this problem is proposed by fixing a number of metric values towards a small multiple of the unit matrix. We discuss, in how far this alternative learning rule can be derived from a constrained optimization of the extended NG cost function. We show that, under reasonable conditions on the size of the eigenvalues, a connection can be established. All of this will be verified in the following sections.

3.2 Batch matrix learning

Classical NG and SOM rely on the Euclidean metric which tends to induce isotropic cluster shapes. Thus the optimal configuration of clusters depends on the scaling or transformation of the data features. General ellipsoidal shapes can be achieved more easily by the generalized metric form

$$d_{\Lambda_i}(\vec{x}, \vec{w}^i) = (\vec{x} - \vec{w}^i)^t \Lambda_i (\vec{x} - \vec{w}^i) \tag{3.1}$$

instead of the squared Euclidean metric (2.2) where $\Lambda_i \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix with $\det \Lambda_i = 1$. These constraints are necessary to guarantee that the resulting formula defines a metric which does not degenerate to a trivial form ($\Lambda_i = 0$ constituting an obvious trivial optimum of the cost functions). The condition $\det \Lambda_i = 1$ could be substituted by an alternative constraint or weaker condition which avoids the convergence of Λ_i towards 0. The setting $\det \Lambda_i = 1$ has the advantage that it can directly be treated analytically and it leads to a very simple and intuitive solution.

A general matrix can induce nonuniform scaling of the data dimensions and include correlations into distance computation, such that a more general form results. This way, the direct connection to vector quantization schemes becomes less clear, but the resulting data representation method becomes more powerful. Obviously, an optimum matrix Λ_i is not known before training. Therefore we optimize the parameter Λ_i according to the given training data. The following cost functions result:

$$E_{\mathrm{NG}}(\vec{w}, \Lambda) = \frac{1}{2} \sum_{i=1}^{n} \int h_\sigma(k_i(\vec{x})) \cdot d_{\Lambda_i}(\vec{x}, \vec{w}^i)\, P(d\vec{x}) \tag{3.2}$$

for matrix NG and

$$E_{\mathrm{SOM}}(\vec{w}, \Lambda) = \frac{1}{2} \sum_{i=1}^{n} \int \delta_{i,I^*(\vec{x})} \cdot \sum_{l=1}^{n} h_\sigma(nd(i,l)) \cdot d_{\Lambda_l}(\vec{x}, \vec{w}^l)\, P(d\vec{x}) \tag{3.3}$$

for matrix SOM, whereby Λ_i is restricted to symmetric positive definite forms with $\det \Lambda_i = 1$ and the assignments $k_i(\vec{x})$ and $I^*(\vec{x})$, respectively, are computed based on all the d_{Λ_i} as follows:

$$k_i(\vec{x}) = |\{\vec{w}^j \mid d_{\Lambda_j}(\vec{x}, \vec{w}^j) < d_{\Lambda_i}(\vec{x}, \vec{w}^i)\}|$$

and

$$I^*(\vec{x}) = \mathrm{argmin}_i \sum_{l=1}^{n} h_\sigma(\mathrm{nd}(i,l)) d_{\Lambda_l}(\vec{x}, \vec{w}^l)\,.$$

Matrix NG

We derive batch optimization schemes for matrix NG and matrix SOM based on these cost functions. First, we consider matrix NG. We assume that a finite number of training data $\vec{x}^1, \ldots, \vec{x}^p$ are given and we consider the associated discrete cost function of NG

$$E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} h_\sigma(k_i(\vec{x}^j)) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i) \tag{3.4}$$

As for batch NG, we introduce hidden variables k_{ij} for $i = 1, \ldots, n$, $j = 1, \ldots, p$ which constitute a permutation of $\{0, \ldots, n-1\}$ for every fixed j, and we extend the cost function to

$$E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij}) = \sum_{ij} h_\sigma(k_{ij}) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)$$

where the hidden variables k_{ij} take the place of the original values $k_i(\vec{x}^j)$, the latter depending on $\vec{w}$ and Λ. Obviously, optimal assignments k_{ij} fulfill the equality $k_{ij} = k_i(\vec{x}^j)$ such that $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij}) = E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda)$ for optimum choices of the hidden variables k_{ij}. Batch optimization, in turn, optimizes the hidden variables k_{ij} for fixed Λ and $\vec{w}$, and it determines optimum parameters Λ and $\vec{w}$ given fixed assignments k_{ij}. Because the cost function $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij})$ has a much simpler form w.r.t. $\vec{w}$ and Λ than the original one, optima can be determined analytically. The learning algorithm is obtained by computing the optima of $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij})$ for fixed k_{ij} and, in turn, for fixed $\vec{w}^i$ and Λ_i, respectively. Based on this consideration update formulas can be derived as follows.

Theorem 3.2.1. *Assume $\vec{w}$ and Λ are fixed. Then optimum assignments k_{ij} of the cost function $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij})$ with the constraint that k_{ij} constitutes a permutation of $\{0, \ldots, n-1\}$ for fixed j, are given by the equation*

$$k_{ij} = k_i(\vec{x}^j). \tag{3.5}$$

Proof: Assume k_{ij} is chosen different from $k_i(\vec{x}^j)$. Then, two indices i_1 and i_2 exist such that $d_{\Lambda_{i_1}}(\vec{x}^j, \vec{w}^{i_1}) < d_{\Lambda_{i_2}}(\vec{x}^j, \vec{w}^{i_2})$ and $k_{i_1 j} > k_{i_2 j}$, thus $h_\sigma(k_{i_1 j}) < h_\sigma(k_{i_2 j})$ due to the monotonicity of h_σ. Thus, we find for the corresponding terms of the cost function $h_\sigma(k_{i_1 j}) \cdot d_{\Lambda_{i_1}}(\vec{x}^j, \vec{w}^{i_1}) + h_\sigma(k_{i_2 j}) \cdot d_{\Lambda_{i_2}}(\vec{x}^j, \vec{w}^{i_2}) > h_\sigma(k_{i_2 j}) \cdot d_{\Lambda_{i_1}}(\vec{x}^j, \vec{w}^{i_1}) + h_\sigma(k_{i_1 j}) \cdot d_{\Lambda_{i_2}}(\vec{x}^j, \vec{w}^{i_2})$, i.e. the assignments are not optimal. Contradiction. □

Theorem 3.2.2. *Given fixed k_{ij}. Assume*

$$S_i := \sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \tag{3.6}$$

is invertible. Then the local optima $\vec{w}$ and Λ of the cost function $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij})$ with constraint $\det \Lambda_i = 1$, Λ_i is symmetric and positive definite, must fulfill

$$\vec{w^i} = \frac{\sum_j h_\sigma(k_{ij})\vec{x}^j}{\sum_j h_\sigma(k_{ij})} \tag{3.7}$$

and

$$\Lambda_i = S_i^{-1}(\det S_i)^{1/m} \tag{3.8}$$

Proof: Note that a positive semidefinite Λ_i with determinant 1 must be positive definite. Therefore, we will deal with the constraint of positive semidefiniteness in the following.

The mapping $g : \mathbb{R}^{m\times m} \to \{\Lambda \in \mathbb{R}^{m\times m} | \Lambda \text{ is symmetric and positive semidefinite}\}$ with $g(\Omega) = \Omega^t\Omega$ is continuous and surjective. Therefore, for every local optimum Λ_i, $\vec{w}$ of E_{NG}^{disc} with the constraint that Λ_i has determinant 1 and Λ_i is symmetric and positive semidefinite, there must exist a local optimum Ω_i, $\vec{w}$ of the corresponding composition E_{NG}^{disc} and g with the constraint that the determinant of $g(\Omega_i)$ is 1 with $\Lambda_i = g(\Omega_i)$. Local optima Ω_i, $\vec{w}$ can be found by Lagrange optimization. Taking into account the constraint on the determinant we arrive at the following Lagrange function for Ω_i and $\vec{w}$:

$$L(\vec{w}, \Omega) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p} h_\sigma(k_{ij}) \cdot d_{g(\Omega_i)}(\vec{x}^j, \vec{w^i}) - \sum_{i=1}^{n} \lambda_i(\det g(\Omega_i) - 1)$$

with Lagrange parameters $\lambda_i \in \mathbb{R}$. We search for stationary points of the Lagrange function with respect to the optimization variables. Hence, we consider the derivatives with respect to Ω and $\vec{w}$.

First, we consider the derivative with respect to $\vec{w}$, more precisely, we consider the directional derivative of L with respect to $\vec{w}$ in direction ξ. This yields

$$\sum_j h_\sigma(k_{ij})(\Lambda_i(\vec{w^i} - \vec{x}^j))^t\xi.$$

Since this must be 0 for every possible ξ, we find $\sum_j h_\sigma(k_{ij})(\vec{w}^i - \vec{x}^j) = 0$ for all i and, hence

$$\vec{w}^i = \frac{\sum_j h_\sigma(k_{ij})\vec{x}^j}{\sum_j h_\sigma(k_{ij})}$$

Taking the derivative of L w.r.t the component $[\Omega_i]_{pq}$ of Ω_i yields

$$\sum_{uv} \frac{\partial L}{\partial [g(\Omega_i)]_{uv}} \cdot \frac{[\partial g(\Omega_i)]_{uv}}{\partial [\Omega_i]_{pq}}$$

because of the chain rule. $[\cdot]_{pq}$ refers to the index pq of a matrix. One can compute

$$\frac{\partial [g(\Omega_i)]_{uv}}{\partial [\Omega_i]_{pq}} = \frac{\partial \sum_l [\Omega_i]_{lu}[\Omega_i]_{lv}}{\partial [\Omega_i]_{pq}} = \begin{cases} 2 \cdot [\Omega_i]_{pq} & \text{if } u = q = v \\ [\Omega_i]_{pv} & \text{if } u = q \neq v \\ [\Omega_i]_{pu} & \text{if } u \neq q = v \\ 0 & \text{otherwise} \end{cases}$$

Therefore, we find

$$\begin{aligned} \sum_{uv} \frac{\partial L}{\partial [g(\Omega_i)]_{uv}} \cdot \frac{\partial [g(\Omega_i)]_{uv}}{\partial [\Omega_i]_{pq}} &= \sum_l \frac{\partial L}{\partial [g(\Omega_i)]_{ql}} \cdot [\Omega_i]_{pl} + \sum_l \frac{\partial L}{\partial [g(\Omega_i)]_{lq}} \cdot [\Omega_i]_{pl} \\ &= 2 \cdot \sum_l \frac{\partial L}{\partial [g(\Omega_i)]_{lq}} \cdot [\Omega_i]_{pl} \end{aligned}$$

because $g(\Omega_i)$ is a symmetric matrix. This vanishes for local optima for all p, q. Hence using $\Lambda_i = g(\Omega_i)$, we find

$$\Omega_i \cdot \frac{\partial L}{\partial [\Lambda_i]_{\bullet q}} = 0$$

for all q, where $[\Lambda_i]_{\bullet q}$ refers to the vectors formed by the columns of Λ_i. Ω_i is invertible because of the constraint $\det \Lambda_i = 1$, thus, we find

$$\frac{\partial L}{\partial [\Lambda_i]_{pq}} = 0$$

for all p and q, i.e. the partial derivatives w.r.t. Λ_i must be 0.

Thus, we consider the derivative with respect to $g(\Omega_i) = \Lambda_i$. This derivative yields

$$\frac{\partial L}{\partial \Lambda_i} = \sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t - \lambda_i(\det \Lambda_i \cdot \Lambda_i^{-1}).$$

Setting this to 0 we obtain

$$\Lambda_i = \left(\sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \right)^{-1} \lambda_i$$

because $\det \Lambda_i = 1$. Using $S_i = \sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t$ we have $S_i \Lambda_i = \lambda_i I$, I being the identity matrix, and, hence, $\det(S_i \Lambda_i) = \det S_i = \lambda_i^m$, thus $\lambda_i = (\det S_i)^{1/m}$. Thus, $\Lambda_i = S_i^{-1} (\det S_i)^{1/m}$ because we assume S_i to be invertible. Thus, the only local optimum of the cost function with $\det \Lambda_i = 1$ yields to Λ_i as specified above.

Note that, obviously, the solution for Λ_i is symmetric because S_i is symmetric, and Λ_i is positive definite because S_i is positive definite. Further, while the solution for Λ_i depends on the optimum value of $\vec{w}$, the solution for $\vec{w}$ is independent of Λ_i, such that the local optima $\vec{w}$, Λ_i of the cost function under the given constraint are given explicitly by these formulas. □

The question occurs under which condition the equality $\det S_i \neq 0$ holds for S_i as defined in eqn. (3.6), such that eqn. (3.8) is well defined. We can find a very simple condition as follows.

Theorem 3.2.3. *The matrix S_i as defined in eqn. (3.6) is invertible if there exist at least $m + 1$ data points $\vec{x}^j$ in general position, i.e. all subsets of m vectors of these data points are linearly independent.*

Proof: Consider $S_i = \sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t$. Note that $h_\sigma(k_{ij}) > 0$ for all σ. Thus, $r_j := \sqrt{h_\sigma(k_{ij})}$ is well defined and positive. (For simplicity, we drop the subscript i.) Define $\vec{y}^j := r_j \cdot (\vec{x}^j - \vec{w}^i)$. If $m + 1$ data points $\vec{x}^j$ in general position exist, we find m linearly independent vectors which do not equal $\vec{w}^i$, say $\vec{x}^1, \ldots, \vec{x}^m$. If these vectors are linearly independent, then also every shift by a constant vector, unless the shift moves one vector to the origin. Thus, $\vec{x}^1 - \vec{w}^i, \ldots, \vec{x}^m - \vec{w}^i$ are also

linearly independent, and so are $\vec{y}^1, \ldots, \vec{y}^m$, which constitute positive multiples of these vectors.

Note that $S_i = \sum_{j=1}^{m} \vec{y}^j (\vec{y}^j)^t + \sum_{j>m} \vec{y}^j (\vec{y}^j)^t =: S^1 + S^2$. We can find an invertible matrix M such that $M\vec{y}^j$ equals the jth unit vector for $j \leq m$ because of the linear independence. Thus, the first sum S^1 can be expressed as $M^{-1} \cdot I \cdot (M^{-1})^t$, I being the identity matrix, which is invertible and positive definite. Because of the matrix equality $(I + AB^t)^{-1} = I - A(I + B^t A)^{-1} B^t$ for matrices A and B, we can express the inverse of the sum $(S^1 + \vec{x}\vec{x}^t)^{-1} = (S^1)^{-1} - ((S^1)^{-1}\vec{x}\vec{x}^t(S^1)^{-1})/(I + \vec{x}^t(S^1)^{-1}\vec{x})$ which is well defined because S^1 is positive definite. Hence, by induction, S_i is invertible. □

The requirement that points lie in general position is usually fulfilled (its probability being one e.g. under the Borel or Lebesgue measure), at least if standard noise is present. Thus, the crucial question when considering this condition is whether the data dimensionality does not exceed the number of training points. This is often fulfilled for 'traditional' data sets where the number of data points usually exceeds the data dimensionality. This is required for many classification and clustering methods since the latter is directly related to the number of free parameters of a classifier or clustering methods. However, with large margin methods becoming available such as the support vector machine, the generalization ability of which depend on the margin achieved by the classifier rather than the data dimensionality, also very high dimensional data sets (e.g. microarray data or mass spectra) have been tackled in the literature, where this condition is violated. We will discuss this issue later.

These considerations give rise to batch optimization for matrix NG. The update formulas (3.5), (3.7), and (3.8) are iteratively applied until convergence. The algorithm is as follows:

Batch matrix NG

init $\vec{w}^i$ (e.g. using random data points)

init Λ_i as identity matrix I

repeat until convergence

determine $k_{ij} := k_i(\vec{x}^j)$

determine $\vec{w}^i := \sum_j h_{\sigma(k_{ij})} \vec{x}^j / \sum_j h_{\sigma(k_{ij})}$

determine $\Lambda_i := S_i^{-1} (\det S_i)^{1/m}$ where

$$S_i := \sum_j h_\sigma(k_{ij})(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t$$

We refer to one iteration, i.e. one adaptation of k_{ij}, $\vec{w}^i$, and Λ_i as one epoch of the algorithm. Usually, the neighborhood σ is annealed to small values during training. Note that the algorithm is well defined if S_i is invertible. As shown above, this holds if at least $m + 1$ training points in general position exist, i.e. invertibility holds for many standard data sets. Note that the incorporation of all data points into the computation of S_i (although possibly only with a small factor due to a small neighborhood) accounts for a robustness of the algorithm such that S_i is (at least theoretically) invertible under a mild condition on the training set.

However, in practice, there will be two problems. On the one hand, a neighborhood close to zero, i.e. very small σ can lead to a matrix S_i which is numerically singular such that matrix inversion is not possible in practice. Further, for high dimensional data, the situation can occur that there exist less training points than data dimensions, hence S_i is always singular. One simple way around this problem is to choose an approximated pseudoinverse $(S_i^t S_i + \epsilon \cdot I)^{-1} S_i$ with small $\epsilon > 0$ and identity matrix I, and the corresponding determinant. This can partially avoid numerical instabilities. Nevertheless, the complexity of this computation is cubic in the data dimensionality such that the learning rule is infeasible for high dimensions. We will discuss later an alternative method with lower complexity by restricting the free parameters of the

learning rule which has been proposed in the literature, and we show first steps of its connection to a variant of the cost function.

Before discussing this issue, we want to have a closer look at the solution which results from these batch optimization formulas. The matrix S_i corresponds to the correlation of the data centered at prototype $\vec{w}^i$ and weighted according to ranked distances from that prototype, ranked relative to distances from other prototypes. For vanishing neighborhood $\sigma \to 0$, the standard correlation matrix of the receptive field is approximated and the distance corresponds to the Mahalanobis distance in the receptive field of the prototype, up to overall scaling of the distance for that prototype. (Note that, as already discussed above, the incorporation of *all* data points into the computation has a stabilizing effect, avoiding singularity of the matrix for reasonable situations.) The Mahalanobis distance corresponds to a scaling of the principal axes of the data space by the inverse eigenvalues in the eigendirections. Thus, ellipsoidal cluster shapes arise whose main directions are centered along the local principal directions of the data, and the scaling is elongated along the main principal components - in these directions, the eigenvalues are large, i.e. their inverses are small; therefore, deviations along these main directions are better tolerated by the induced metric than deviations along the minor components. We conclude that matrix learning performs implicit local PCA and the main local principal components at $\vec{w}^i$ can be discovered by looking at the minor principal components of Λ_i. Unlike standard PCA, neighborhood cooperation is applied to both prototype adaptation and matrix learning during batch training. This is beneficial e.g. for small clusters where the corresponding Mahalanobis distance would be singular since only a small number of points contribute to the correlation matrix.

We would like to point out that it is easily possible to use only one global matrix Λ instead of local matrices Λ_i attached to the prototypes, as proposed in [29]. The resulting formula for Λ is easily obtained from the ones given above by means of the

sum of the local matrices S_i. This way, the number of parameters is reduced and a global Mahalanobis distance which is derived from the data centered around their respective class prototypes is obtained.

Matrix k-means is obtained as the limit $\sigma \to 0$, i.e. only the data points in a receptive field contribute to the location of the prototype and the corresponding matrix.

Matrix SOM

Matrix learning for SOM can be derived in a similar way. We only shortly sketch the corresponding derivation. As before, we introduce hidden variables k_{ij} for $i = 1, \ldots, n$, $j = 1, \ldots, p$ which are contained in $\{0, 1\}$ such that $\sum_i k_{ij} = 1$ for all $\vec{x}^j$. The cost function of SOM for a given discrete data set is substituted by

$$E_{\mathrm{SOM}}^{disc}(\vec{w}, \Lambda, k_{ij}) = \frac{1}{2} \sum_{ij} k_{ij} \cdot \sum_{l} h_\sigma(nd(i,l)) \cdot d_{\Lambda_l}(\vec{x}^j, \vec{w}^l)$$

where the hidden variables k_{ij} substitute the term $\delta_{i,I^*(\vec{x}^j)}$ which depends on $\vec{w}$ and $\vec{\Lambda}$. For optimum assignments k_{ij}, the two values are obviously identical. As before, iterative optimization of $E_{\mathrm{SOM}}^{disc}(\vec{w}, \Lambda, k_{ij})$ w.r.t. $\vec{w}$, Λ and k_{ij} is done, using the respective local optima:

- Assume $\vec{w}^i$ and Λ_i are fixed. Then optimum values of the cost function $E_{\mathrm{SOM}}^{disc}(\vec{w}, \Lambda, k_{ij})$ under the constraint $k_{ij} \in \{0, 1\}$ such that $\sum_i k_{ij} = 1$ for all j are given by

  $$k_{ij} = \delta_{i,I^*(\vec{x}^j)} \tag{3.9}$$

  where the winner $I^*(\vec{x}^j)$ is given by (3.2).

- Assume k_{ij} are fixed. Then, the parameters $\vec{w}^i$ and matrices Λ_i with constraint $\det \Lambda_i = 1$ that yield local optima of the cost function $E_{\mathrm{SOM}}^{disc}(\vec{w}, \Lambda, k_{ij})$ can be computed as follows: As before, we can enforce positive semidefiniteness by

setting $\Lambda_i = \Omega_i^t \Omega_i$. The fact that the derivative of the Lagrangian with respect to Ω_i is 0, can be reduced to the fact that the derivative with respect to Λ_i is vanishing, as beforehand. The corresponding Lagrange function is

$$L(\vec{w}^i, \Lambda_i) = \frac{1}{2} \sum_{ij} k_{ij} \cdot \sum_l h_\sigma(nd(i,l)) \cdot d_{\Lambda_l}(\vec{x}^j, \vec{w}^l) - \sum_i \lambda_i (\det \Lambda_i - 1)$$

with Lagrange parameters $\lambda_i \in \mathbb{R}$.

The directional derivative of the Lagrange function w.r.t. $\vec{w}^i$ into direction ξ vanishes for all directions, hence one can compute

$$\vec{w}^i = \sum_{jl} k_{lj} h_\sigma(nd(l,i)) \vec{x}^j / \sum_{jl} k_{lj} h_\sigma(nd(l,i)) \tag{3.10}$$

Setting the derivative w.r.t Λ_i to 0 we obtain

$$\Lambda_i = \left(\sum_{lj} k_{lj} h_\sigma(nd(l,i)) (\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \right)^{-1} \lambda_i$$

We set

$$S_i := \sum_{lj} k_{lj} h_\sigma(nd(l,i)) (\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t \tag{3.11}$$

and obtain

$$\Lambda_i = S_i^{-1} (\det S_i)^{1/m} \tag{3.12}$$

as before, whereby, now, the correlation matrix S_i is measured taking the lattice structure of SOM into account. As before, Λ_i is symmetric because S_i is symmetric, and Λ_i is positive definite if S_i is positive definite. As before, this holds if at least $m+1$ data points $\vec{x}^j$ in general position exist. Further, vanishing neighborhood $\sigma \to 0$ approximates the standard correlation, i.e. ellipsoidal clusters aligned along the local principal components as for matrix NG.

We substitute the adaptation rules of matrix NG by these new adaptation rules given by eqns.(3.9)-(3.12) to arrive at matrix SOM.

Convergence

Here we show convergence of matrix learning for NG, as an example, the argumentation for SOM being similar. Thereby, we assume that S_i is invertible, e.g. because $m+1$ data points in general position exist. Further, we consider a fixed neighborhood radius σ at first.

First, we prove that the assignments in every epoch constitute not only local optima, but global optima of the cost function if the remaining parts are fixed. This has already been proved for the assignments k_{ij}. For the remaining parameters, we obtain the following:

Theorem 3.2.4. *Assume the assignments k_{ij} are fixed. Assume at least $m+1$ data points $\vec{x}^i$ in general position exist. Then, the choice of $\vec{w}^i$ as given by eq. (3.7) and Λ_i as given by eq. (3.8) constitute global optima of the cost function $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij})$ with respect to $\vec{w}^i$ and Λ_i with the constraints $\det \Lambda_i = 1$, Λ_i is symmetric and positive definite.*

Proof: Note that the solution of $\vec{w}^i$ as given by eq. (3.7) is independent of Λ_i, such that these settings are well defined. They constitute the only local optima of the cost function, as already shown previously.

The feasible range of $\vec{w}^i$ is $\mathbb{R}^m$, i.e. a closed set. The feasible range of Λ_i is the intersection of the matrices with determinant one (a closed set) and the positive definite matrices (also a closed set), i.e. a closed set. We equip the space of prototypes with the standard Euclidean metric and the space of matrices with the Frobenius norm, both denoted by $\|\cdot\|$. We will show that, if $\|\vec{w}^i\| \to \infty$ or $\|\Lambda_i\| \to \infty$, then the cost function $E \to \infty$. Then, because E is continuous and the considered feasible range is closed, the minimum of E with respect to $\vec{w}^i$ and Λ_i must be attained. Since there is only one candidate for a local optimum given by eqns. 3.7 and 3.8, this candidate must be a

global optimum.

Now we show that E increases beyond bounds in the Euclidean norm if a prototype or a matrix does in the Euclidean resp. Frobenius norm. If $\|\vec{w}^i\| \to \infty$, we can find one component $[\vec{w}^i]_l \to \infty$. (This notation refers to component l of the vector.) We can represent every symmetric matrix Λ_i with determinant 1 uniquely in the form $\Lambda_i = U_i^t D_i U_i$ where U_i is contained in the orthogonal group $O(m)$ and D_i is a diagonal matrix with determinant 1. Since $O(m)$ is compact $\|\Lambda_i\| \to \infty$ includes that $[D_i]_{ll} \to \infty$ for one diagonal element of D_i.

There exist at least $m+1$ points in general position, we can assume without loss of generality that these are enumerated as $\vec{x}^1, \ldots, \vec{x}^{m+1}$. We can find a positive ϵ such that a ball with radius ϵ is contained in the convex hull of these points. This property is preserved if the points are transformed by a conformal mapping, since lengths and angles are preserved by such a mapping.

Consider the cost function E. It can be written as

$$\begin{aligned} E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij}) &= \frac{1}{2} \cdot \sum_{ij} h_\sigma(k_{ij}) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i) \\ &= \frac{1}{2} \cdot \sum_{ij} h_\sigma(k_{ij}) \cdot (U_i(\vec{x}^j - \vec{w}^i))^t \cdot D_i \cdot (U_i(\vec{x}^j - \vec{w}^i)) \\ &= \frac{1}{2} \cdot \sum_{ij} h_\sigma(k_{ij}) \cdot \sum_l [D_i]_{ll} [U_i(\vec{x}^j - \vec{w}^i)]_l^2 \,. \end{aligned}$$

Obviously, E decomposes into positive summands. Note that $\vec{x} \mapsto U_i(\vec{x} - \vec{w}^i)$ constitutes a conformal mapping, hence a ball of radius ϵ is contained in the convex hull of the first $m+1$ such points.

Assume $[\vec{w}^i]_l \to \infty$ for some prototype $\vec{w}^i$ and component l. Since $O(m)$ is compact, the vector $U_i(\vec{x}^j - \vec{w}^i)$ contains at least one component which converges to ∞. Thus, either $E \to \infty$ or $[D_i]_{ll} \to 0$ for some matrix element. In the latter case, because of $\det D_i = 1$, we find a matrix element of D_i which converges to ∞, i.e. we are in the

following setting.

Assume $[D_i]_{ll} \to \infty$. Since a ball of radius ϵ is contained in the convex hull of $U_i(\vec{x}^j - \vec{w}^i)$ for $j = 1, \ldots, m+1$ and every choice of $\vec{w}^i$ and U_i, we find coefficients $\alpha_j \geq 0$ with $\sum_{j=1}^{m+1} \alpha_j = 1$, defining a point on the convex hull of the transformed data points, with $[\sum_{j=1}^{m+1} \alpha_j U_i(\vec{x}^j - \vec{w}^i)]_l^2 > \epsilon^2$. Hence $\max_j [U_i(\vec{x}^j - \vec{w}^i)]_l^2 > \epsilon^2$ and, therefore, $E \to \infty$. Thus, in all cases, the cost function converges $E \to \infty$. □

This result allows us to prove convergence of matrix NG.

Theorem 3.2.5. *Assume there exist $m+1$ data points $\vec{x}^i$ in general position. Assume the neighborhood parameter $\sigma > 0$ is fixed. Then, matrix NG converges after a finite number of epochs.*

Proof: Note that the discrete cost function of NG can be written as

$$E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda) = \sum_{i=1}^{n} \sum_{j=1}^{p} f_1(k_{ij}(\vec{w}, \Lambda)) \cdot f_2^{ij}(\vec{w}, \Lambda) \tag{3.13}$$

where $f_1(k_{ij}) = h_\sigma(k_i(\vec{x}^j))$ and $f_2^{ij} = d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)$.

Batch optimization substitutes the dependent values $k_{ij}(\vec{w}, \Lambda)$ by new hidden parameters which are optimized under the constraint that k_{ij} constitute a permutation of $\{0, \ldots, n-1\}$ for every fixed i. For optimum values k_{ij} given fixed $\vec{w}$, Λ, the equality $k_{ij} = k_{ij}(\vec{w}, \Lambda)$ holds. Batch clustering in turn finds optimum values $\vec{w}$ and Λ, given fixed assignments k_{ij} and it determines optimum assignments k_{ij} given fixed parameters $\vec{w}$ and Λ, as we have already shown. We can assume that the respective optima are unique, which is obvious from the formulas for $\vec{w}$ and Λ, and which can be achieved for k_{ij} by breaking ties deterministically.

Consider the function

$$Q(\vec{w}, \Lambda, \vec{w}', \Lambda') = \sum_{i=1}^{n} \sum_{j=1}^{p} f_1(k_{ij}(\vec{w}, \Lambda)) \cdot f_2^{ij}(\vec{w}', \Lambda') \tag{3.14}$$

where $k_{ij}(\vec{w}, \Lambda)$ denotes the unique optimum assignment of k_{ij} for the cost function (3.13), given fixed $\vec{w}$ and Λ. It holds $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda) = Q(\vec{w}, \Lambda, \vec{w}, \Lambda)$. Denote by $\vec{w}(t)$ and $\Lambda(t)$ the values derived in batch clustering in epoch t. Then we find

$$\begin{aligned} E_{\mathrm{NG}}^{disc}(\vec{w}(t+1), \Lambda(t+1)) &= Q(\vec{w}(t+1), \Lambda(t+1), \vec{w}(t+1), \Lambda(t+1)) \\ &\leq Q(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) \end{aligned}$$

because $k_{ij}(\vec{w}(t+1), \Lambda(t+1))$ are optimum assignments for given $\vec{w}(t+1)$ and $\Lambda(t+1)$. Further, we have a strict inequality if the assignments change i.e. $k_{ij}(\vec{w}(t+1), \Lambda(t+1)) \neq k_{ij}(\vec{w}(t), \Lambda(t))$. Further, we find

$$\begin{aligned} Q(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) &\leq \\ Q(\vec{w}(t), \Lambda(t), \vec{w}(t), \Lambda(t)) &= E_{\mathrm{NG}}^{disc}(\vec{w}(t), \Lambda(t)) \end{aligned}$$

because $\vec{w}(t+1)$ and $\Lambda(t+1)$ are chosen as optimum values for the assignments $k_{ij}(\vec{w}(t), \Lambda(t))$. Hence

$$E_{\mathrm{NG}}^{disc}(\vec{w}(t+1), \Lambda(t+1)) \leq E_{\mathrm{NG}}^{disc}(\vec{w}(t), \Lambda(t))$$

Thus, the cost function (3.14) is decreased in consecutive steps and the inequality is strict if the assignments k_{ij} change. Since the assignments k_{ij} stem from a finite set and the respective optimum values are unique, the algorithm converges in a finite number of steps towards a fixed point $\vec{w}^*$, Λ^* of the algorithm. □

Further, one can show that this fixed point is at a local optimum of the cost function $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda)$ (and not only the correlated function $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda, k_{ij})$) under mild conditions on the solution.

Theorem 3.2.6. *Assume $\vec{w}^*$, Λ^* is a fixed point of matrix NG with fixed neighborhood parameter σ. Assume that there do not exist prototypes $\vec{w}^i$ and $\vec{w}^j$ in $\vec{w}^*$ and data points $\vec{x}$ with $d_{\Lambda_i}(\vec{w}^i, \vec{x}) = d_{\Lambda_j}(\vec{w}^j, \vec{x})$. Then $\vec{w}^*$ and Λ^* is a local optimum of $E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda)$.*

Proof: If $d_{\Lambda_i}(\vec{w}^i, \vec{x}) \neq d_{\Lambda_j}(\vec{w}^j, \vec{x})$ for every two prototypes in the found solution $\vec{w}^*$, the underlying cost function (3.13) is continuous at $\vec{w}^*$, Λ^*, and, since the assignments k_{ij} are discrete, the function $k_{ij}(\vec{w}, \Lambda)$ is constant in a vicinity of $\vec{w}^*$, Λ^*. Hence $E_{\mathrm{NG}}^{disc}(\cdot)$ and $Q(\vec{w}^*, \Lambda^*, \cdot)$ are identical in a neighborhood of $\vec{w}^*$, Λ^*, i.e. local optima of Q correspond to local optima of E. As shown beforehand, we arrive at a local optimum of this function simultaneously for the matrices and prototypes given fixed assignments k_{ij} . Hence, we arrive at a local optimum of E itself, if no distances of a data point to two different prototypes coincide (which is a set of measure zero). □

It is guaranteed that the algorithm converges to a local optimum of the cost function for fixed neighborhood radius σ. Note, however, that the found solution can be a local but not a global optimum of the cost function E.

One can extend the result on convergence to the case of varying neighborhood parameter $\sigma \to 0$. As before, one epoch refers to one loop, i.e. one adaptation of k_{ij}, $\vec{w}^i$, and Λ_i. We refer to the neighborhood parameter in the t'th epoch by σ_t. We will need in the proof of the convergence that the assignment of data points based on the distance to their winner is unique. To guarantee this a priori, we need a further assumption on the data set: When performing matrix NG, the parameters k_{ij} obtained in an epoch give rise to winner assignments $\alpha_{ij} \in \{0, 1\}$ with $\alpha_{ij} = 1 \iff k_{ij} = 0$. The limit of the corresponding prototypes for $\sigma \to 0$ is given by $\vec{w}^i(\alpha) := \sum_j \alpha_{ij}\vec{x}^j / \sum_j \alpha_{ij}$ and the matrices are formed by $\Lambda_i(\alpha) := S_i^{-1}(\det S_i)^{1/m}$ where $S_i = \sum_j \alpha_{ij}(\vec{x}^j - \vec{w}^i)(\vec{x}^j - \vec{w}^i)^t$. We refer to these values as limit prototype and limit matrix, respectively. Assume that, for a choice $\alpha_{ij} \in \{0, 1\}$ the corresponding matrices S_i are invertible and no two indices i and i' and a data point $\vec{x}^j$ exist with $d_{\Lambda_i(\alpha)}(\vec{x}^j, \vec{w}^i(\alpha)) = d_{\Lambda_{i'}(\alpha)}(\vec{x}^j, \vec{w}^{i'}(\alpha))$. We refer to this property as the fact that *winner assignments are unique for the data set*. Note that this property is fulfilled with probability one since the regions with equal distance from two prototypes have

Lebesgue measure 0 since they are defined by a (differentiable) equation. Matrices are invertible if the number of data points exceeds the dimensionality. However, in practice, problems concerning matrix invertibility can occur in particular for large data dimensionality.

Theorem 3.2.7. *Assume matrix NG is performed using neighborhood parameter σ_t in the t'th epoch such that $\sigma_t \geq \sigma_{t+1}$ for all t and $\lim_{t\to\infty} \sigma_t = 0$. Assume that winner assignments are unique for the given data set in every epoch. Then, the algorithm converges.*

Proof: In epoch t, the cost function, extended by the hidden variables, is given by

$$E_{\mathrm{NG}}^{\sigma_t}(\vec{w}, \vec{\Lambda}, k_{ij}) = \sum_{ij} h_{\sigma_t}(k_{ij}) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)\,.$$

In each epoch, optimum values are determined as assignments

$$k_{ij}(t) = |\{\vec{w}^l(t-1) \mid d_{\Lambda_l(t-1)}(\vec{x}^j, \vec{w}^l(t-1)) < d_{\Lambda_i(t-1)}(\vec{x}^j, \vec{w}^i(t-1))\}|\,,$$

prototype vectors

$$\vec{w}^i(t) = \sum_j h_{\sigma_t}(k_{ij}(t))\vec{x}^j \Big/ \sum_j h_{\sigma_t}(k_{ij}(t))$$

and matrices

$$\vec{\Lambda}_i(t) = S_i(t)^{-1}(\det S_i(t))^{1/m}$$

where

$$S_i(t) = \sum_j h_{\sigma_t}(k_{ij}(t))(\vec{x}^j - \vec{w}^i(t))(\vec{x}^j - \vec{w}^i(t))^t\,.$$

Note that $k_{ij}(t)$ is independent of σ, rather, it depends on the prototypes and matrices. We write

$$k_{ij}(\vec{w}, \Lambda) = |\{\vec{w}^l \mid d_{\Lambda_l}(\vec{x}^j, \vec{w}^l) < d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)\}|$$

to make this functionality explicit.

The function Q^{σ_t} which corresponds to $E_{\mathrm{NG}}^{\sigma_t}$ can be defined as above

$$Q^{\sigma_t}(\vec{w}, \Lambda, \vec{w}', \Lambda') = \sum_{ij} h_{\sigma_t}(k_{ij}(\vec{w}, \Lambda)) \cdot d_{\Lambda_i'}(\vec{x}^j, (\vec{w}')^i) \,.$$

As before, we find

$$\begin{aligned} E_{\mathrm{NG}}^{\sigma_{t+1}}(\vec{w}(t+1), \Lambda(t+1)) &= Q^{\sigma_{t+1}}(\vec{w}(t+1), \Lambda(t+1), \vec{w}(t+1), \Lambda(t+1)) \\ &\leq Q^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) \end{aligned}$$

because $k_{ij}(\vec{w}(t+1), \Lambda(t+1))$ are optimum assignments for given $\vec{w}(t+1)$ and $\Lambda(t+1)$ (independent of σ_{t+1}). Further, we find

$$\begin{aligned} Q^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) &\leq \\ Q^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t), \vec{w}(t), \Lambda(t)) &= E_{\mathrm{NG}}^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t)) \end{aligned}$$

because $\vec{w}(t+1)$ and $\Lambda(t+1)$ are chosen as optimum value for the assignments $k_{ij}(t+1) = k_{ij}(\vec{w}(t), \Lambda(t))$ and σ_{t+1}. Hence

$$E_{\mathrm{NG}}^{\sigma_{t+1}}(\vec{w}(t+1), \Lambda(t+1)) \leq E_{\mathrm{NG}}^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t))$$

Further, it holds

$$h_{\sigma_t}(x) \geq h_{\sigma_{t+1}}(x)$$

for every $x > 0$ and $\sigma_t \geq \sigma_{t+1}$, thus

$$E_{\mathrm{NG}}^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t)) \leq E_{\mathrm{NG}}^{\sigma_t}(\vec{w}(t), \Lambda(t)) \,.$$

Thus, the energy function decreases in every step.

We assume that winner assignments are unique in every step, as defined above. Since the number of different possible winner assignments α_{ij} is finite, we can find some $\epsilon > 0$ such that, for every data point, its distance to the winner computed from the corresponding limit prototypes and limit matrices is at least ϵ smaller than the distance from the second closest prototype.

Note that prototypes are contained in the convex hull of data points, i.e. prototypes are contained in a bounded set. The same holds for the matrices if σ_t is small enough under the assumption that the limit matrices corresponding to the current k_{ij} (and thus also a sufficiently close approximation thereof) exist and are invertible. Therefore, distances which are computed based on these data are also bounded.

Consider the function

$$Q^{\sigma_{t+1}}(\vec{w}(t+1), \Lambda(t+1), \vec{w}(t+1), \Lambda(t+1)) \,.$$

The winner assignment which corresponds to $k'_{ij}(\vec{w}(t+1), \Lambda(t+1))$ is referred to as α. As before, $\vec{w}^i(\alpha)$ and $\Lambda_i(\alpha)$ denote the limit prototypes and matrices computed based on these values. Assume that, for some j_0, the winner assignment changes in this step, i.e. there exist $i_0 \neq i'_0$ with $k_{i_0 j_0}(\vec{w}(t+1), \Lambda(t+1)) = 0$ and $k_{i'_0 j_0}(\vec{w}(t), \Lambda(t)) = 0$. Because of the assumptions made above, it holds

$$d_{\Lambda_{i_0}(\alpha)}(\vec{x}^{j_0}, \vec{w}^{i_0}(\alpha)) + \epsilon \leq d_{\Lambda_{i'_0}(\alpha)}(\vec{x}^{j_0}, \vec{w}^{i'_0}(\alpha)) \,.$$

Because of the continuity of the operation and the boundedness of the parameters, we can assume

$$|d_{\Lambda_i(t+1))}(\vec{x}^j, \vec{w}^i(t+1)) - d_{\Lambda_i(\alpha)}(\vec{x}^j, \vec{w}^i(\alpha)| \leq \epsilon/4$$

for every i and j and large enough t because of $\lim_{t\to\infty} \sigma_t = 0$. Further, again because of the boundedness of the distances we can assume that

$$\sum_{i \,|\, k_{ij}(t+1)\neq 0} h_{\sigma_{t+1}}(k_{ij}(t+1)) \cdot d_{\Lambda_i(t+1)}(\vec{x}^j, \vec{w}^i(t+1)) \leq \epsilon/4$$

for every j and large enough t. Further, $k_{ij}(\vec{w}(t+1), \Lambda(t+1))$ are determined as

optimum values independently for every j. Therefore, we find

$$
\begin{aligned}
& Q^{\sigma_{t+1}}(\vec{w}(t+1), \Lambda(t+1), \vec{w}(t+1), \Lambda(t+1)) \\
= \; & \textstyle\sum_{i \neq i_0, j \neq j_0} h_{\sigma_{t+1}}(k_{ij}(\vec{w}(t+1), \Lambda(t+1))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^j, \vec{w}^i(t+1)) \\
& + \textstyle\sum_{i \neq i_0} h_{\sigma_{t+1}}(k_{ij_0}(\vec{w}(t+1), \Lambda(t+1))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^{j_0}, \vec{w}^i(t+1)) \\
& + h_{\sigma_{t+1}}(k_{i_0 j_0}(\vec{w}(t+1), \Lambda(t+1))) \cdot d_{\Lambda_{i_0}(t+1)}(\vec{x}^{j_0}, \vec{w}^{i_0}(t+1)) \\
\leq \; & \textstyle\sum_{i \neq i_0, j \neq j_0} h_{\sigma_{t+1}}(k_{ij}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^j, \vec{w}^i(t+1)) \\
& + \epsilon/4 \\
& + 1 \cdot d_{\Lambda_{i_0}(\alpha)}(\vec{x}^{j_0}, \vec{w}^{i_0}(\alpha)) + \epsilon/4 \\
\leq \; & \textstyle\sum_{i \neq i_0, j \neq j_0} h_{\sigma_{t+1}}(k_{ij}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^j, \vec{w}^i(t+1)) \\
& + \epsilon/4 + \textstyle\sum_{i \neq i'_0} h_{\sigma_{t+1}}(k_{ij_0}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^{j_0}, \vec{w}^i(t+1)) \\
& + 1 \cdot d_{\Lambda_{i'_0}(\alpha)}(\vec{x}^{j_0}, \vec{w}^{i'_0}(\alpha)) - \epsilon + \epsilon/4 \\
\leq \; & \textstyle\sum_{i \neq i_0, j \neq j_0} h_{\sigma_{t+1}}(k_{ij}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^j, \vec{w}^i(t+1)) \\
& + \epsilon/4 + \textstyle\sum_{i \neq i'_0} h_{\sigma_{t+1}}(k_{ij_0}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^{j_0}, \vec{w}^i(t+1)) \\
& + h_{\sigma_{t+1}}(k_{i'_0 j_0}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_{i'_0}(t+1)}(\vec{x}^{j_0}, \vec{w}^{i'_0}(t+1)) - \epsilon + \epsilon/4 + \epsilon/4 \\
= \; & \textstyle\sum_{ij} h_{\sigma_{t+1}}(k_{ij}(\vec{w}(t), \Lambda(t))) \cdot d_{\Lambda_i(t+1)}(\vec{x}^j, \vec{w}^i(t+1)) - \epsilon/4 \\
= \; & Q^{\sigma_{t+1}}(\vec{w}(t), \Lambda(t), \vec{w}(t+1), \Lambda(t+1)) - \epsilon/4
\end{aligned}
$$

Thus, if the winner assignment changes, the cost function decreases by at least $\epsilon/4$ for large enough t. This can happen at most a finite number of times, i.e. the winner assignments are fixed after a finite number of steps. The limit of the prototypes depends on the winner assignments only, as does the limit for matrices Λ_i, therefore, the algorithm converges. □

3.3 Matrix learning using low rank matrices

High dimensional data

We derived batch formulas for a full matrix adaptation of SOM and NG. Unlike previous work on matrix adaptation or related local PCA methods such as [23, 28, 37, 29] the algorithm has been derived from a single cost function which directly extends the standard cost function of NG and SOM and convergence of the algorithm can be guaranteed. Note that, unlike the approaches [23, 28, 37] neighborhood cooperation is included into the matrix update or the corresponding PCA, respectively, to make sure that a global cost function is optimized by the methods. This has the further benefit that the corresponding generalized Mahalanobis matrix exists if at least $m + 1$ data points in general position exist. Once the connection of matrix learning and PCA is established by means of this general derivation, heuristics which obtain an optimum matrix by means of alternative PCA schemes such as variations of the Oja rule (e.g. [28]) are justified since they arrive at the same result for Λ as the above method. Note that, to guarantee convergence of the algorithm, it is sufficient to guarantee improvement of the cost function in every step, but a global optimum of the considered parameter need not necessarily be found. Thus convergence of alternative optimization schemes including online schemes such as gradient methods directly follows.

Batch clustering usually converges after only a very small number of epochs since the update rules can be interpreted as (fast) Newton method [3, 7]. However, we are not aware of a formal upper bound on the number of epochs. The main complexity of one epoch of matrix learning is due to the update of the matrix Λ which requires matrix inversion and a normalization by means of the determinant, i.e. steps with complexity $O(m^3)$, m being the data dimensionality. While this can easily be done for low dimensional data sets, it becomes infeasible for high dimensionality. Therefore,

the learning rule must be changed for high dimensional data. In addition, numerical instabilities can occur for higher dimensionalities and full matrix adaptation, since the variance can become very small in the minor principal components of the data. Therefore, alternatives should be used.

The approach [28] proposes an intuitive alternative regularization scheme to reduce numerical instabilities and to obtain a lower complexity: it computes only $k \ll m$ major principal components and it scales all remaining dimensions uniformly. Note that iterative approaches which compute the major k principal components online exist, such as the classical Sanger's rule or improvements [34, 60]. These approaches are linear for one epoch of the algorithm. Commonly, the convergence speed depends on the relative size of the eigenvalues. If, for matrix learning, a fixed number of iterations of eigenvalue and eigenvector learning is included into one epoch of matrix learning, the resulting complexity is only linear in one epoch of matrix learning.

Note that the weighting of any eigendirection should not be set to 0 to guarantee the metric property and locality of clusters, i.e. eigenvalues of matrices Λ_i that are exactly 0 should be avoided. In the approach [28] the following metric is suggested (up to normalization which is achieved in [28] by subtracting the logarithm of the determinant of the matrix):

$$d_\Lambda(\vec{x}, \vec{w}) = (\vec{x}-\vec{w})^t \Lambda (\vec{x}-\vec{w}) + \frac{1}{\lambda^*}((\vec{x}-\vec{w})^t(\vec{x}-\vec{w}) - (\vec{x}-\vec{w})^t U^t U(\vec{x}-\vec{w})) \quad (3.15)$$

where $\Lambda = U^t D U$ is the eigenvalue decomposition of the rank k matrix Λ. The matrix U contains the k major principal components of the correlation matrix and diagonal matrix D contains the inverses of the corresponding eigenvalues. λ^* is an estimation of the variance in the remaining $m-k$ directions. Note that this metric corresponds to

the choice

$$
\begin{aligned}
&(\vec{x}-\vec{w})^t \begin{pmatrix} U \\ U^\perp \end{pmatrix}^t \cdot \begin{pmatrix} D & 0 \\ 0 & D_{\lambda^*} \end{pmatrix} \cdot \begin{pmatrix} U \\ U^\perp \end{pmatrix} (\vec{x}-\vec{w}) = \\
&(\vec{x}-\vec{w})^t U^t D U (\vec{x}-\vec{w}) + (\vec{x}-\vec{w})^t (U^\perp)^t D_{\lambda^*} U^\perp (\vec{x}-\vec{w}) = \\
&(\vec{x}-\vec{w})^t U^t D U (\vec{x}-\vec{w}) + \tfrac{1}{\lambda^*} \cdot (\vec{x}-\vec{w})^t (U^\perp)^t U^\perp (\vec{x}-\vec{w}) = \\
&(\vec{x}-\vec{w})^t U^t D U (\vec{x}-\vec{w}) + \tfrac{1}{\lambda^*} \cdot ((\vec{x}-\vec{w})^t (\vec{x}-\vec{w}) - (\vec{x}-\vec{w})^t U^t U (\vec{x}-\vec{w}))
\end{aligned}
$$

where $U^\perp$ denotes an orthogonal projection to the space dual to U, and D_{λ^*} refers to the diagonal matrix with elements $(1/\lambda^*)$. (In the derivation, we use that $(U^\perp)^t U$ vanishes and the horizontal concatenation of U and $U^\perp$ is orthonormal.) This way, only k principal components need to be computed explicitly.

It has been demonstrated in [28] that this proposal performs well in practice. It would be interesting to see whether this proposal corresponds to a cost function as well. Thus, we consider the alternative metric (3.15). In the approach [28], this metric has been integrated into an interleaved PCA and NG scheme to obtain very promising results. We provide a theoretical justification for this procedure under the assumption that the average λ^* is small in comparison to the explicit eigenvalues in Λ.

Assume, as before, the cost function (3.4)

$$E_{\mathrm{NG}}^{disc}(\vec{w}, \Lambda) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} h_\sigma(k_i(\vec{x}^j)) \cdot d_{\Lambda_i}(\vec{x}^j, \vec{w}^i)$$

where $d_{\Lambda_i}(\vec{x}, \vec{w}^i)$ is determined by a low rank matrix with rank k, i.e. we reduce the degrees of freedom of Λ_i to symmetric positive definite matrices with determinant 1 which can be decomposed into k orthonormal directions with arbitrary scaling and $m-k$ directions with uniform scaling. In formulas, this means that the equation

$$\Lambda_i = \sum_{j=1}^{k} \alpha_j^i \cdot \vec{y}_i^j (\vec{y}_i^j)^t + \sum_{j=k+1}^{m} (\alpha^*)^i \cdot \vec{y}_i^j (\vec{y}_i^j)^t \tag{3.16}$$

holds with α_j^i, $(\alpha^*)^i > 0$ and orthonormal vectors $\vec{y}_i^j$. As before, we can investigate optimum parameters of the cost function for a batch optimization scheme, i.e. optimum assignments given fixed prototypes and matrix parameters, and, in turn, optimum prototypes $\vec{w}^i$, and optimum matrices Λ_i characterized by orthonormal directions $\vec{y}_i^j$ and scaling terms α_j^i and $(\alpha^*)^i$, given fixed assignments k_{ij}. If the respective parameters are set to the optimum values (or set in such a way that the cost function is decreased in every step), convergence is guaranteed which can be seen in the same way as before.

Obviously, optimum rank assignments are given by the same formula as for standard matrix NG, the same holds for optimum prototypes. Thus, it remains to compute optimum matrices under the constraints posed on Λ_i. This can be computed independently for every matrix Λ_i, thus we omit the index i for simplicity. Further, for simplicity, we set $\vec{z}^j := \sqrt{h_\sigma(k_{ij})} \cdot (\vec{w}^i - \vec{x}^j)$ and $C := \sum_j \vec{z}^j (\vec{z}^j)^t$ as the generalized Mahalanobis distance which includes neighborhood cooperation. We assume that the eigenvalues of C are pairwise different (which is fulfilled almost surely). Then, the potential for matrix $\Lambda = \Lambda_i$ has the form

$$E(\Lambda) := \sum_j (\vec{z}^j)^t \Lambda \vec{z}^j = \sum_{l \leq k} \alpha_l \cdot (\vec{y}^l)^t C \vec{y}^l + \sum_{l>k} \alpha^* \cdot (\vec{y}^l)^t C \vec{y}^l \tag{3.17}$$

which has to be minimized for orthonormal $\vec{y}^l$, and α_l, $\alpha^* > 0$ such that $\prod_l \alpha_l \cdot (\alpha^*)^{m-k} = 1$.

Step 1: The derivative (with respect to $\vec{y}^l$) of $E(\Lambda) + \sum_l \lambda_l ((\vec{y}^l)^t \vec{y}^l - 1)$ with Lagrange parameter λ_l enforcing the normality of the vectors yields $2\alpha_l C^t \vec{y}^l + 2\lambda_l \vec{y}^l$ for $l \leq k$ and $2\alpha^* C^t \vec{y}^l + 2\lambda_l \vec{y}^l$ for $l > k$. This is 0 for local optima, hence all vectors $\vec{y}^l$ are eigenvectors of C.

Step 2: Assume the eigenvalue of C for eigenvector $\vec{y}^l$ is $\lambda_{\vec{y}^l}$. We consider the derivative of $E(\Lambda) + \lambda \cdot \left(\left(\prod_{l' \leq k} \alpha_{l'} \right) \cdot (\alpha^*)^{m-k} - 1 \right)$ with Lagrange parameter λ enforcing determinant one with respect to α_l and α^*. The derivative with respect to α_l for

$l \leq k$ yields $(\vec{y}^l)^t C \vec{y}^l + \lambda \left(\prod_{l' \leq k, l' \neq l} \alpha_{l'} \right) \cdot (\alpha^*)^{m-k}$. This should be 0 for local optima. Hence, setting $K := -\lambda \left(\prod_{l' \leq k} \alpha_{l'} \right) \cdot (\alpha^*)^{m-k}$, we find $\alpha_l = K / \lambda_{\vec{y}^l}$. Similarly, the derivative with respect to α^* yields $\sum_{l'>k} (\vec{y}^{l'})^t C \vec{y}^{l'} + \lambda \left(\prod_{l' \leq k} \alpha_{l'} \right) \cdot (m-k) \cdot (\alpha^*)^{m-k-1}$. This should be 0 for local optima. Hence $\alpha^* = K / \left(\sum_{l'>k} \lambda_{\vec{y}^{l'}} / (m-k) \right)$. Because the determinant must yield 1, we find $K = \left(\left(\prod_{l \leq k} \lambda_{\vec{y}^l} \right) \cdot \left(\sum_{l'>k} \lambda_{\vec{y}^{l'}} / (m-k) \right)^{m-k} \right)^{1/m}$, i.e. up to normalization, the coefficient α_l is given by the inverse eigenvalue of $\vec{y}^l$ for $l \leq k$, and α^* is given by the inverse of the average eigenvalues of $\vec{y}^l$ for $l > k$.

Step 3: We want to show that the optima of $E(\Lambda)$ are obtained if $\vec{y}^l$ for $l \leq k$ refers to the k eigenvectors with largest eigenvalues of C under the assumption that the eigenvalues $\vec{y}^l$ are small for $l > k$. We will derive an explicit bound on the eigenvalues which guarantees this fact. Assume two eigenvectors $\vec{y}^p$ and $\vec{y}^q$ are chosen with $\lambda_{\vec{y}^p} > \lambda_{\vec{y}^q}$. We want to show that $E(\Lambda)$ is smaller if $p \leq k$ and $q > k$ in comparison to the value $E(\Lambda)$ which is obtained for $p > k$ and $q \leq k$ under assumptions on the size of the eigenvalues. Thereby, the other eigenvalues remain in the same position. By repetition, we can then conclude that the indices $l \leq k$ refer to the k largest eigenvalues of C. If we choose $\vec{y}^l$ as eigenvectors of C and α_l, α^* corresponding to the eigenvalues, $E(\Lambda)$ is proportional to the normalization constant K:

$$E(\Lambda) = m \cdot \left(\left(\prod_{l \leq k} \lambda_{\vec{y}^l} \right) \cdot \left(\frac{\sum_{l>k} \lambda_{\vec{y}^l}}{m-k} \right)^{m-k} \right)^{1/m}$$

This value is smaller if $p \leq k$ and $q > k$ than for the case $p > k$ and $q \leq k$ if and only if $\lambda_{\vec{y}^p} \left(\prod_{l \leq k, l \notin \{p,q\}} \lambda_{\vec{y}^l} \right) \cdot \left(\frac{\sum_{l>k, l \notin \{p,q\}} \lambda_{\vec{y}^l} + \lambda_{\vec{y}^q}}{m-k} \right)^{m-k} \leq \lambda_{\vec{y}^q} \left(\prod_{l \leq k, l \notin \{p,q\}} \lambda_{\vec{y}^l} \right) \cdot \left(\frac{\sum_{l>k, l \notin \{p,q\}} \lambda_{\vec{y}^l} + \lambda_{\vec{y}^p}}{m-k} \right)^{m-k}$. Note that, on both sides, the same remaining terms are contained in the summands and products, respectively, since we assume that the other eigenvalues decompose into the same two subsets when considering the largest and smallest eigenvalues, respectively. Hence this is equivalent to the inequality $\lambda_{\vec{y}^p} \cdot \left(\sum_{l>k, l \notin \{p,q\}} \lambda_{\vec{y}^l} + \lambda_{\vec{y}^q} \right)^{m-k} \leq \lambda_{\vec{y}^q} \cdot \left(\sum_{l>k, l \notin \{p,q\}} \lambda_{\vec{y}^l} + \lambda_{\vec{y}^p} \right)^{m-k}$ which is equivalent

to the inequality $\sum_{l>k,l\notin\{p,q\}} \lambda_{\vec{y}^l}(\lambda_{\vec{y}^p}^{1/(m-k)} - \lambda_{\vec{y}^q}^{1/(m-k)}) \leq \lambda_{\vec{y}^p}\lambda_{\vec{y}^q}^{1/(m-k)} - \lambda_{\vec{y}^q}\lambda_{\vec{y}^p}^{1/(m-k)}$ Hence, we obtain the inequality

$$\sum_{l>k,l\notin\{p,q\}} \lambda_{\vec{y}^l} \leq \frac{\lambda_{\vec{y}^p}(\lambda_{\vec{y}^q})^{1/(m-k)} - \lambda_{\vec{y}^q}(\lambda_{\vec{y}^p})^{1/(m-k)}}{(\lambda_{\vec{y}^p})^{1/(m-k)} - (\lambda_{\vec{y}^q})^{1/(m-k)}} \tag{3.18}$$

where we exclude p and q in the sum on the left hand side. The bound on the right side of (3.18) becomes large for $\lambda_{\vec{y}^p} \gg \lambda_{\vec{y}^q}$. Hence indices of the dominant eigenvectors are contained in the set $l \leq k$ for optimum solutions if the variance in the remaining directions is small enough. For intermediate eigenvectors, the optimum is not clear and depends on the situation at hand – however, it can be expected that the exact order of the intermediate eigenvectors has only a minor effect on the value of the cost function $E(\Lambda)$. In the particularly relevant case that k is chosen as the intrinsic dimensionality of the data cluster, the inequality (3.18) is usually fulfilled for all indices $p \leq k$ and $q > k$ if and only if the indices $l \leq k$ correspond to the k major eigendirections, since a huge gap is present between the value $\lambda_{\vec{y}^k}$ and $\lambda_{\vec{y}^{k+1}}$, the latter resulting only from noise.

Thus, if the inequality (3.18) holds for the chosen eigenvalue directions, convergence of the method is guaranteed. In this case, matrix learning can be accelerated: instead of a full matrix inversion, the major k eigenvectors and eigenvalues of the generalized correlation matrix are determined, leading to a low rank matrix $\Lambda = U^t \cdot D \cdot U$. The variation in the remaining directions is given by the term $\sum_j (\vec{z}^j(\vec{z}^j)^t - U\vec{z}^j(U\vec{z}^j)^t)/(m-k)$ where $\vec{z}^j = \sqrt{h_\sigma(k_{ij})} \cdot (\vec{w}^i - \vec{x}^j)$ as before. After normalization such that the determinant is 1, distances can be directly computed.

CHAPTER 4

Matrix clustering for pattern recognition

4.1 Recognition and clustering

We test the algorithm for several illustrative two dimensional examples and benchmarks from the UCI repository [2] as well as an application for image compression. Note that the main focus of our work is on the theoretical link of a cost function, matrix learning, and local PCA, and a corresponding proof of convergence. We add experiments for the batch learning rules merely as a proof of concept which demonstrates the usefulness of this method (or alternatives) for matrix adaptation. For all experiments, initial neighborhood ranges between $n/2$ (for NG) and $n/12$ (for SOM) are used and the neighborhood is multiplicatively annealed to 0 during training. (These choices are borrowed from rules of thumb often used for standard SOM and NG, respectively, and worked well in practice, see [25].) Training takes place for 100 epochs since this number turned out to be sufficient for robust convergence, where, as before, one epoch refers to one loop of matrix NG (one iteration of assigning the k_{ij}, $\vec{w}^i$, and Λ_i). For performance measures, the average results over 10 repetitions are reported with different prototype initializations choosing small random numbers. Note that class labels are not used to train SOM and NG, such that an evaluation on the full data set is possible without bias from overfitting methods. We train standard k-means, neural gas, and SOM with two-dimensional rectangular neighborhood structure with and without full matrix adaptation. Note that the choice of the number of clusters n constitutes a

critical parameter for matrix clustering just as for standard k-means, SOM, and NG. In general, n should be chosen to obtain a good compromise between a sparse model (small n) and sufficient capacity to represent the underlying probability distribution (large n). Since training takes place in an unsupervised way, overfitting for large n is usually not observed. Methods to automatically determine the number of clusters based on the optimization of criteria which formally evaluate this compromise have been proposed e.g. in [61, 62]. These methods could be extended to matrix clustering. However, since the focus of this work is a formal investigation of the basic algorithm and a connection to a cost function, this issue is beyond the scope of this thesis. In the experiments, we picked a reasonable number n of clusters according to the given data set, which is experimentally determined such that adding further prototypes does not or only slightly increase the accuracy of the results. For the artificial data sets, the number n was chosen according to the underlying distribution.

Gaussian clusters

The first dataset consists of 4 two-dimensional ellipsoidal clusters as shown in Fig. 4.1. Thereby, each cluster consists of 500 points whereby the ratio between the first and second principal component is 16:1, as can be seen in the figure. We train NG, SOM, and k-means with full matrix adaptation and four prototypes. The final results are the same for all methods: after about 30 epochs, convergence can be observed and the found clusters well resemble the data clusters. For every step, we depict the major principal component of the data correlation matrix of the receptive field and the minor principal component of the matrix Λ_i assigned to the prototypes. As already mentioned, these directions should become identical during training and, thus, implicit local PCA is performed by matrix NG. One can observe that the directions deviate at the beginning of training due to the influence of the neighborhood which causes an averaging of matri-

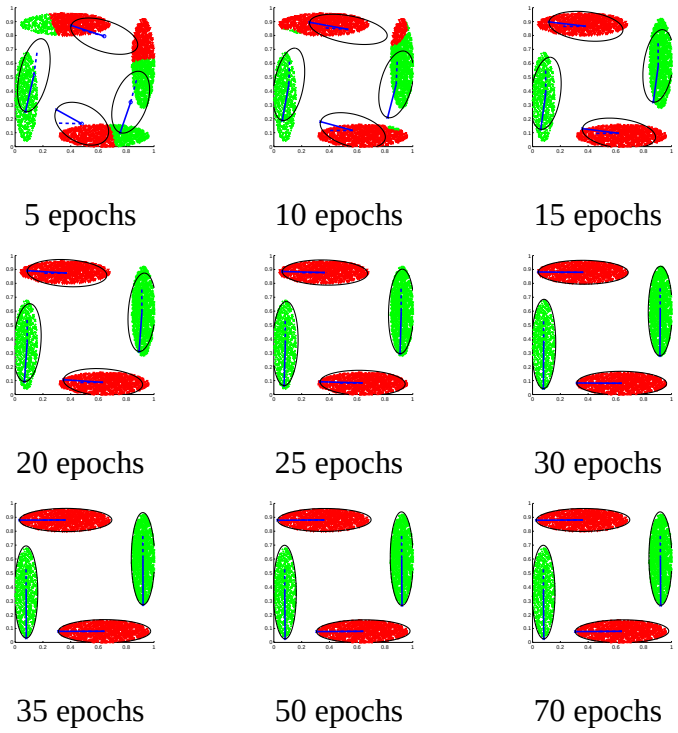


Figure 4.1: The resulting prototypes and eigenvalues of the matrix for NG and an illustrative two-dimensional data set. The solid line depicts the minor principal component of the found matrix, the dashed line gives the main principal component of the data in the receptive field. Ellipsoids visualize the resulting cluster shapes. Colors indicate cluster assignments.

ces. At the end, perfect agreement can be observed. One can observe that elongated clusters can be taken into account and the matrices follow the ellipsoidal cluster shapes in the data.

Spiral clustering

The clustering of spiral data set is depicted in Fig. 4.2. It consists of 2000 points arranged on a spiral with small bandwidth as depicted in the figure. It is trained by matrix

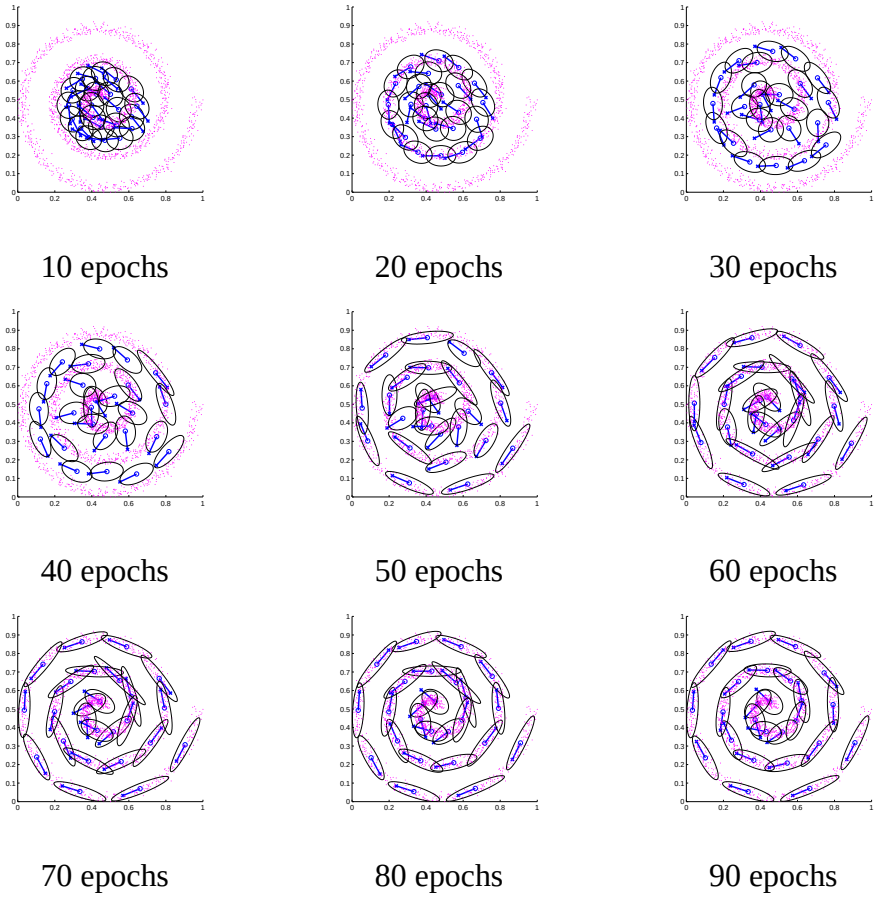
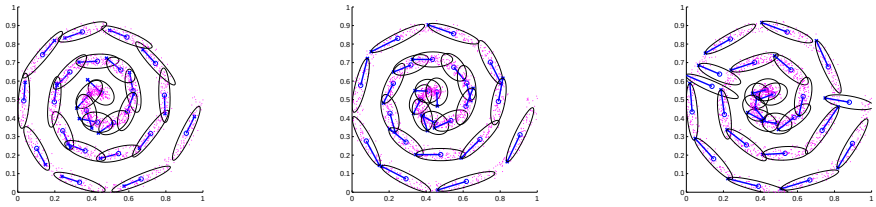


Figure 4.2: Matrix NG for the spirals data set. Convergence of the prototypes can be observed after about 60 epochs.

NG using 25 prototypes. As before, convergence can be observed after few steps. The prototypes are located on the spiral curve and the cluster shapes are adapted to the spiral, showing elongated ellipses at the outer regions and almost circular behavior in the middle. For matrix SOM, the same behavior can be observed, whereas matrix k-means suffers from local optima. Fig. 4.3 displays the superior result obtained by neighborhood integration for matrix SOM and NG.



matrix NG matrix SOM matrix k-means

Figure 4.3: Results of matrix NG, SOM, and k-means after 100 epochs for the spirals data set. Obviously, matrix k-means suffers from local optima.

Spirals classification

The classification of two spirals data set is depicted in Fig. 4.4. It consists of two spirals with 1000 points each. It is trained using 34 prototypes with matrix learning and standard k-means, SOM, and NG, respectively. The results show that, for matrix learning the cluster shapes are adapted to the spiral, showing elongated ellipses at outer regions and circular shapes in the middle, while standard clustering without matrix adaptation cannot adjust the cluster forms to the given data as can be seen in Fig. 4.4 (bottom line). For both cases, the results of SOM and NG are virtually indistinguishable, whereas k-means suffers from local optima. We evaluate the results by their ability to classify the points correctly into two classes according to the two spirals. Thereby, we use posterior labeling of prototypes based on a majority vote. The classification accuracy is shown in Fig. 4.4, demonstrating the superior behavior of matrix learning and neighborhood integration.

Checkerboard

In the checkerboard data set, data are arranged according to a checkerboard structure with almost circular and elongated clusters, respectively, as depicted in Fig. 4.5 (top

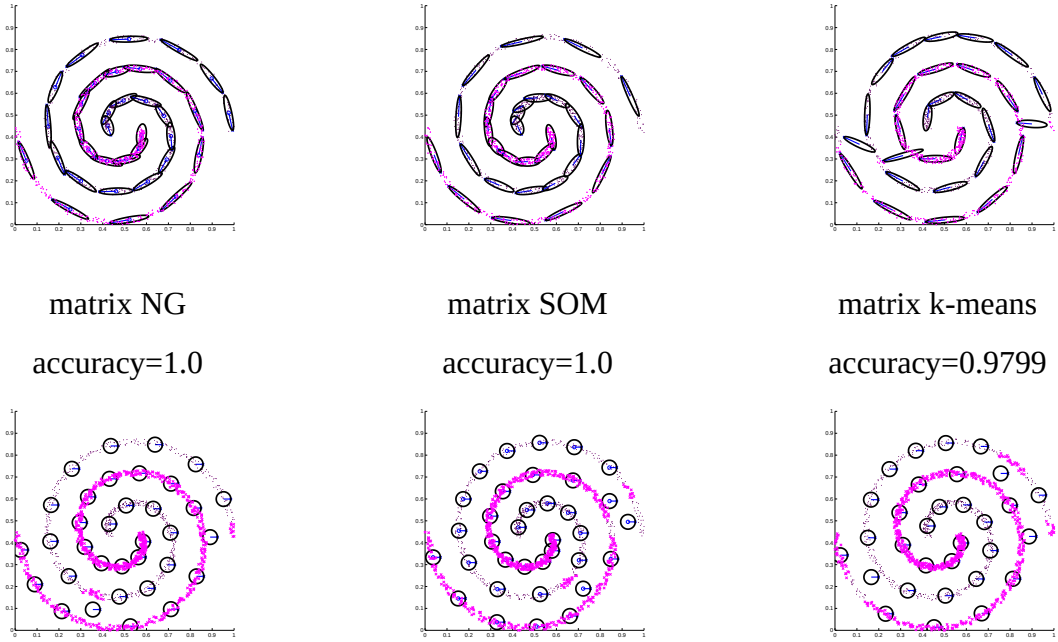
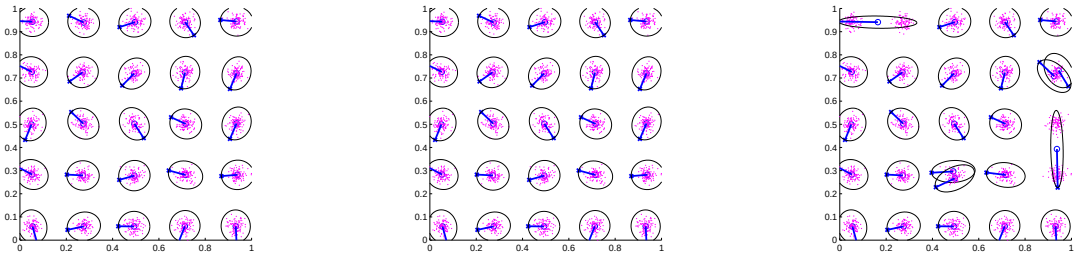


Figure 4.4: Results of matrix NG, SOM, and k-means (top) and standard NG, SOM, and k-means (bottom) after 100 epochs for the spirals data set. Obviously, k-means suffers from local optima. Further, the shape is better represented by matrix clustering as can be seen by the classification accuracy of the maps.

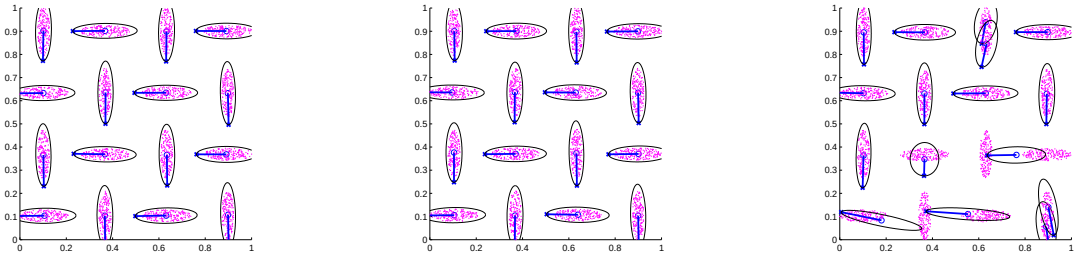
and bottom, respectively). 25 clusters with 100 points each, respectively 16 clusters with 200 points each are chosen with isotropic normal distribution, respectively a ratio of the variance of the first and second principal component of the clusters as 16:1.

25 respectively 16 prototypes are adapted using matrix NG, SOM, and k-means. In addition, we depict a characteristic isobar for every prototypes. Obviously, NG and SOM yield excellent and robust results, whereas k-means suffers from local optima. In all cases, the major axes of the found ellipsoids corresponds to the local main principal

component.



matrix NG matrix SOM matrix k-means



matrix NG matrix SOM matrix k-means

Figure 4.5: Multimodal checkerboard data with circular shapes (top) and elongated clusters (bottom), respectively. The results of matrix k-means, SOM, and NG after 100 epochs are depicted.

UCI classification data

We consider three benchmark datasets from the UCI machine learning repository [2]:

- Iris: The popular iris data set consists of 150 data points assigned to 3 classes. Data are numerical with 4 dimensions.
- The Wisconsin diagnostic breast cancer data set contains 569 data of 2 classes represented by 32 dimensional vectors.
- The ionosphere data contains 351 examples in 2 classes, data are given as 34 dimensional vectors.

Class labels are available for the data sets, i.e. we can assign a label c_i to every data point $\vec{x}^i$ in the data. Note, however, that k-means, NG, and SOM and their matrix variants do not use the prior class information for training, rather, learning takes place in a fully unsupervised way. We evaluate the methods by the ability to group the data into the priorly given classes. For this purpose, we consider two different evaluation measures:

- *Cluster coherence:* Denote by $I(\vec{x}^i)$ the cluster given by a learned clustering, i.e. the number of the winning prototype. Then, we use the evaluation measure

$$C_1 := \sum_{i=1}^{p} \sum_{j=i+1}^{p} \frac{\delta\left(\delta(c_i = c_j) - \delta(I(\vec{x}^i) = I(\vec{x}^j))\right)}{0.5 \cdot p(p-1)}$$

  where δ is the indicator function $\delta(0) = 1$ and $\delta(t) = 0$ for $t \neq 0$, and the terms $c_i = c_j$ and $I(\vec{x}^i) = I(\vec{x}^j)$ refer to the question whether the labels c_i and c_j or the winner prototypes of $\vec{x}^i$ and $\vec{x}^j$, respectively, are identical. Obviously, C_1 measures the homogeneity of clusters with respect to the given classification. This measure is particularly suited if we use the same number of clusters as given classes.

- *Classification accuracy:* Given prototypes, we assign a label $C(\vec{w}^j)$ to every prototype $\vec{w}^j$ which is given by the majority of the labels of data points in the receptive field of $\vec{w}^j$. For a data point $\vec{x}^i$, the label of the winner is $C(\vec{w}^{I(\vec{x}^i)})$, where $I(\vec{x}^i)$ denotes the index of the winner, as before. The classification accuracy of this function is given by

$$C_2 := 1 - \sum_{i=1}^{p} \delta\left(c_i = C(\vec{w}^{I(\vec{x}^i)})\right)/p$$

  This measure can be used for any number of prototypes and classes.

For comparison, we consider the supervised clustering which is obtained from the class-wise Mahalanobis distance; this means, the same number of clusters as given classes is considered, its prototypes are set to the centres of the classes, and the matrix is set to the standard Mahalanobis distance of the respective class, i.e. the inverse correlation matrix of the respective class. Note that the class labels have to be known for this procedure. The model corresponds to a Gaussian mixture model with unimodal classes for known class centers and correlation matrices. We refer to the result as direct Mahalanobis. Note that neither NG nor SOM has been designed for classification purposes, such that the results can only serve as an indicator of the capacity of matrix-enhanced learning. However, most evaluation measures which are based on unsupervised information only (such as e.g. quantization error, trustworthiness, ...) depend on the metric of data points, thus a fair comparison of Euclidean and non-Euclidean clustering using these alternatives is not possible.

The results (error measure C_1 and C_2 and the standard deviation) of standard NG, SOM, and k-means and NG, SOM, and k-means with matrix learning are given in Tab. 4.1, whereby we use the same number of cluster centres as given classes. Standard clustering yields isotropic clusters whereas matrix learning adapts ellipsoidal shapes according to the data. Obviously, neighborhood integration improves the robustness

and classification ability of the method in most cases. Further, matrix adaptation accounts for an improvement of 1-9% depending on the data set due to the larger flexibility of the metric. Thus, in general, neighborhood integration to obtain better regularization and matrix learning to obtain greater flexibility of the models is advisable. This corresponds to best results for matrix NG in all but one case (ionosphere), where, seemingly, the neighborhood structure of the data is misleading for the final result and, in consequence, k-means without neighborhood integration performs slightly better. In all cases, matrix learning improves the result which can be accounted for by the better capability of the model of following the underlying probability distribution with less prototypes. Interestingly, the result of a direct Mahalanobis distance is worse compared to matrix clustering in breast cancer and ionosphere data sets, although the method uses the given class information and the same number of prototypes as for matrix clustering is available. A potential reason for this is that noise in the data set disrupts the relevant information. Unlike the direct Mahalanobis method, matrix clustering uses only those data for matrix computation which lie in the receptive field of the prototype, i.e. the geometric form is taken into account instead of the prior labeling, thus, better robustness with respect to outliers is obtained.

Naturally, the direct Mahalanobis method cannot be used in the case of unknown prior labeling or in the case of multimodal classes which should be represented by more than one prototype. Matrix clustering can directly be used in these settings.

coherence C_1	matrix k-Means	matrix SOM	matrix NG	direct Mahalanobis
iris	0.8877 ± 0.0885	0.8917 ± 0.0838	0.9009 ± 0.0778	0.9740
breast cancer	0.8119 ± 0.0553	0.8243 ± 0.0407	0.8445 ± 0.0345	0.8194
ionosphere	0.6076 ± 0.0916	0.5969 ± 0.0647	0.6083 ± 0.0543	0.5323
	k-Means	SOM	NG	
iris	0.8446 ± 0.0521	0.8591 ± 0.0414	0.8737 ± 0.0000	
breast cancer	0.7504 ± 0.0000	0.7504 ± 0.0000	0.7504 ± 0.0000	
ionosphere	0.5871 ± 0.0068	0.5882 ± 0.0011	0.5868 ± 0.0008	
accuracy C_2	matrix k-Means	matrix SOM	matrix NG	direct Mahalanobis
iris	0.8606 ± 0.1356	0.8909 ± 0.1175	0.9147 ± 0.0847	0.9800
breast cancer	0.8953 ± 0.0226	0.9024 ± 0.0245	0.9135 ± 0.0192	0.8998
ionosphere	0.7320 ± 0.0859	0.7226 ± 0.0673	0.7197 ± 0.0600	0.6410
	k-Means	SOM	NG	
iris	0.8499 ± 0.0864	0.8385 ± 0.0945	0.8867 ± 0.0000	
breast cancer	0.8541 ± 0.0000	0.8541 ± 0.0000	0.8541 ± 0.0000	
ionosphere	0.7074 ± 0.0159	0.7114 ± 0.0013	0.7097 ± 0.0009	

Table 4.1: Classification results of the clustering methods with and without matrix adaptation for various data sets from UCI. The mean heterogeneity of clusters measured by C and the standard deviation are reported. Note that differences of clustering with and without matrix adaptation are mostly fairly large, while the situation is less clear when comparing the methods.

4.2 Image compression

Vector quantization on the one hand and principal component analysis on the other hand constitute standard methods for lossy image compression [22, 6]. For vector quantization, images are decomposed into blocks of size $m = m_1 \times m_1$ and a standard vector quantization using n prototypes in $\mathbb{R}^m$ is applied to this data. Afterwards, every block of the image is substituted by the coordinates of the winning prototype, i.e. every block can be encoded using only $\log_2(n)$ bits as a reference to the corresponding prototype. Since the space to represent the prototype vectors is constant and independent of the size of the image, an average of $\log_2(n)/m$ bits per pixel (bpp) are needed for this compression scheme, corresponding to the encoding of the winning prototype per pixel divided by the number of pixels

Principal component techniques constitute transformation coding techniques of images, since they represent pixels by means of a parameterized function. Again, images are decomposed into blocks of size $m = m_1 \times m_1$ and a standard PCA is applied to these m-dimensional data vectors. The main k principal components represent the transformation. Every block is then represented by k coefficients (these numbers are stored using T bits, in our case $T = 8$). The image is reconstructed by taking the sum of the k principal component directions weighted by these coefficients instead of the full information. Since the principal components can be stored independently of the image size (they depend on the block size which we assume to be fixed), this compression method leads to $k \cdot T/m$ bpp corresponding to the representation of the scaling of k princicipal components per pixel divided by the number of pixels.

Naturally, a global principal component can be substituted by local PCA techniques which arise from combined vector quantization and PCA techniques. This way, every block is represented by k PCA coefficients and a reference to the winning prototype. Reconstruction takes place as the weighted sum of the main principle components at-

Lena	4.25 bpp	2.25 bpp	1.25 bpp
NG (train)	39.4967±0.0786	37.0219±0.0297	30.0105±0.0382
(test)	36.0334±0.3119	34.6203±0.0248	29.8789±0.2107
PCA (train)	37.4612±0.0000	31.5699±0.0000	28.4870±0.0000
(test)	38.1723±0.0000	32.8029±0.0000	29.8617±0.0000
VQPCA (train)	39.6026±0.3241	33.3710±0.1680	29.7557±0.0830
(test)	40.2752±0.4355	34.2123±0.2086	29.9780±0.1115
MoPPCA (train)	38.2034±0.1278	32.6634±0.1248	29.6081±0.0608
(test)	39.6010±0.4374	33.5733±0.2238	30.1682±0.2223
matrix NG (train)	39.3886±0.1177	32.8789±0.0408	29.3469±0.0512
(test)	40.3762±0.2122	34.2653±0.1468	30.4726±0.1246

Table 4.2: Results of image compression using different compression methods and parameters (as reported in the text) for the Lena image. The mean PSNR and its standard deviation are depicted.

tached to the prototype translated by the class center. This way, $(k \cdot T + \log_2(n))/m$ bpp are needed for the compression scheme correspondigng to the winning prototype and the coefficients of the main k principal components per pixel divided by the number of pixels.

Note that these bpp values ($\log_2(n)/m$, $k \cdot T/m$ bpp, and $(k \cdot T + \log_2(n))/m$ bpp, respectively) are valid for the limit of large images only. Strictly speaking, the ratio should also take into account the size which is necessary to represent the compressor and possibly also the decompression method. However, we can assume that the latter is universal, thus we drop it. Further, the size of the compressor is given by the terms kTm, nTm, and $(k+1)nTm$, respectively, which can in addition be compressed using e.g. standard Huffman encoding, and which is divided by the image size to give its contribution to the bpp value. For an image of size 512×512, as considered below,

and the parameter values as chosen below, it accounts for less than 2 % of the bpp value.

We evaluate matrix clustering for image compression in comparison to a simple PCA scheme with only one set of principle components (referred to as PCA), a standard vector quantization scheme, in this case NG without matrix adaptation (referred to as NG), and to alternative transform coding schemes proposed in [38] and [19]. Note that we do not use matrix SOM, for comparison, since it is restricted to the prior topological structure and hence yields slightly worse results than matrix NG with data optimum topological interaction. The method proposed in [19] (referrred to as VQPCA) directly combines vector quantization and PCA in a heuristic way. [38] proposes a probabilistic background for an interleaved PCA and vector quantization by means of a mixture of Gaussians (referred to as MoPPCA). We use a direct Matlab implementation of VQPCA, and we use the code provided in [24] for MoPPCA.

Evaluation of all compression schemes is done by the peak signal-to-noise ratio (PSNR) which is defined as follows: Assume x_i is the value of the original image at a certain position and x_i' the reconstructed value. Assume the image consists of N pixels. The mean squared error (MSE) is defined as $\sum_i \|x_i - x_i'\|^2/N$. The PSNR consists of the scaled MSE

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}} \right)$$

where MAX denotes the maximum possible pixel value of the image (e.g. it is 255 for images represented by 8 bits per pixel.) Obviously, the higher PSNR, the better. We evaluate the PSNR of the left half of the image which is used for training, and, in addition, we report the PSNR of the right half of the image which is solely used for testing to judge the generalization ability of the models. Both values are reported together with the standard deviation.

The results for three different images (Lena, House, and Church) and different bpp

House	4.25 bpp	2.25 bpp	1.25 bpp
NG (train)	41.6597±0.3472	41.4012±0.1252	33.9519±0.1021
(test)	36.8280±0.6140	35.8252±0.1290	30.5782±0.1601
PCA (train)	43.9537±0.0000	37.3119±0.0000	33.3293±0.0000
(test)	40.5958±0.0000	34.2860±0.0000	30.4564±0.0000
VQPCA (train)	45.6874±0.3366	39.3378±0.1088	35.3971±0.1180
(test)	41.9069±0.2468	35.8355±0.0957	31.7602±0.2207
MoPPCA (train)	44.3200±0.3305	38.5901±0.2378	34.7363±0.1742
(test)	41.1701±0.1731	35.2210±0.4565	31.330±0.1499
matrix NG (train)	45.2712±0.1515	39.0032±0.0926	34.5012±0.1394
(test)	42.0525±0.2625	35.9050±0.1391	31.5334±0.0471

Table 4.3: Results of image compression using different compression methods and parameters (as reported in the text) for the House image. The mean PSNR and its standard deviation are depicted.

rates can be found in Tab. 4.2- 4.4. We choose parameters of the algorithms such that the bpp rate becomes comparable (4.25, 2.25, and 1.25 bpp, respectively). The parameters have been chosen such that a reasonably good performance is obtained (full parameter optimization has not been done due to the quite large computation time.) The parameters are:

- vector quantization based compression using simple neural gas: number of prototypes $n = 19$, block size $m = 1$ (for $4.2479 \approx 4.25$ bpp), $n = 512$, $m = 4$ (for 2.25 bpp), and $n = 32$, $m = 4$ (for 1.25 bpp)
- image compression by simple PCA: block size $m = 64$, number of principal components $k = 34$ (for 4.25 bpp), $m = 64$, $k = 18$ (for 2.25 bpp), $m = 64$, $k = 10$ (for 1.25 bpp)

Church	4.25 bpp	2.25 bpp	1.25 bpp
NG (train)	35.3130±0.4311	31.0077±0.1002	25.4105±0.0419
(test)	35.5276±0.2295	28.3431±0.0631	23.4840±0.0316
PCA (train)	29.9463±0.0000	25.4431±0.0000	23.1356±0.0000
(test)	25.8948±0.0000	22.2513±0.0000	20.7304±0.0000
VQPCA (train)	32.6274±0.0950	27.4652±0.0411	24.6618±0.1747
(test)	28.1356±0.0602	23.8068±0.0594	21.7557±0.1484
MoPPCA (train)	31.8378±0.1382	27.1528±0.0441	24.3518±0.0575
(test)	27.5666±0.1279	23.5823±0.0325	21.6546±0.0273
matrix NG (train)	32.6134±0.0347	27.5124±0.0208	24.2528±0.0837
(test)	28.2460±0.0141	23.9505±0.0299	21.8169±0.0088

Table 4.4: Results of image compression using different compression methods and parameters (as reported in the text) for the Church image. The mean PSNR and its standard deviation are depicted.

- image compression by combined vector quantization and PCA: number of prototypes $n = 16$, block size $m = 16$, number of principal components $k = 8$ (for 4.25 bpp), $n = 16$, $m = 16$, $k = 4$ (for 2.25 bpp), $n = 16$, $m = 16$, $k = 2$ (for 1.25 bpp)

Image compression by local PCA methods is in our results consistently superior to global PCA. When using the same bpp rate, simple VQ encoding yields in some cases a better PSNR than local PCA methods, which is mostly due to the fact that local PCA methods require a comparably high number of bits for the representation of the PCA coefficient (8 bits each). This could certainly be reduced. Interestingly, however, the PSNR rate for image compression using local PCA methods is in many cases competitive to VQ based encoding or even superior, although the latter uses

Figure 4.6: Lena image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper right), global PCA (lower left), and local PCA by means of matrix NG (lower right), respectively.

smaller window sizes and more prototypes for representation. Interestingly, the three methods considered for compression by means of local PCA seem largely comparable. Thereby, the probabilistic model based on mixtures of Gaussians gives slightly worse PSNR, which might be attributed to the different (probabilistic) cost function instead of (some form of) the standard quantization error. Note, however, that we did not perform a full parameter optimization of n, m, and k for each method and each bpp, which could affect the conclusions.

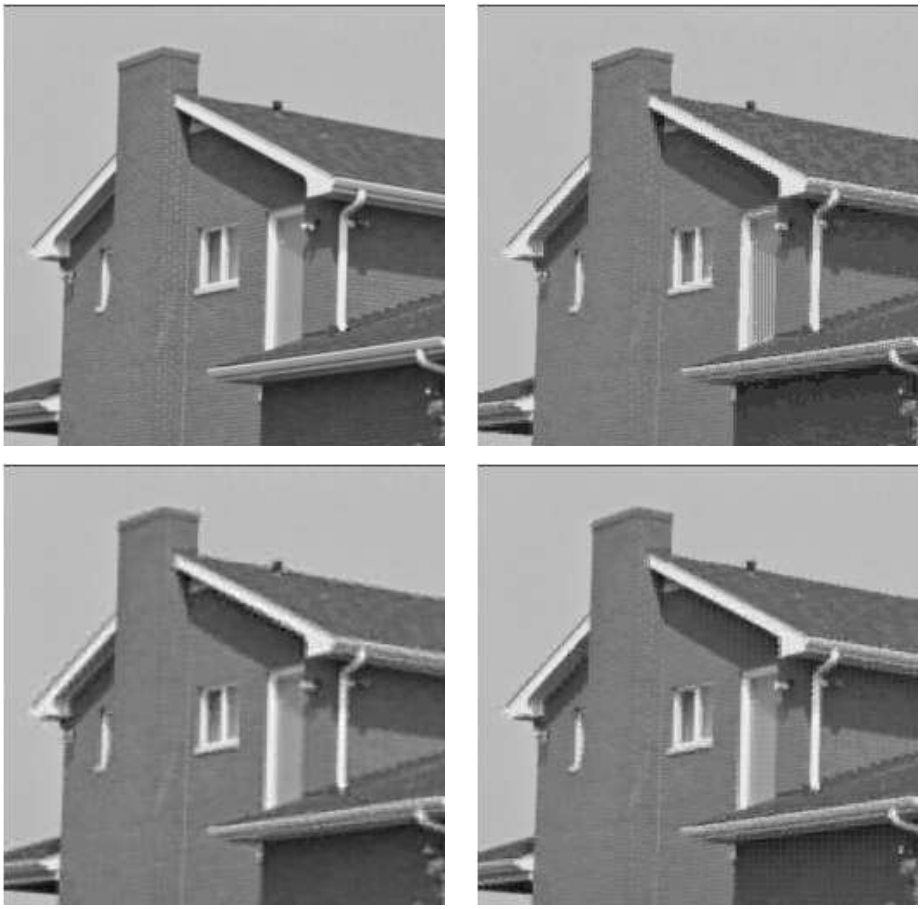


Figure 4.7: House image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper right), global PCA (lower left), and local PCA by means of matrix NG (lower right), respectively.

We show the results of the different image compression methods in Figs. 4.6, 4.7, 4.8 for 1.25 bpp. While this low bpp rate does not give satisfactory image reconstruction, as expected, the principal characteristics of the compression methods becomes obvious at this extreme setting: vector quantization based image compression leads to 'pixelized' representations, whereas compression based on global PCA yields a blurred impression. Local PCA averages between these two extremes. This seems particularly appropriate for regular textures such as the brick wall or roof of the House

Figure 4.8: Church image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper right), global PCA (lower left), and local PCA by means of matrix NG (lower right), respectively.

image. It seems less appropriate for the representation of parts of the images which are very detailled and which show a large variance such as Lena's face. Here, a compression based on small windows as taken for standard vector quantization gives a better reconstruction.

The three local PCA methods are very similar with respect to the global impression, however, when looking at image details, a few characteristic differences can be observed. In Figs. 4.9, 4.10, a detail of the Lena and House images, respectively, is

Figure 4.9: Detail of Lena image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper middle), global PCA (upper right), and local PCA by means of VQPCA (lower left), MoPPCA (lower middle) and matrix NG (lower right), respectively.

shown for a 1.25 bpp compression. Again, one can clearly observe the pixelized compression of standard vector quantization and the blurring of global PCA. Compared to MoPPCA and VQPCA, compression by local PCA using matrix NG seems particularly suited to preserve edges in the images, as one can see from Lena's hat and the door of the house, respectively. Overall, compression by matrix NG seems particularly suited for images which display pronounced textures and edges but not very small detail. Note, however, that this application on image compression application is merely

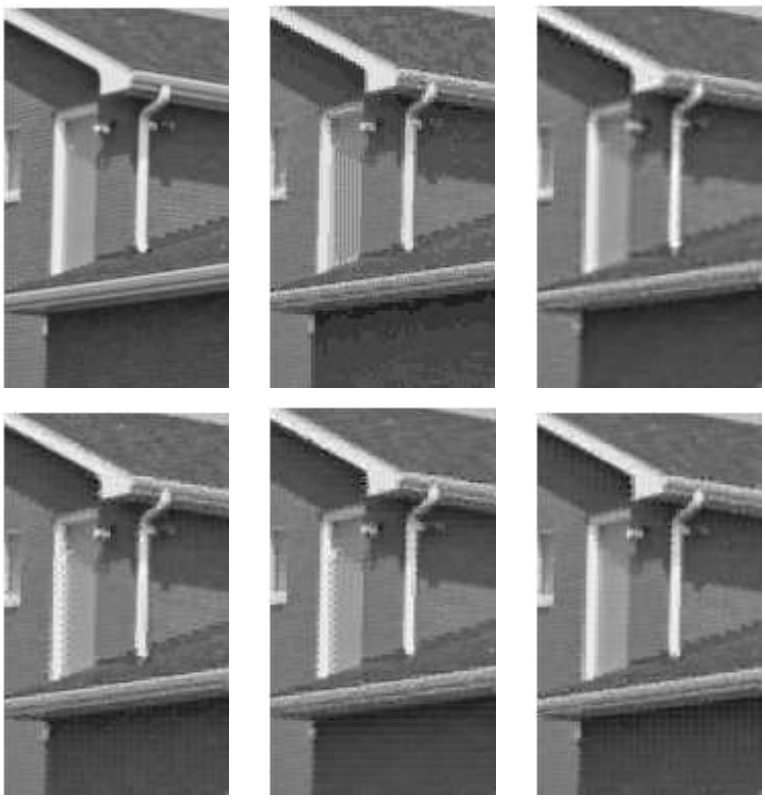


Figure 4.10: Detail of House image: original (upper left), and compressed image using 1.25 bpp and neural gas (upper middle), global PCA (upper right), and local PCA by means of VQPCA (lower left), MoPPCA (lower middle) and matrix NG (lower right), respectively.

a proof of concept, the main focus of the work being the convergence proof and the relation of matrix learning and cost functions to local PCA. It is not claimed that the proposed methods can compete with dedicated modern image compression techniques in this field.

4.3 Chapter summary

We considered an extension of neural based clustering such as NG and SOM towards a full matrix which is directly based on standard cost functions of these methods. With matrix adaptation, the SOM/NG can more easily represent scaling and correlation of dimensions with the local metric. The result leads to a generalized Mahalanobis distance and resembles approaches as proposed in the literature. Thus, a formal connection of known techniques to an underlying cost function results. Based on batch optimization, an adaptation scheme for matrix learning has been proposed and convergence of the method has been proved. As demonstrated in experiments, matrix NG can improve the clustering performance compared to versions which rely on isotropic cluster shapes. The method has been demonstrated on several benchmark examples.

The found results support work to derive local PCA methods in previous approaches such as [28]. Unlike these previous methods, the presented approach relies on a single cost function and convergence is guaranteed. This guarantee can be transferred to efficient local PCA methods as proposed in the literature if they arrive at similar iterative update schemes such as [28]. As demonstrated in the experiments, the convergence behavior of the method is very good in practical applications. Note that, once the connection of local PCA methods to matrix learning is formally established, the use of heuristics for the optimization steps in batch learning is justified, as long as they improve the respective cost terms. One possibility are online optimization methods which directly optimize the global cost function e.g. by means of a stochastic gradient descent. Alternatively, heuristics such as iterative PCA methods as presented in [28] can be included. These methods seem particularly important if the adaptation of full rank matrices is infeasible due to the data dimensionality. In this case, a restriction to a low rank adaptive approximation can be used. This can be formally justified by means of the same cost function as matrix NG if the eigenvalues in the remaining directions

are small.

Note that SOM and NG have been proposed as topographic maps and the notion of topology preservation has been well defined in this context, see e.g. [63]. The situation is not clear if adaptive matrices are considered since, this way, the metric of the data space is changed during training. It remains a subject of future research to specify in which sense topology preservation can be formalized in this setting.

CHAPTER 5

Matrix clustering for manifold learning

The amount of electronic data available today doubles roughly every 20 months, and the explosion of electronic information has reached almost every aspect of commercial and daily life including, for example, technical applications, robotics, health care, e-commerce, electronic communication, data bases, and the web. At the same time, better and better sensors and preprocessing methods as well as better storage and compression techniques have lead to an increase of the information contained in one data point, i.e. its dimensionality. By now, it is common to deal with data which displays several hundred or thousand dimensions such as data stemming from mass spectrometry, microarrays, medical images, and the like. The sheer amount of data makes it impossible for humans to manually browse through the data. At the same time, visualization becomes more and more problematic due to the high dimensionality, i.e. single dimensions hardly carry any information at all. This development has lead to an increasing interest in general data inspection and visualization methods which provide generic tools to quickly inspect data visually on a computer screen.

A multitude of data organization schemes such as clustering or topographic mapping have emerged, one of the most popular methods being the self-organizing map as proposed by Kohonen. Further, powerful high-quality data visualization schemes have been developed, and an open source MATLAB toolbox for dimensionality reduction as well as an extensive comparison of popular methods for artificial and real life data sets became available just recently [24, 65, 77]. Due to the remarkable and efficient vi-

sual perception abilities of humans, important aspects of the data like cluster structure, outliers, global form and extension can be efficiently inspected this way.

Depending on the task at hand, different aspects of data are relevant. As an example, it might be the case that a data point is judged as an outlier in one situation (e.g. a simple mistype) while carrying important information in another (e.g. representing a rare disease in a medical data set). Hence, data clustering, inspection and visualization constitute inherently ill posed problems and, as a consequence, a variety of different methods has evolved based on different principles which shall be preserved when processing data. A very popular method which achieves both, data clustering and visualization at the same time, is offered by the self-organizing map (SOM) as proposed by Kohonen [20]. Data are represented by means of prototypes which are arranged on a regular low-dimensional lattice such that topological aspects of the lattice structure capture topological aspects of the underlying data manifold. In this respect, SOM generalizes the popular k-means clustering [9] which aims at minimization of the standard quantization error. Neural gas (NG) as introduced by Martinetz constitutes a compromise of these methods [25], since it infers a data optimum topology during training in addition to vector quantization, which can possibly no longer be visualized directly. A variety of alternative clustering methods exist such as probabilistic models or fuzzy variants, see e.g. [3, 15, 41].

5.1 Extension of matrix clustering to data visualization

The extension of NG and SOM towards an adaptive metric offers the opportunity to directly use these models for advanced data processing tasks. A focus of this chapter will be the incorporation of matrix learning into global nonlinear data visualization methods. Data visualization is the process of transforming information into a graphical representation allowing the user to perceive and interact with the information. Since

humans often have difficulty comprehending data in many dimensions, we assume that the data of interest lies on an embedded nonlinear manifold within the higher-dimensional observed space. If the manifold is of low enough dimension then the data of interest can be visualised in the low dimensional space. Though data like image or other scientific measurement are often high-dimensional, the relevant number of the features is not as high as we observe. Reducing the dimensionality of the original data until features emerge is the process of identifying and removing as much as possible the irrelevant and redundant information. It aims at choosing a small subset of attributes that is sufficient to describe the original data set. Thus, dimension reduction can be used to improve the efficiency of visualization of large, multidimensional data sets.

Another need of scientific visualization stems from the limitation of the human vision system. Human vision is physically restricted to three dimensional displays. Therefore, in practice, pre-processing original data to two or three dimensions is necessary in order to represent the characteristics to the human observer. In general, exploratory data analysis and information visualization seek for an approximate perception of abstract data which can include specific features embedded in high dimensionality. Dimensionality reduction techniques can be used in such work to project the high dimensional data onto lower dimensional representations. Below is a summary of some of the important algorithms from the history of manifold learning and nonlinear dimensionality reduction.

Popular methods include, for example, globally linear methods which preserve as much statistical information as possible of the data such as (unsupervised) principal component analysis (PCA) and (supervised) linear discriminant analysis, both methods still representing astonishingly powerful tools with widespread applications [65]. The principle of distance preservation is behind nonlinear models such as mul-

tidimensional scaling (MDS), Sammon's nonlinear mapping (NLM), and curvilinear component analysis (CCA) [76, 33, 72]. These methods employ the Euclidean metric as distance measure. Other methods use the same projection principle such as Isomap, GeoNLM, and curvilinear distance analysis (CDA) [68, 73, 74, 75], but they are based on the true geodesic distances of the underlying manifold and usually provide a more reliable visualization than Euclidean based methods.

An alternative objective is to preserve local characteristics of the data manifold as accurately as possible such as the local data topology, a principle underlying the popular self-organizing map (SOM) [20], or the local linear relationships which can be observed in the data, as is the case for locally linear embedding (LLE) [66]. These methods, however, only map the given data points to a low-dimensional representation and out-of-sample extensions require additional effort since no explicit mapping from the embedded high dimensional space to low dimensions is constructed. Further, drawbacks for certain data sets can be observed depending on the used technology as extensively discussed in [65].

These facts provide a justification for the desire to design further models which provide an explicit map function rather than only the mapped data points. Locally linear coordination (LLC) and manifold charting (MC) as proposed in [67, 64] constitute two very interesting nonlinear approaches in this direction. Both methods rely on the notion of a smooth manifold which can locally be approximated by simple linear functions. For a global visualization, these local representations have to be glued together such that the coordination of the pieces fits the manifold structure. LLC relies on the ideas of LLE and assembles the pieces such that the local relationships of the data manifold are preserved, while manifold charting glues the pieces together such that they coincide on overlapping pieces. Interestingly, both objectives can be formalized in such a way that a global optimum can be computed directly by means of an eigen-

vector decomposition of appropriate matrices. For both models, however, it is not a priori clear how to best arrive at locally linear descriptions of the data manifold.

Original MC locates a local manifold chart at every given point of the data space and fits the linear model by means of optimization of a statistical model of the environment. This leads to convex problems which can be optimally solved explicitly, however, since a local chart is associated with every data point, a rather costly model with no compression at the data level arises. In [67], an alternative is proposed which can be combined using both, manifold charting and locally linear coordination: a mixture model is fitted to the data manifold which represents the data using only a limited number of different local representations, such as, for example, a Gaussian mixture model obtained e.g. by mixtures of probabilistic PCA [38] or factor analyzers [70]. This procedure leads to a much more compact representation of the embedding and a much smaller problem for the glueing of the pieces, since the size is determined by the number of local charts. This benefit is paid for by the fact that an optimization of the local factors or local Gaussian mixture components on the manifold can no longer be solved explicitly analytically and multiple optima can occur in this nonconvex optimization problem. The standard way to optimize the underlying likelihood functions is given by an EM approach which is usually very sensitive to initialization, such that the overview reported in [65] even comes to the conclusion that these methods are in general not competitive to alternatives such as Isomap or LLE.

In this chapter, we propose to substitute this initialization sensitive local charting step by matrix neural gas since it displays a better convergence and a more robust behavior. This way, a competitive visualization method results which also provides an explicit mapping of the data manifold onto lower dimensionality. We evaluate this method in artificial and real life benchmark examples as proposed in the extensive comparison of data visualization methods in the article [65]. It turns out that MNG

together with manifold charting is at least competitive if not better compared to Isomap and LLE, and it provides in most cases better results than the statistical approach based on manifold charting and probabilistic principal component analysis to arrive at the local linear projections. Like the latter method, however, an explicit map is found which describes the embedding of the data manifold into low dimensions, and which can be further explored e.g. to get approximate inverse images of low dimensional points.

In the next section, we will focus on applications of matrix learning for manifold visualization, which become possible due to the additional information provided by the local matrices. Note that, depending on the dimensionality m of the original data points, full matrix learning in MNG is rather time consuming, requiring matrix inversion of order $O(m^3)$ in every step. Since only the minor d eigenvalues of the matrix Ω_i are relevant for the local projection, we can priorly reduce the matrix such that the scaling of only the minor d principal components is individually adapted while the scaling remains identical (and nonzero to avoid degeneration) for all other directions. This can be achieved efficiently with any algorithm which extracts the major d principal components of the generalized data correlation matrix, e.g. a generalized Sanger rule as proposed in [28] . Note that the scaling must not be zero for any direction since this would lead to an annulation of the dimensions with most statistical importance, i.e. the dimensions with largest variance would vanish.

5.2 Manifold charting based on MNG

Data visualization and low dimensional embedding constitute important problems of data mining. MNG provides local linear transformations of the data induced by the local matrices Λ_i and the corresponding eigenvectors, as follows: Assume the eigenvalue

decomposition

$$\Lambda_i = \Omega_i^t \cdot D_i \cdot \Omega_i$$

is given with a diagonal matrix D_i of eigenvalues and eigenvectors collected in Ω_i. Assume data should be mapped to dimensionality d where, often, $d \in \{1, 2, 3\}$ such that visualization is possible. Then we can reduce D_i to only the d smallest eigenvalues (which are the main eigenvalues of S_i, i.e. they belong to the main principal components of the receptive field) getting the $d \times m$ matrix D_i^{red}. The formula

$$A_i : \mathbb{R}^m \to \mathbb{R}^d, \vec{x} \mapsto D_i^{\text{red}} \cdot \Omega_i^t \cdot (\vec{x} - \vec{w}^i) \tag{5.1}$$

gives the local linear projection of the data points to the main principal components of the receptive field induced by the ith chart of the data manifold. If d is chosen at most 3, every map A_i provides a linear visualization of the manifold which is faithful within the receptive field of prototype $\vec{w}^i$ because it corresponds to the main eigenvalues of the local chart. Now the question occurs how these local projections can be extended to a global low-dimensional embedding of the data manifold.

Manifold charting as introduced in [64] is based on local linear mappings of data points which are glued together such that the projections fit on the overlapping pieces. In the original framework, a linear map is attached to every data point, yielding a rather time consuming scheme. Later, a reduction to only few charts was proposed e.g. in [67]. However, the glueing method works for every given set of local linear mappings as long as it is smooth on neighbored pieces and visualization in low dimensions is possible at all. We will use these ideas to obtain a global data visualization method from local matrix learning.

Assume that linear projections $A_1, \ldots, A_n$ are given (e.g. by formula (5.1)) which define n local projections $\vec{z}^{1i} = A_1(\vec{x}^i), \ldots, \vec{z}^{ni} = A_n(\vec{x}^i)$ of the data points $\vec{x}^1, \ldots, \vec{x}^p$. Assume that, in addition, a responsibility value $p_{ji} = p_j(\vec{x}^i)$ is specified for every data point $\vec{x}^i$ and chart A_j which defines the responsibility of this chart for the data point,

whereby $\sum_j p_{ji} = 1$ for every i. The goal is to combine the local charts A_i by means of local affine mappings $B_i : \mathbb{R}^d \to \mathbb{R}^d$ to a global mapping such that the compositions lead to matching points if more than one chart is responsible for a data point. The final points are then obtained by the formula $\vec{x}^j \mapsto \sum_i p_{ij} B_i(\vec{z}^{ij})$. The mappings B_i are determined in such a way that the following costs are minimized

$$E_{\text{charting}} = \frac{1}{2} \cdot \sum_{i,j,k} p_{ji} p_{ki} \| B_j(\vec{z}^{ji}) - B_k(\vec{z}^{ki}) \|^2 \tag{5.2}$$

which (as can be seen by a simple algebraic transformation) also give the difference of the globally mapped points and the local affine transformations of the points. As shown in [64] a unique solution of this problem can be found as follows: set $D \in \mathbb{R}^{n(d+1) \times n(d+1)}$ as the block diagonal matrix with blocks $D_i = \sum_{j=1}^{p} p_{ij} \cdot [\vec{z}^{ij}\ 1][\vec{z}^{ij}\ 1]^t$ where the entry 1 is added to account for the offset in the affine transformations. Let $U \in \mathbb{R}^{n(d+1) \times p}$ be the matrix with block entries $u_{ij} = p_{ij}[\vec{z}^{ij}\ 1]$ where the components of $\vec{z}^{ij}$ are integrated into the block index i of the matrix. Then, the cost function (5.2) equals $E_{\text{charting}} = B^t(D - U^tU)B$ where matrix B collects the free parameters of the affine transformations B_i which have to be found. This formulation can be linked to a Rayleigh quotient, hence the maximum is obtained by the d largest generalized eigenvalues v of the generalized eigenvalue problem $(D - U^tU)v = \lambda U^tUv$. Thus, an analytic solution of the problem can be found provided the projections $\vec{z}^{ij}$ and responsibilities p_{ij} are available.

Now, we can apply this charting procedure to the results of a MNG clustering where the projections are given by formula (5.1). Note that MNG provides crisp receptive fields and, hence, there is no overlap of neighbored receptive fields. We change the crisp assignments of MNG to arrive at probabilistic setting with a small overlap of neighbored receptive fields. We simply use Gaussians centered at the prototypes $\vec{w}^i$ computed by MNG using a covariance matrix given by the learned matrices. More precisely, set $N_i = |\{\vec{x}^j \mid I(\vec{x}^j = i\}|$ as the number of points in the ith receptive field.

Let S_i be the (unnormalized) correlation matrix of the ith receptive field as computed in MNG and $\tilde{S}_i = S_i/N_i$ the associated covariance matrix. Then we set the responsibility of the ith receptive field for point $\vec{x}^j$ as

$$\tilde{p}_{ij} = \tilde{p}_i(\vec{x}^j) = \frac{N_i}{p} \cdot \frac{1}{(2\pi)^{m/2}\sqrt{\det \tilde{S}_i}} \cdot \exp(-0.5 \cdot (\vec{x}^j - \vec{w}^i)^t \cdot \tilde{S}_i^{-1} \cdot (\vec{x}^j - \vec{w}^i)) \quad (5.3)$$

where the prior N_i/p refers to the relative number of points in chart i. The responsibilities p_{ij} are obtained thereof by normalization $p_{ij} = \tilde{p}_{ij}/\sum_i \tilde{p}_{ij}$.

Note that, depending on the distribution of the prototypes, the assignments will be sparse since p_{ij} will be almost zero for receptive fields i which lead to a high rank w.r.t $\vec{x}^j$. To speed up the computation, it is possible to cut off these small assignments and work with sparse matrices. Based on these choices of p_{ij} provided by (5.3) and $\vec{z}^{ij}$ provided by the affine transformations (5.1), MNG can be combined with manifold charting to give a global nonlinear embedding or visualization of data. Obviously, this visualization can be described by an explicit mapping of $\mathbb{R}^m \to \mathbb{R}^d$ by means of the formula

$$\vec{x} \mapsto \sum_i p_i(\vec{x}) \cdot B_i(A_i(\vec{x})) \quad (5.4)$$

where $p_i(\vec{x})$ is computed according to (5.3), the local linear mappings A_i are given by (5.1), and the affine transformations B_i to glue the charts together are determined by solving equation (5.2).

Inverse map

To arrive at an approximate inverse map, we take a simple point of view which allows us to compute the inverse algebraically. Note that every local linear projection A_i possesses an approximate inverse A_i^{-1} induced by the pseudo-inverse of $D_i^{\mathrm{red}} \cdot \Omega_i$. Since A_i maps to lower dimensions, this is, of course, no exact inverse but its best approximation in a least squares sense. Further, obviously, the affine transformations

B_i can be inverted exactly. Thus, for every $\vec{z} \in \mathbb{R}^d$, an approximate inverse of the image of (5.4) can be determined in the following way: for a given $\vec{x}$, we determine the inverse $\tilde{x}$ under chart i:

$$\tilde{x} \leftarrow A_i^{-1} \circ B_i^{-1}(\vec{z}). \tag{5.5}$$

From these possibly preimages, we take the one with maximum responsibility according to (5.3).

5.3 Manifold visualization

A variety of benchmarks for dimensionality reduction tasks has been collected in [65] together with an evaluation of several popular models on these tasks. Thus, we will basically use the setting as proposed in [65] to achieve comparability to a wide range of alternative methods. The article [65] investigates the following methods: PCA, Isomap, Maximum Variance Unfolding, kernel PCA, Diffusion Maps, Autoencoder networks, LLE, Laplacian Eigenmaps, Hessian LLE, Local Tangent Space Analysis, LLC, and MC. Thereby, parameters have been optimized in a reasonable range by extensive search. We will restrict our settings to mixtures of probabilistic PCA as a direct competitor to matrix neural gas, and LLE and Isomap as two of the most well-known data projection methods.

In all cases, the possibility to preserve the local structure of the manifold while embedding in lower dimensions are evaluated by a one nearest neighbor (1-NN) classification scheme in [65]. This is a reasonable scheme, since 1-NN measures local properties of the data, which should be preserved by projections. Further, the classification task is connected to the manifold structure in the sense that clusters are smooth on the original manifold. Partially, this is by definition of the clustering task: for all artificial tasks, a cluster structure which resembles a checkerboard has been arranged

on the map, such that 1-NN displays good performance iff the manifold structure is preserved. For the real-life applications, the evaluation of a 1-NN in the original data space measures the degree according to which the cluster structure is correlated to the original manifold topology. However, a 1-NN classification need not coincide with the geometrical structure of the manifold which should be preserved in these cases. Therefore, the trustworthiness of the projection as proposed in [71] is also computed. This quantity measures the proportion of points that are too close together in the projection compared to the original manifold. Assume $k > 0$ is fixed, and the k nearest neighbors are computed for every point in the original data space and the projection, respectively. Denote by $U_i^{(k)}$ the points which are among the k-nearest neighbors of i in the projection, but not in the original data space. Further, denote the rank of a point j to i in the k-nearest neighbor graph by $r(i,j)$. Then the trustworthiness is given by the formula

$$T(k) = 1 - \frac{2}{pk(2p - 3k - 1)} \sum_{i=1}^{p} \sum_{j \in U_i^{(k)}} (r(i,j) - k)$$

where p denotes the number of points. It gives the percentage of points which local projection can be considered as trustworthy w.r.t. the local neighborhood structure.

Since the original data sets of [65] are not available, rather a description or a generation method is provided by [65], we construct the data sets in the same way as described in [65]. We always evaluate the result of a 1-NN classification for the original data (getting values close to [65] which indicates that our settings are comparable) and we evaluate, besides our MNG-charting algorithm, the schemes provided by Isomap, LLE, and manifold charting combined with mixtures of probabilistic PCA. Thereby, we use the Matlab algorithms as provided in [65] for the other projection algorithms and for the charting step.

Artificial data sets

Artificial data sets are generated according to the program provided in [65], whereby 2000 points where randomly sampled for every run. The results are averaged over ten runs using different random data samples, also displaying standard deviation. The parameters for LLE and Isomap (the number of neighbors k) where chosen optimum in the range of $k \in \{20, \ldots, 40\}$ for the experiments. The number of clusters used for matrix clustering and mixture of probabilistic PCA, respectively, are given in Tab. 5.1. All data sets are projected to 2D except for HD which is projected to 5D. The column 'None' refers to the result of a 1-NN classifier in the original data space, for comparison. The respective best result obtained by a projection method is depicted in boldface. Tab.5.2 shows the trustworthiness of the projection methods in the same settings using $k = 12$ neighbors for evaluation.

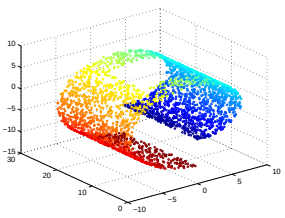
As can be seen, MNG with manifold charting shows competitive results in almost all settings (except for the twinpeaks data where it is a bit worse, but it still gives very acceptable result, in particular since a 'correct' projection of this data set is not clear, see Fig.5.1). Unlike manifold charting combined with mixture of probabilistic PCA, the results are very stable independent of the initialization of the algorithm. Unlike LLE and Isomap, an explicit mapping of the data manifold to low dimensions is provided. Representative projections computed in this way are depicted in Figs. 5.1 - 5.4.

Dataset	None	LLE	Isomap	MoPPCA	MNG	clusters
swissrole	3.36±0.40	27.10±1.69	19.84±14.18	6.07±5.83	**3.87±0.60**	12
swissrole with hole	3.28±0.12	21.06±1.22	11.92±4.28	**3.20±0.45**	3.66±0.25	16
broken swissrole	4.23±0.50	27.23±2.63	23.52±2.18	29.98±15.16	**14.50±4.92**	32
toroidal helix	1.29±0.35	4.36±2.13	12.72±15.19	**1.42±0.32**	2.00±0.61	60
twinpeaks	0.73±0.25	1.36±0.43	**0.45±0.21**	0.80±0.25	1.38±0.57	32
cornerplanes	10.38±0.60	10.60±0.39	10.59±0.55	**10.58±0.72**	10.74±0.65	8
punctured sphere	6.64±0.68	19.49±1.64	14.49±1.48	18.98±3.27	**9.29±1.28**	50
gaussian shape	6.30±0.08	**6.26±0.08**	6.28±0.11	**6.26±0.08**	8.35±1.02	16
HD	23.73±0.80	22.93±1.13	**21.73±1.36**	32.01±7.34	28.53±3.40	32

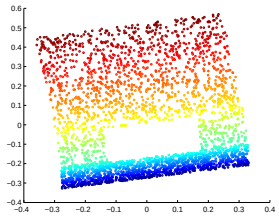
Table 5.1: Results of a 1-NN classification if the artificial data sets are embedded in low dimensions using different methods.

Dataset	LLE	Isomap	MoPPCA	MNG
swissrole	0.8753±0.0033	0.9284±0.0692	0.9734±0.0595	**0.9993±0.0013**
swissrole with hole	0.8905±0.0099	0.9619±0.0103	0.9994±0.0012	**0.9999±0.0002**
broken swissrole	0.8409±0.0115	**0.9526±0.0083**	0.8990±0.1045	0.9139±0.0572
toroidal helix	0.9940±0.0059	0.9118±0.1212	**1.0000±0.0000**	0.9975±0.0035
twinpeaks	0.9949±0.0016	**0.9997±0.0003**	0.9941±0.0009	0.9875±0.0084
cornerplanes	**1.0000±0.0000**	**1.0000±0.0000**	**1.0000±0.0000**	**1.0000±0.0000**
punctured sphere	0.9727±0.0077	0.9970±0.0004	0.9871±0.0026	**0.9997±0.0006**
gaussian shape	**1.0000±0.0000**	**1.0000±0.0000**	**1.0000±0.0000**	0.9985±0.0020
HD	**1.0000±0.0000**	**1.0000±0.0000**	0.9906±0.0110	0.9955±0.0043

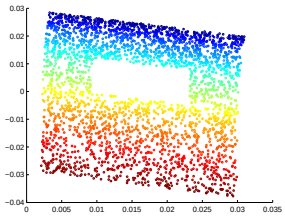
Table 5.2: Trustworthiness of the projections obtained for the artificial datasets.



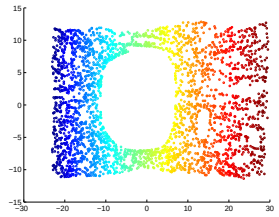
(a) original data



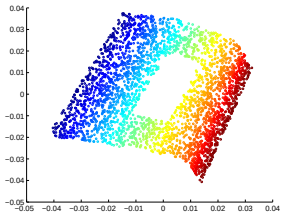
(b) charting + MoPPCA



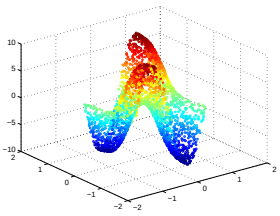
(c) charting + MNG



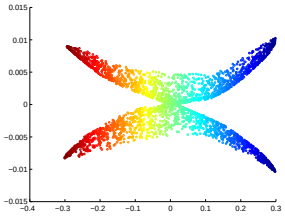
(d) ISOMAP



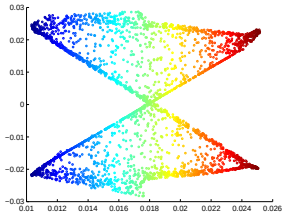
(e) LLE



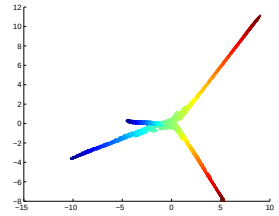
(a) original data



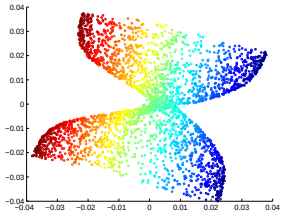
(b) charting + MoPPCA



(c) charting + MNG

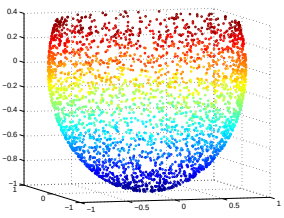


(d) ISOMAP

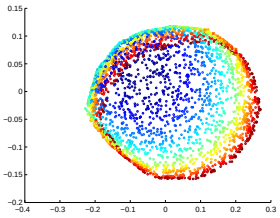


(e) LLE

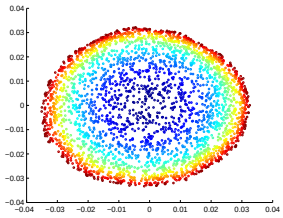
Figure 5.1: Projection of the swiss roll with hole and twin peaks data sets using manifold charting and mixture of probabilistic PCA, manifold charting and MNG, Isomap, and LLE, respectively



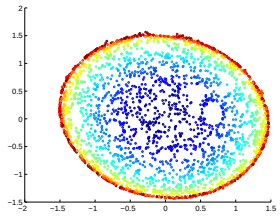
(a) original data



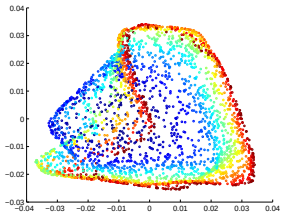
(b) charting + MoPPCA



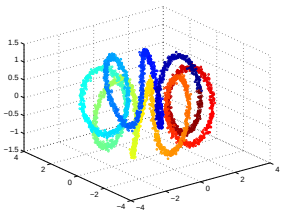
(c) charting + MNG



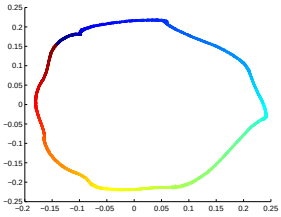
(d) ISOMAP



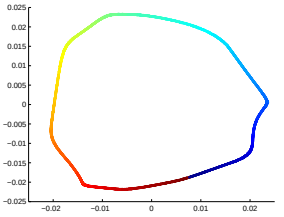
(e) LLE



(a) original data

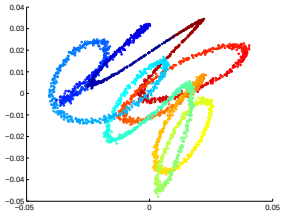


(b) charting + MoPPCA



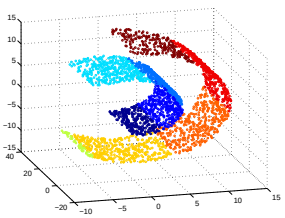
(c) charting + MNG

(d) ISOMAP

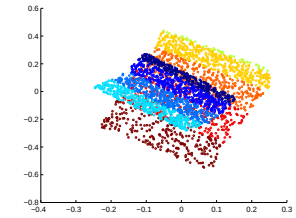


(e) LLE

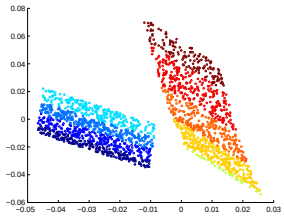
Figure 5.2: Projection of the punctured sphere and toroidal helix data sets using manifold charting and mixture of probabilistic PCA, manifold charting and MNG, Isomap, and LLE, respectively



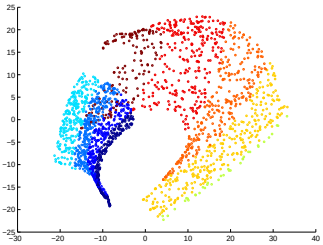
(a) original data



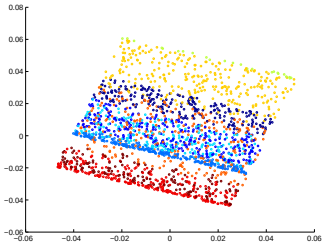
(b) charting + MoPPCA



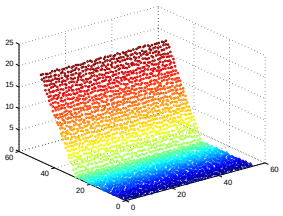
(c) charting + MNG



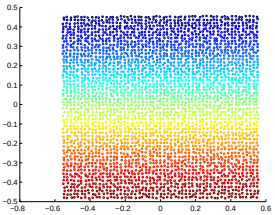
(d) ISOMAP



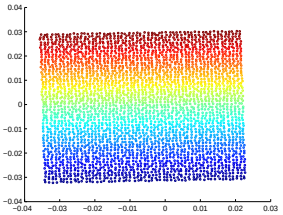
(e) LLE



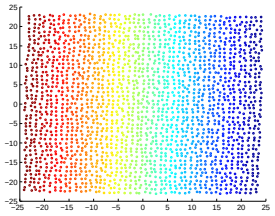
(a) original data



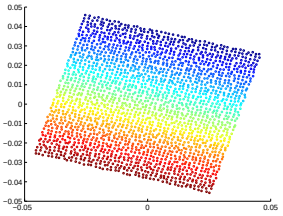
(b) charting + MoPPCA



(c) charting + MNG

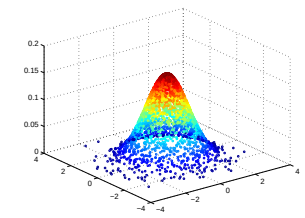


(d) ISOMAP

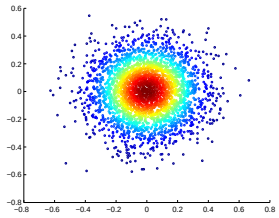


(e) LLE

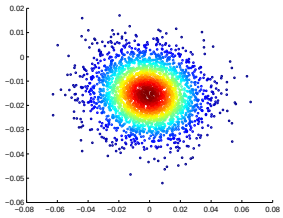
Figure 5.3: Projection of the broken swissrole and corner planes data sets using manifold charting and mixture of probabilistic PCA, manifold charting and MNG, Isomap, and LLE, respectively



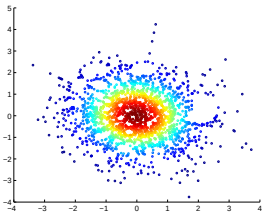
(a) original data



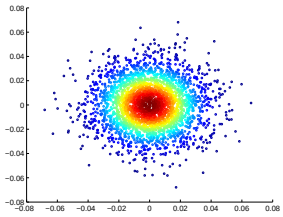
(b) charting + MoPPCA



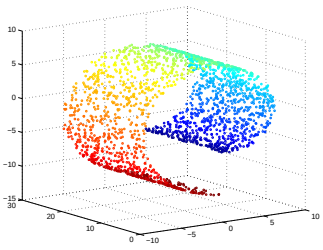
(c) charting + MNG



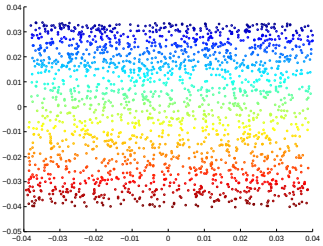
(d) ISOMAP



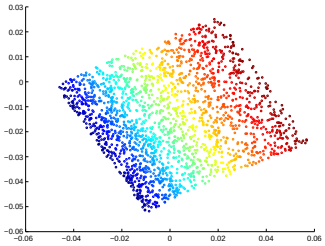
(e) LLE



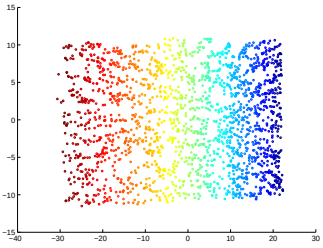
(a) original data



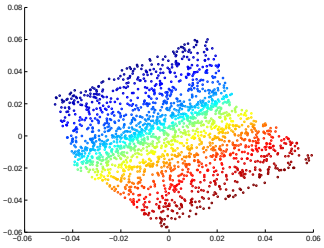
(b) charting + MoPPCA



(c) charting + MNG



(d) ISOMAP



(e) LLE

Figure 5.4: Projection of the gaussian and swiss role data sets using manifold charting and mixture of probabilistic PCA, manifold charting and MNG, Isomap, and LLE, respectively

Real life data sets

Four real life datasets where tackled as proposed in [65]. As before, averages over ten runs are reported. The MNIST data set contains 60000 images of handwritten digits displaying classes 0 to 9 with dimensionality 28×28. For every run, a random sample of 300 images per class was taken. Data was preprocessed by PCA extracting the main 30 principal components. The dimensionality reduction techniques as described above were used to further map these data to dimensionality 20. The COIL20 data set consists of 20 objects taken from 72 viewpoints each. Again, the original dimensionality 32×32 was reduced to 30 using PCA, and further projected to 5 dimensions using the embedding techniques. The ORL data set contains 40 different faces displayed in grayscale of size 112×92, displaying 10 different expressions per face. Again, PCA was used to project to 30 dimensions and dimensionality reduction was used to nonlinearly embed in dimensionality 8. Finally, the HIVA dataset is a benchmark drug discovery data set with two classes and 3845 data points with dimensionality 1617, which was reduced to 30 using PCA and, further, to 15 using nonlinear data visualization. The results of the methods are reported in Tab. 5.3 and 5.4 for the 1-NN classification error and the trustworthiness with $k = 12$ neighbors. For matrix clustering and mixture of probabilistic PCA, only 4 clusters have been used. Obviously, matrix learning for neural gas together with manifold charting shows very good results for these real life data sets, reaching the best performance in all but one cases, see Tabs. 5.3 and 5.4. Again, it displays much better stability than mixture of probabilistic PCA, and it even shows superior results to LLE and Isomap, which, as already claimed in [65], do not always achieve the same quality for real-life data sets as for artificial ones.

Dataset	None	LLE	Isomap	MoPPCA	MNG
MNIST	7.98±0.33	11.68±1.56	16.31±5.44	53.87±38.31	**9.51±0.28**
COIL20	0.00±0.00	6.94±2.40	8.33±0.00	10.16±1.60	**4.70±1.09**
ORL	5.50±0.00	24.27±4.47	18.75±0.00	33.90±17.93	**13.10±1.51**
HIVA	4.573±0.00	5.38±0.17	5.35±0.00	5.27±0.43	**5.21±0.22**

Table 5.3: Generalization errors and standard deviation of 1-NN classifiers trained on natural datasets

Dataset	LLE	Isomap	MoPPCA	MNG
MNIST	0.9916±0.0027	0.9790±0.0231	0.7419±0.2204	**0.9975±0.0007**
COIL20	0.9567±0.0102	**0.9937±0.0000**	0.9757±0.0089	0.9907±0.0035
ORL	0.9423±0.0078	0.9876±0.0000	0.9319±0.0580	**0.9889±0.0008**
HIVA	0.9703±0.0057	0.9675±0.0000	0.9480±0.1482	**0.9954±0.0005**

Table 5.4: Trustworthiness trained on natural datasets

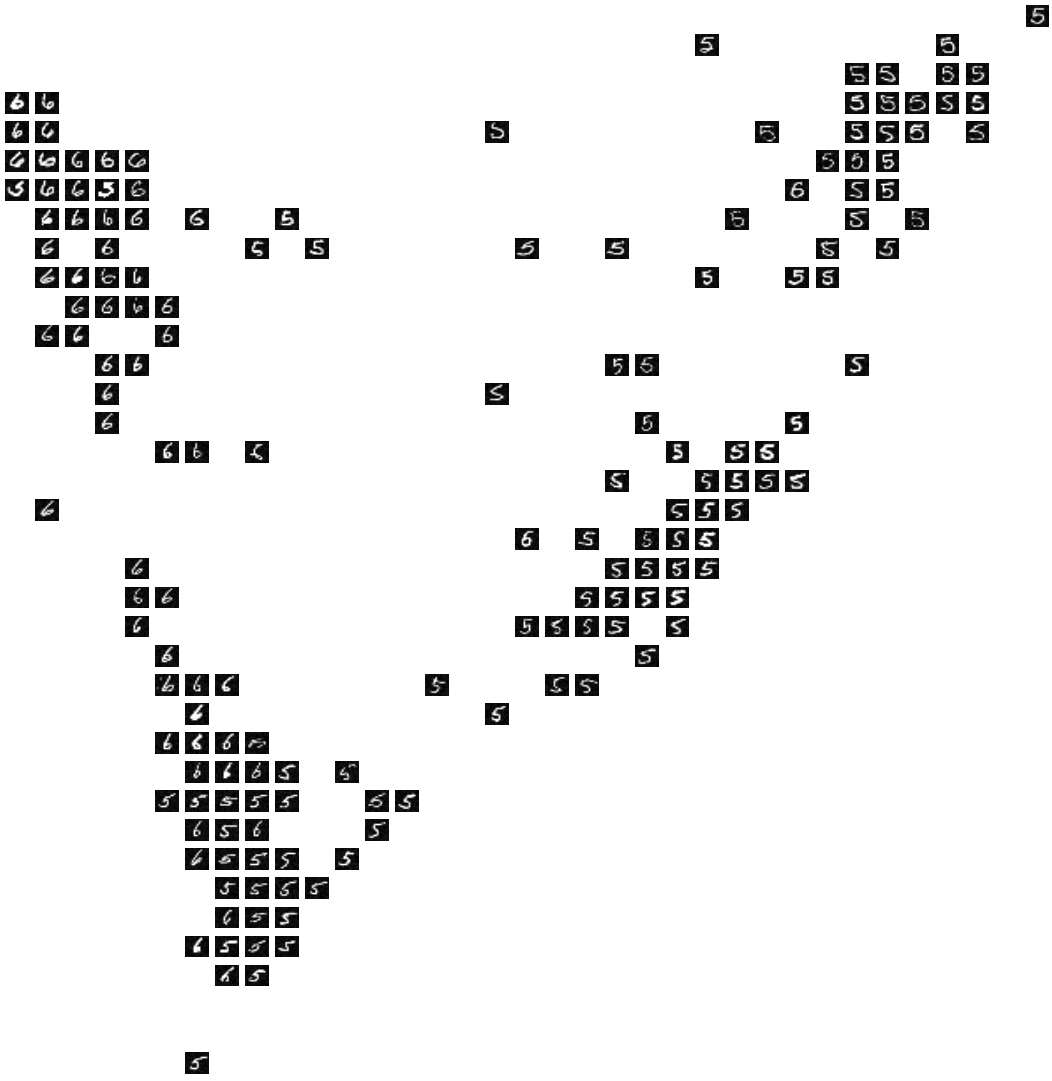
Visual comparison on the MNIST data

We exemplarily show the projection obtained by these methods for the MNIST data in two dimensions. For this purpose, we randomly select 300 samples of digits 5 and 6 each, respectively, which display a similar shape and stroke. We would like to evaluate the possibility of the projection methods to distinguish between these two digits when projecting data to 2 dimensions. Again, we use 4 subclusters for MNG and MoPPCA, respectively, and we project the data which has been preprocessed by linear PCA, as before, to two dimensions. Fig. 5.5 shows the resulting 2-D projections, whereby we attach the respective closest original digit to the projected points on a grid. These images constitute a representative subsample of the original 600 images and their arrangement in two dimensions. Obviously, MNG well separates the images of digits 5 and 6, the digit 6 being mostly in the upper left, the digit 5 in the right part of the

projection. MoPPCA leads to a mixture of the two classes with the digits 5 and 6 arranged in one big cluster. Similarly, the two classes are displayed in an only weakly separated big cluster when using LLE or Isomap, respectively. For Isomap, the digit 5 is displayed in the left part, the digit 6 in the right part. For LLE, the classes are mixed. On the contrary, MNG decomposes the space into separated clusters according to the underlying prototypes, which, in this case, nicely resemble the original classes of the digits with a clear separation of the left and right have of the image. For comparison, we report the result obtained by a 12×12 standard SOM trained in batch mode. Data are preprocessed in the same way, and the image closest to a prototype is displayed at the respective prototype position. Obviously, SOM separates the two clusters quite well, whereby the gradient between the clusters is quite smooth. While a few neurons remain idle in this case, these are not located at class boundaries, i.e. the two clusters are located directly beneeth each other. Unlike the other methods, SOM does not provide a mapping of the points to disjoint positions nor an explicit embedding function like Charting in combination with MNG. Rather, it projects the data onto a finite fixed number of given prototype locations in the map.

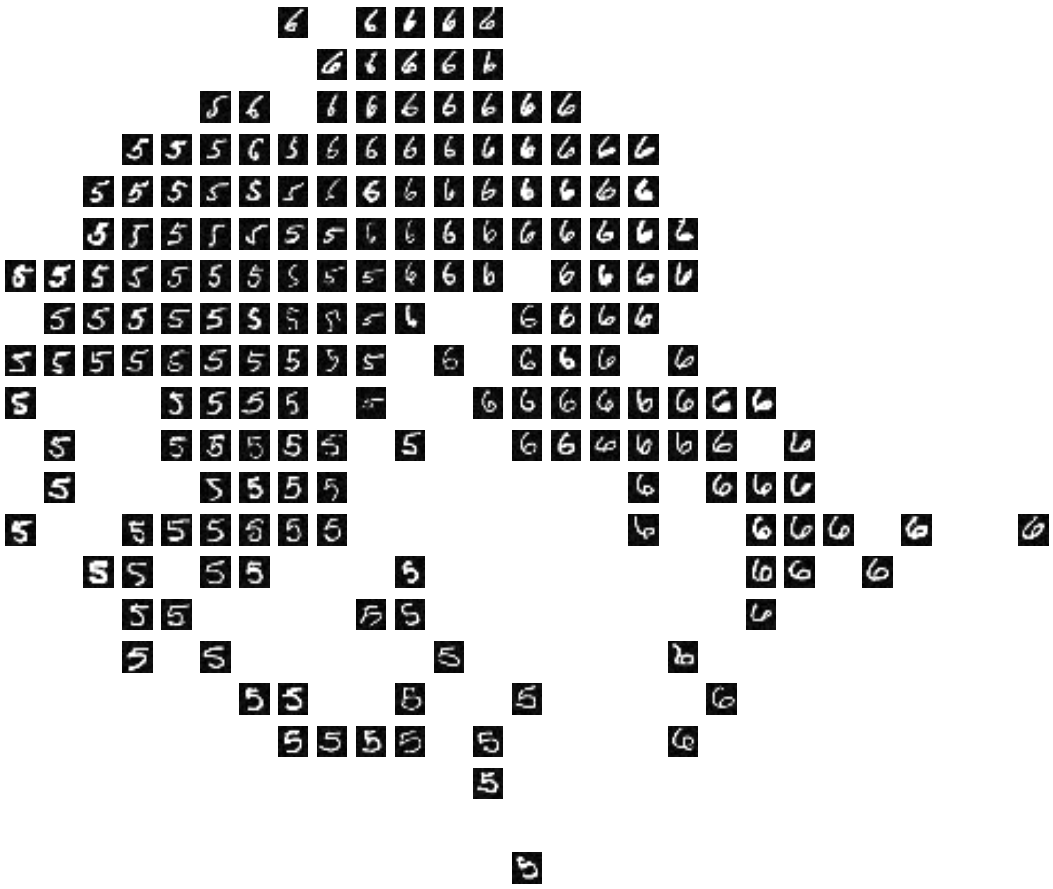
Note that we used preprocessing by PCA for the MNIST data set as well as the other data sets before projection by nonlinear methods, as suggested in [65]. This way, the linear parts in the data are first linearly compressed in an obvious way before using nonlinear projection methods to arrive at a suitable low-dimensional presentation of the data. Preprocessing data using PCA has the benefit of a reduced computational complexity of the subsequennt nonlinear projection methods as well as a greater numerical stability. For matrix clustering, the possibility to directly process high-dimensional data has been discussed extensively in the contributions [69, 28, 82, 88]. Note that matrix inversion, as required by MNG, is cubic with respect to input dimensionality, such that already dimensionalities around 100 lead to quite long runs. Even more severely, matrix inversion constitutes a numerically non-trivial problem for large dimensions

with standard matrices close to singularity. Although, in theory, the involved matrices are non-singular for almost all cases ($n + 1$ data points in general position being a sufficient condition that the involved matrices are nonsingular) numerical problems occur in practice due to matrices which are close to singular. Therefore, it is not advisable to use matrix clustering directly for data sets with dimensionality much larger than 50. Preprocessing with standard methods such as PCA or variations of MNG which are specially tuned for such data sets (see [28]) should be applied in such cases.

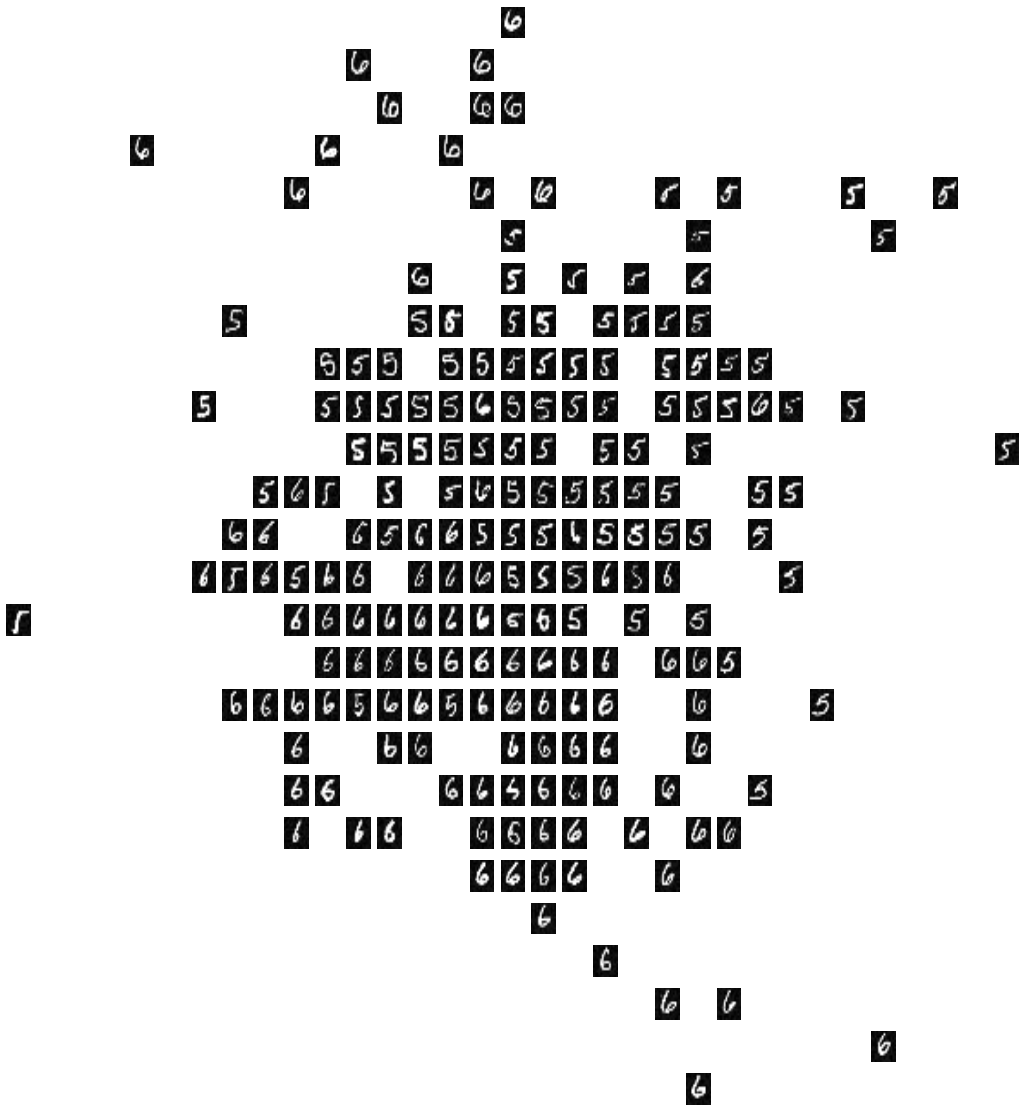


(a) charting + MNG

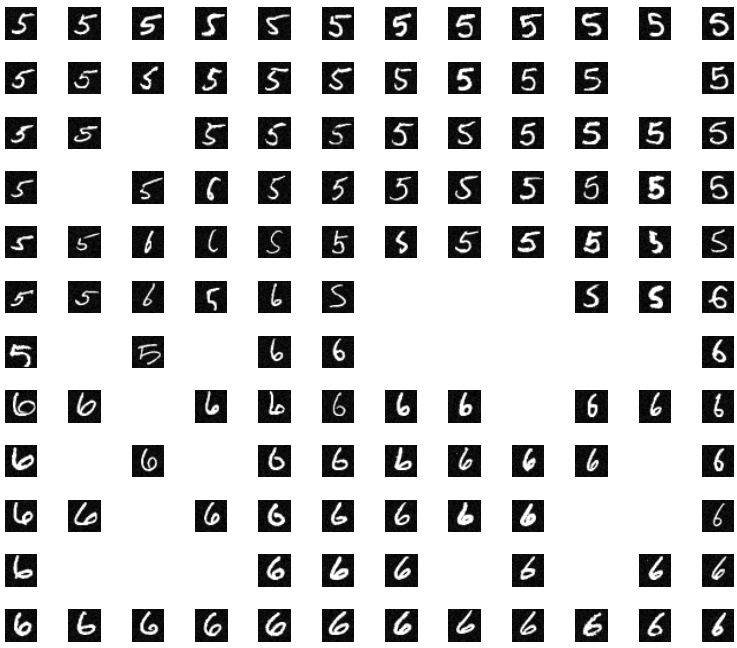
(b) charting + MoPPCA



(c) ISOMAP



(d) LLE



(e) standard SOM

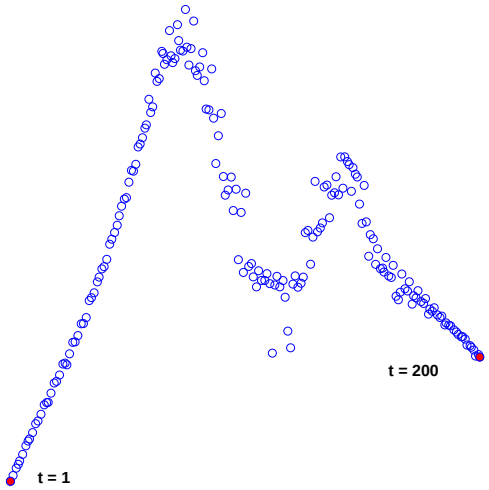
Figure 5.5: 2-D projections of data from the MNIST data set when using MNG, MoPPCA, Isomap, LLE, and standard SOM respectively. Interestingly, LLE, Isomap, charting with MoPPCA, and SOM map the data onto one big cluster, while charting with MNG displays a better separation of the clusters formed by the two digits on the left and right of the projection. The separation of the digits is quite clear for charting with MNG, ISOMAP, and SOM, whereas charting with MoPPCA and LLE result in a mixed arrangement of the two digits. These latter arrangements follow subtleties in the writing of the figures, such as a more curly stroke of the digits 6 displayed in the upper part of the projection by LLE, or a slope to the right of digits in the right part of the projection provided by charting with MoPPCA.

Structure representation of images

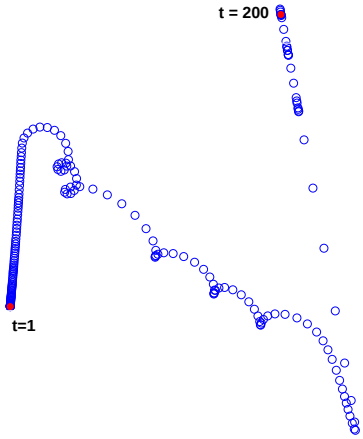
One goal of nonlinear dimensionality reduction is to extract the internal structure of very high dimensional data. RGB image constitute the high dimensional data when vectorized into a vector, in which, the dimensionality is the number of pixels multipled according to the color palette. More generally, an image sequence of human faces or solid objects can be used to study the underlying structure and the possibility to preserve it by matrix learning. In this section, we use images of a porcelain teapot which are created by viewing the teapot from different angles. The whole dataset consists of 200 color images which are derived from a rotation of 180 degrees. Each image contains 76×101 pixels, with RGB color palette. Therefore, each image can be seen as a point in a 23028 dimensional data space.

Although very high dimensional, this data set depends on only one parameter: the angle of rotation. Because of this fact, the internal structure of the image sequence can be presented in a low dimensional space. We use nonlinear techniques such as ISOMAP, LLE, charting based on MoPPCA and MNG to project these images onto a 2D embedding space. Fig. 5.6 demonstrates 2D representation of the teapot images. The outputs are arranged in a trajectory and ordered according to their angle of rotation. The start and end points of rotation are marked in red on the trajectory.

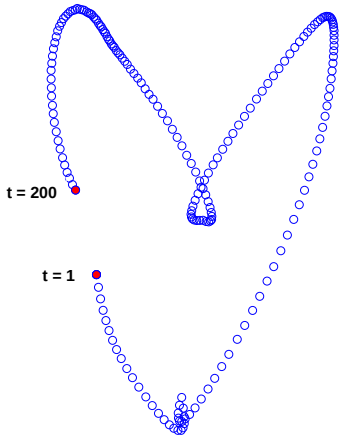
ISOMAP yields a non-smooth trajectory while LLE produces an irregular unfolding result, in which, the early period of projected points has high density but the latter part is sparse. Charting based on MoPPCA gives a smooth and regular trajectory better than the results of ISOMAP and LLE, but still has the problem of two knots as can be seen in Fig. 5.6 (c). Obviously, the result obtained from MNG and Charting is a well-ordered trajectory which is appropriate to represent the correct structure. Figs. 5.7 and 5.8 show the trajectories with representative images corresponding to their location in the embedding space.



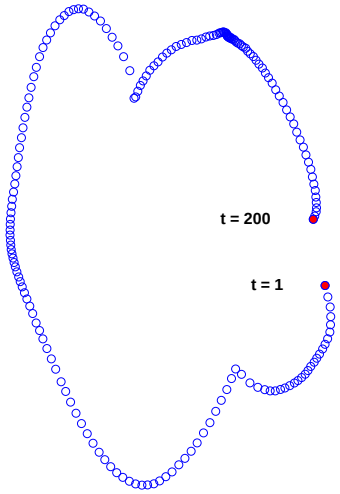
(a) ISOMAP



(b) LLE

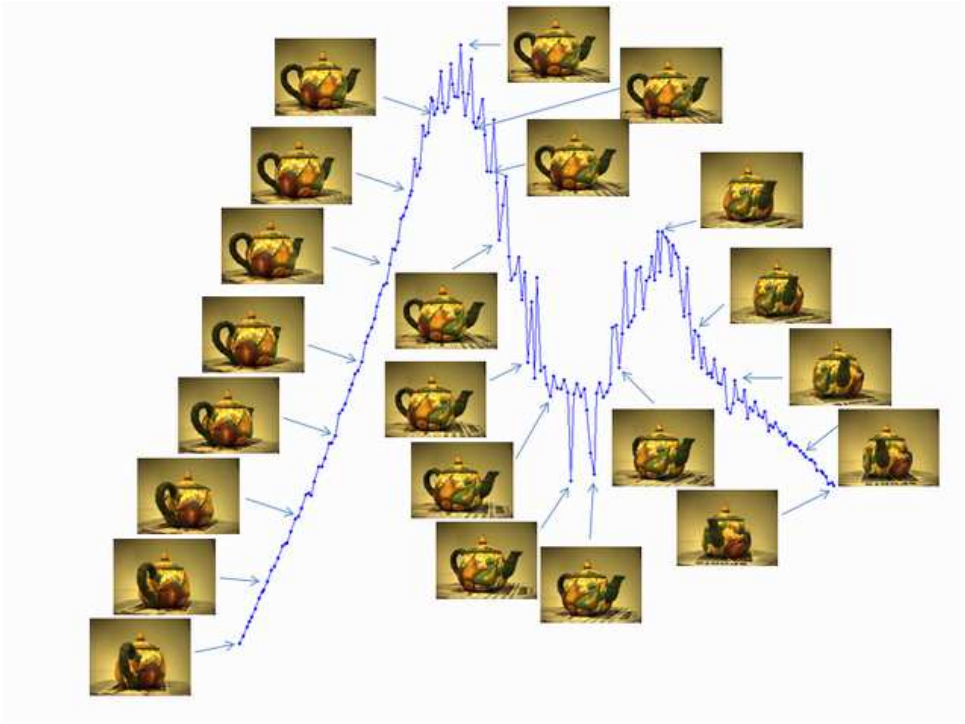


(c) Charting + MoPPCA

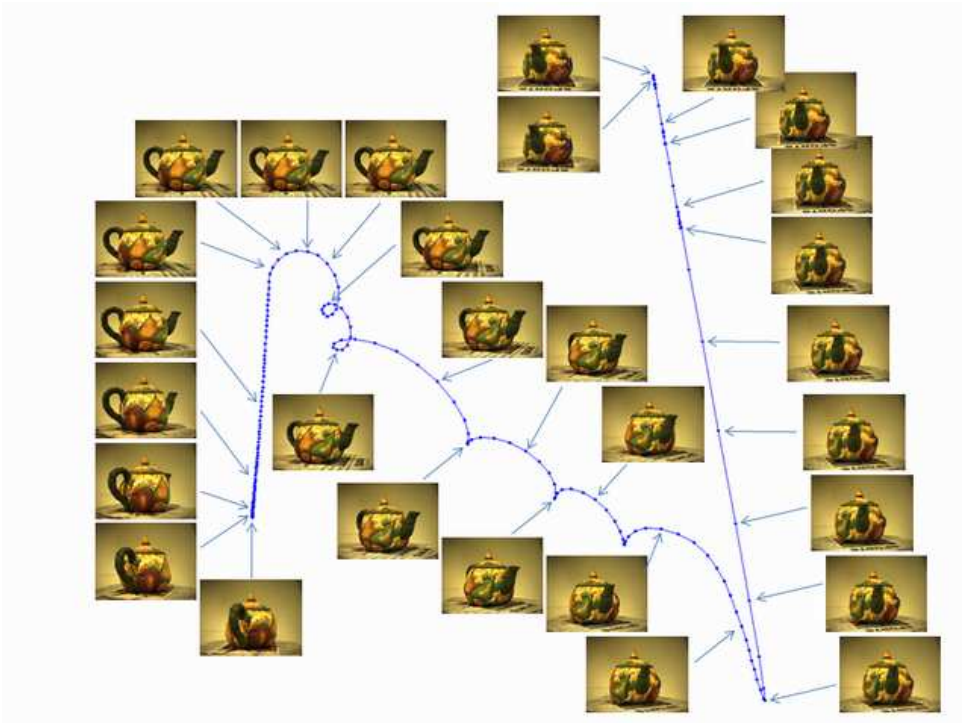


(d) Charting + MNG

Figure 5.6: Trajectories in 2D embedding space from the teapot data set when using Isomap (upper left), LLE (upper right), MoPPCA (lower left), and MNG (lower right), respectively.

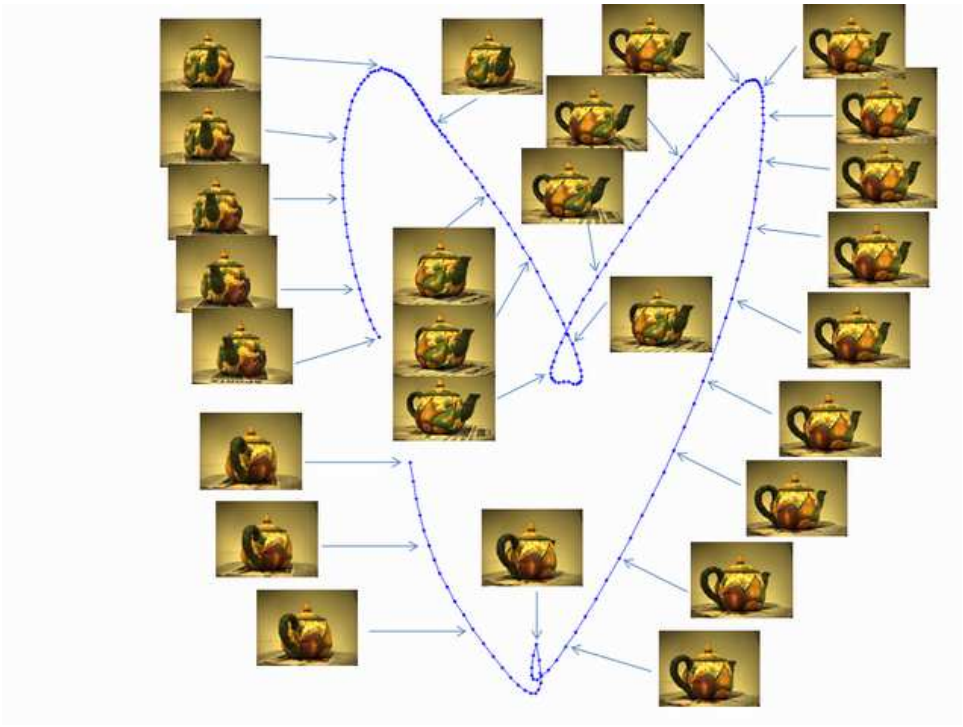


(a) ISOMAP

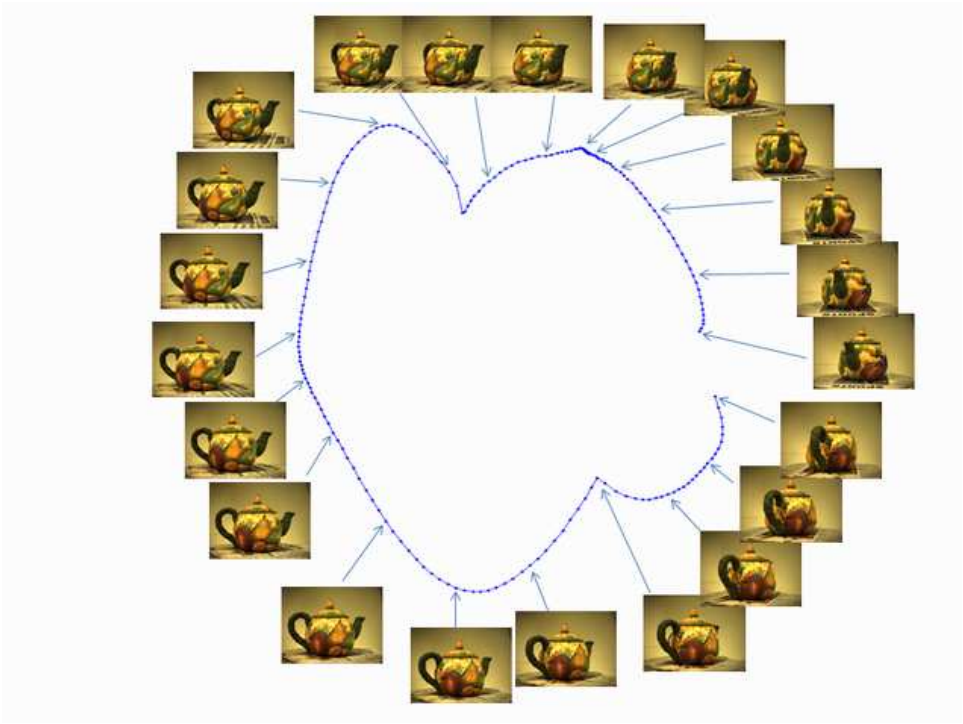


(b) LLE

Figure 5.7: 2D representations of teapot image sequence projected by ISOMAP and LLE, respectively. Representative images are demonstrated according to their location on embedding space.



(a) Charting + MoPPCA



(b) Charting + MNG

Figure 5.8: 2D representations of teapot image sequence projected by Charting based on MoPPCA and MNG, respectively. Representative images are demonstrated according to their location on embedding space.

5.4 Dynamic Texture Analysis and Synthesis

In chapter 3, extensions of NG have been proposed which also adapt local matrices during training such as to minimize the quantization error. This corresponds to local coordinate systems which represent smooth local principal directions of data. It has been demonstrated in section 5.2, that these additional parameters offer sufficient information to extract explicit local coordinate systems from the data which can be further processed to obtain a global nonlinear projection of the underlying manifold e.g. using manifold charting [64]. This way, an explicit mapping together with its approximate inverse is obtained which can map high dimensional data into low dimensional space. In this chapter, the possibility to use this mapping for low dimensional data visualization and representation has been explored in comparison to popular alternative visualization schemes as referenced e.g. in [24]. Unlike popular alternative visualization methods such as locally linear embedding, maximum variance unfolding, or stochastic neighbor embedding, manifold charting in combination with matrix neural gas does not only embed the given data points, but it provides an explicit low dimensional embedding of the data manifold and an approximate inverse of the map. Hence additional information is available which can be explicitly used in further applications.

In this section, we will demonstrate the suitability of manifold embedding by matrix neural gas for an interesting problem from computer graphics, the efficient synthesis of dynamic texture based on given examples. Dynamic texture synthesis is the process of producing an animation of dynamic textures which preserve the behavior of the system similar to its original appearance. There exist two fundamentally different approaches for dynamic texture synthesis: physics-based methods generate dynamic texture based on mathematical models of the natural phenomena, see e.g. [86, 83]. Physics-based models provide a flexible synthesis, since the dynamic texture can be controlled through a few parameters in a mathematical equation system. The drawback

of this approach is that each model is appropriate only for a particular texture and cannot be transferred to other domains. As an alternative, image-based models overcome this limitation. They use a global model for different textures and synthesize dynamic textures from a model based on the appearance of the whole texture in a series of images. Different principled approaches can be distinguished such as simple extensions of static texture synthesis to 3D [87], spatiotemporal models based on the pixel level [85], or dynamical models on the image level, such as proposed in [78, 79]. The latter approach is particularly promising since it can capture common non-trivial dynamical development such as rotation. This way, dynamic texture synthesis becomes a problem of system identification based on a sequence of image data such that dynamic texture can be directly generated along the trajectory of the system based on given initial conditions.

Typically, image sequences possess a very high dimensionality such that system identification is not possible in the raw image space. Therefore, the approaches presented in [78, 79] first project the images onto low dimensional space with a standard principal component analysis (PCA), performing system identification e.g. using classical linear auto-regressive (AR) models in the low dimensional projection space, afterwards. Since PCA gives rise to an approximate inverse by means of the pseudo-inverse of the transformation matrix, dynamic textures can be generated this way. The resulting model is rather flexible, but it has the drawback that a global linear embedding is used such that images are not appropriately sampled and represented in particular at points in time with rapid movements (e.g. flapping flag). Therefore, it has been proposed e.g. in [80, 81] to use recent nonlinear manifold learning techniques as proposed in machine learning instead of a global linear embedding.

In this section, we demonstrate that matrix learning for neural gas together with manifold charting give rise to a nonlinear manifold embedding which can successfully

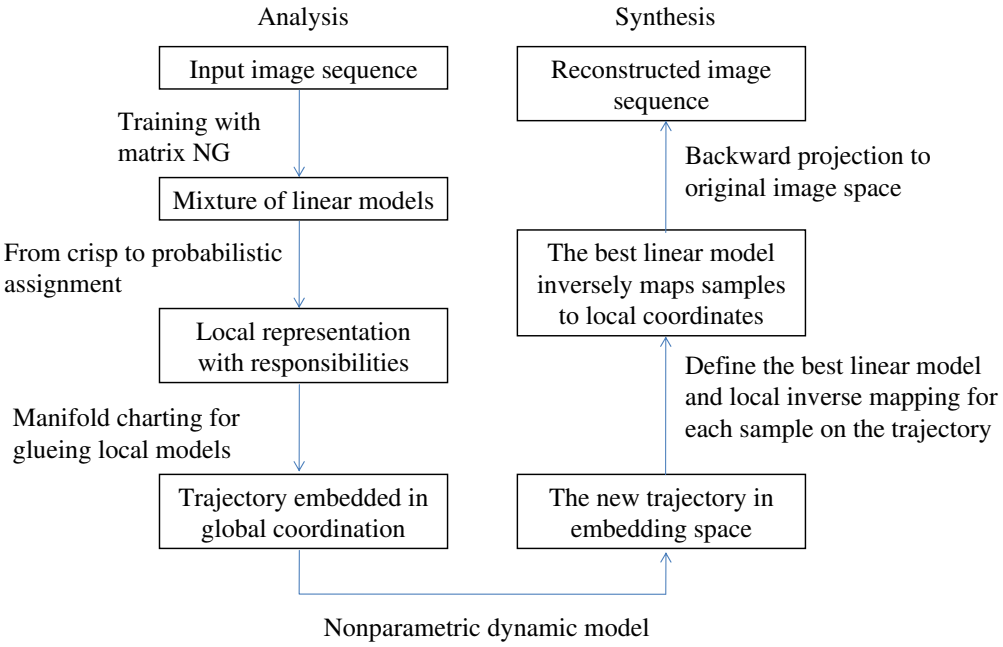


Figure 5.9: Diagram of our approach for dynamic texture analysis and synthesis of a given image sequence.

be used in this context. This way, neural low-dimensional nonlinear manifold embedding can serve as an essential step in the highly non-trivial application in computer graphics to automated dynamic texture synthesis.

Dynamic texture synthesis by system traversal

In [80, 81], dynamic texture synthesis is modelled as a system identification problem. First, every m-dimensional image, $m =$ number of pixels times number of color channels, is nonlinearly mapped to a lower d-dimensional space. In this work, we

choose manifold charting based on MNG and MoPPCA as nonlinear projection, see section 5.2. In low dimensions, a method to track temporal developments based on initial conditions is defined. Since every point in the embedding space can be inversely mapped to a point in the original high dimensional space using an approximate inverse as defined in equation (5.5), a sequence of images corresponding to a dynamic texture results. This way, a compressed representation of dynamic textures can be obtained since it is sufficient to store the parameters of the global nonlinear map and its inverse and only the low dimensional projections of the given image sequence which, for $d \ll m$, requires much less space than the original texture sequence. Further, interpolation of texture images as well as generation of texture based on new starting points becomes possible since a model is available to track the dynamics in the low dimensional projection space.

The contribution [81] relies on a mixture of probabilistic principal component analysis (MPPCA) together with global coordination to obtain a global nonlinear embedding of the data manifold [38, 67]. Here we propose to substitute MPPCA by matrix NG since, as we will demonstrate in experiments, a greater robustness and smoothness of the method can be achieved. Thus, we combine global coordination based on matrix NG as described in the previous section with the tracking dynamics in the low dimensional projection space as introduced in [81], where figure 5.9 depicts a step-by-step way of our approach.

For convenience, we shortly describe the nonparametric model based on traversal mechanism as proposed in [81]. Assume a dynamic texture is given which, making the temporal dependency explicit, is denoted as $\vec{x}(t)$, $\vec{x}(t+1)$, $\ldots \in \mathbb{R}^m$. The corresponding low-dimensional projections are denoted as $\vec{z}(t)$, $\vec{z}(t+1)$, $\ldots \in \mathbb{R}^d$. Motion prediction starts from a sequence of at least two points $\vec{g}(t-1)$, $\vec{g}(t)$ in $\mathbb{R}^d$ which are obtained as projections of images. Now the temporal successors of $\vec{g}(t)$ is obtained in

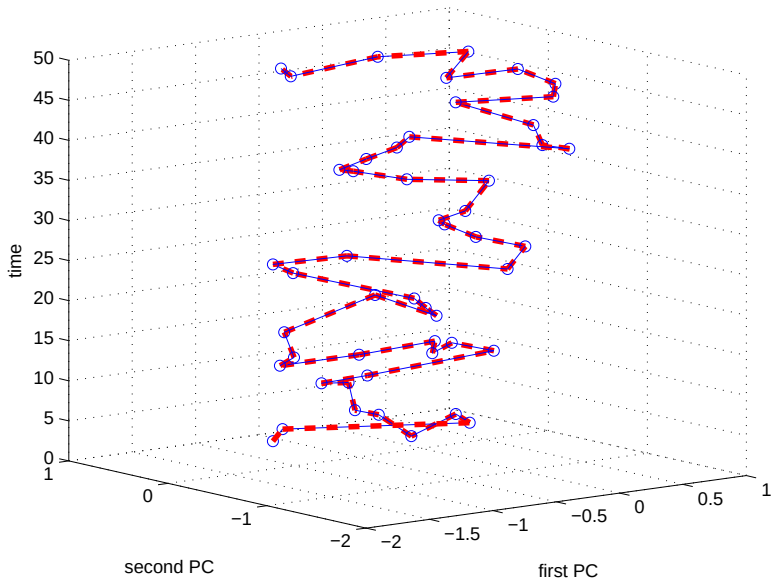


Figure 5.10: Using the nonparametric model as proposed in [81] to generate the new trajectory on the embedding space. The original and new trajectories are represented by the blue and red dash lines, respectively.

six steps based on the low dimensional vectors $\vec{z}(i)$ as follows (σ_1, α, σ_2 are positive parameters):

1. Sampling neighbors: K nearest neighbors N of $\vec{g}(t)$ are sampled from the data $\vec{z}(i)$ and exponentially weighted according to the distance from $\vec{g}(t)$ with weight $W_i^1 := \exp(-\|\vec{g}(t) - \vec{z}(i)\|^2/\sigma_1^2)$.

2. Temporal smoothness: The similarity of the difference vectors $d\vec{z}(i) := \vec{z}(i) - \vec{z}(i-1)$ of the neighbors $\vec{z}(i)$ in N and the considered trajectory $d\vec{g}(t) := \vec{g}(t) - \vec{g}(t-1)$ is computed based on the cosine distance and exponentially weighted, yielding weight $W_i^2 := \exp(\alpha(d\vec{z}(i)^t d(\vec{g}(t))/(\|d\vec{z}(i)\| \cdot \|d\vec{x}(t)\|) - 1))$.

3. Noise perturbation: for every neighbor $\vec{z}(i)$ in N, noisy successors of $\vec{g}(t)$ are sampled using the direction of the trajectory at $\vec{z}(i)$ and a Gaussian noise vector $\vec{\nu}_j$ with components $\sim N(0, \sigma_2)$ leading to possible positions $\vec{g}^{ij}(t+1) = \vec{g}(t) +$

$(\vec{z}(i+1) - \vec{z}(i)) + \vec{\nu}_j$.

4. Drift prevention: Each candidate is weighted according to its distance from the trajectory leading to the weight $p(\vec{g}^{ij}(t+1)) = W_i^1 W_i^2 \sum_l \varphi((\vec{g}^{ij}(t+1) - \vec{z}_k)/h)$ where φ is a window function with window width h.

5. Normalization: These weights are normalized such that $\sum_{ij} p(\vec{g}^{ij}(t+1)) = 1$.

6. Prediction: The successor is chosen from these points according to the probability $p(\vec{g}^{ij}(t+1))$.

This way, the overall direction of the trajectory gives rise to the respective successor of a given starting position. Slight noise accounts for typical effects when dealing with natural phenomena. An additional neighborhood integration makes sure that the created trajectory does not diverge from the dynamics as determined by the given data set. The main idea of using a nonparametric model is to traverse along the trajectory of the given data set embedded using global coordination. This method resembles curve tracing of a given trajectory. The estimated trajectory is obtained by superposition of the original one. In figure 5.10, the image sequence is projected onto 2D coordinates using manifold charting. The set of blue circles corresponds to the location of each image in the embedding space. The blue line connects each frame in the original order. We add a time axis to the 2D embedding space to consider the smoothness and accuracy of the synthesized frames on the trajectory. The new trajectory represented with the red dashed line is a good approximation which does not drift far away from the original trajectory.

Modelling of natural phenomena

An important example of dynamic texture is given by image sequences of natural phenomena as available in the DynTex database [84]. Each pixel gradually changes its

intensity or color level depending on the kind of image sequence. Examples of natural phenomena used in this work are referred to as wave, escalator, smoke, straw, and fall. All images have an original size of 288 by 352 pixels with RGB color codes. Because of the high dimensionality, we resized the images to 50% with respect to the original size resulting in 144 times 176 pixels. Then, each image corresponds to a 76,032 dimensional vector formed by the RGB values of the pixels. Before applying matrix NG, data were projected to 100 dimensions using simple principal component analysis.

We compare the result of matrix NG and mixtures of probabilistic component analysis as described in [38] together with global manifold charting and trajectory traversal as described above. The dimensionality d has been chosen as 40. The number n of local components is chosen such that, on average, about 50 frames are represented by one local model. The lengths of the considered dynamic textures and the number of local models is shown in Tab. 5.5.

We evaluate the method by the mean absolute distance of the generated images and the original images averaged over time, as shown in Tab. 5.6. Further, we exemplarily show a visual comparison of the images as obtained by MNG and MPPCA in comparison to the original image in Figs. 5.12 - 5.15. The synthesis results indicate that manifold embedding based on MNG is able to generate high-quality video, while charting based on MPPCA generates lower visual quality of video over time. The synthesized image sequences by MNG are smooth with respect to temporal evolution due to the included neighborhood cooperation, and sharp features are better preserved in the single images. MPPCA contains blurring in single images and a larger trend towards discontinuities when generating image sequences. This manifests in a larger error of the generated images. The development of the absolute error over time per pixel is depicted in Fig. 5.11. Obviously, for MPPCA, the error is not uniformly distributed but it accumulates at points in time such that errors are clearly observable for

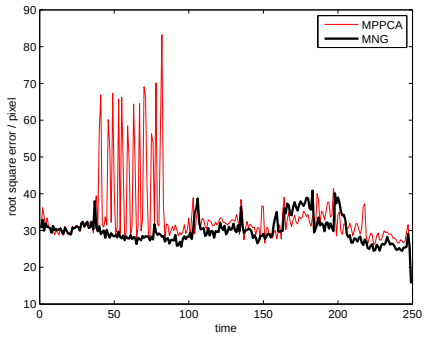
MPPCA. In comparison, the error of MNG is very smooth such that no abrupt changes in the visual appearance can be observed.

Dataset	no. local models	no. frames
wave	3	200
escalator	4	251
smoke	5	251
fall	2	200
straw	4	251

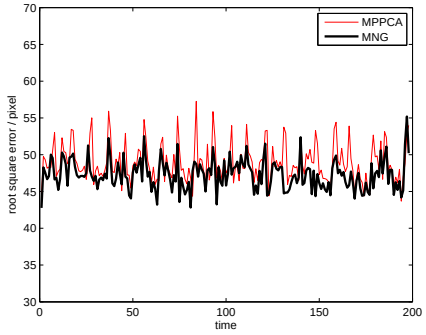
Table 5.5: Number of local models used to map the respective dynamic texture into low dimensional space and number of frames included in the dynamic textures.

Dataset	MPPCA	MNG
wave	33.7319 ± 2.7425	31.3815 ± 0.5540
escalator	30.8114 ± 2.0043	24.5836 ± 0.1290
smoke	14.2246 ± 0.5515	12.5402 ± 0.2278
fall	48.4483 ± 0.8815	47.3001 ± 0.0537
straw	37.3818 ± 0.5610	36.7924 ± 0.1960

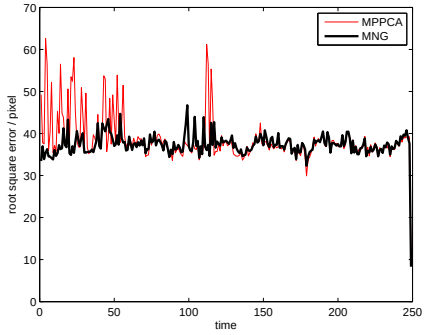
Table 5.6: Average absolute reconstruction errors averaged over the number of frames on the given image sequences and standard deviations.



(a) wave



(b) fall



(c) straw

Figure 5.11: Absolute error per pixel between the generated images and the true image sequence over time for MPPCA and MNG for the wave texture (top), fall texture (middle), and straw texture (bottom). Obviously, the error obtained by MPPCA is large for some time points in which a low quality of the reconstructed texture can visually be observed.

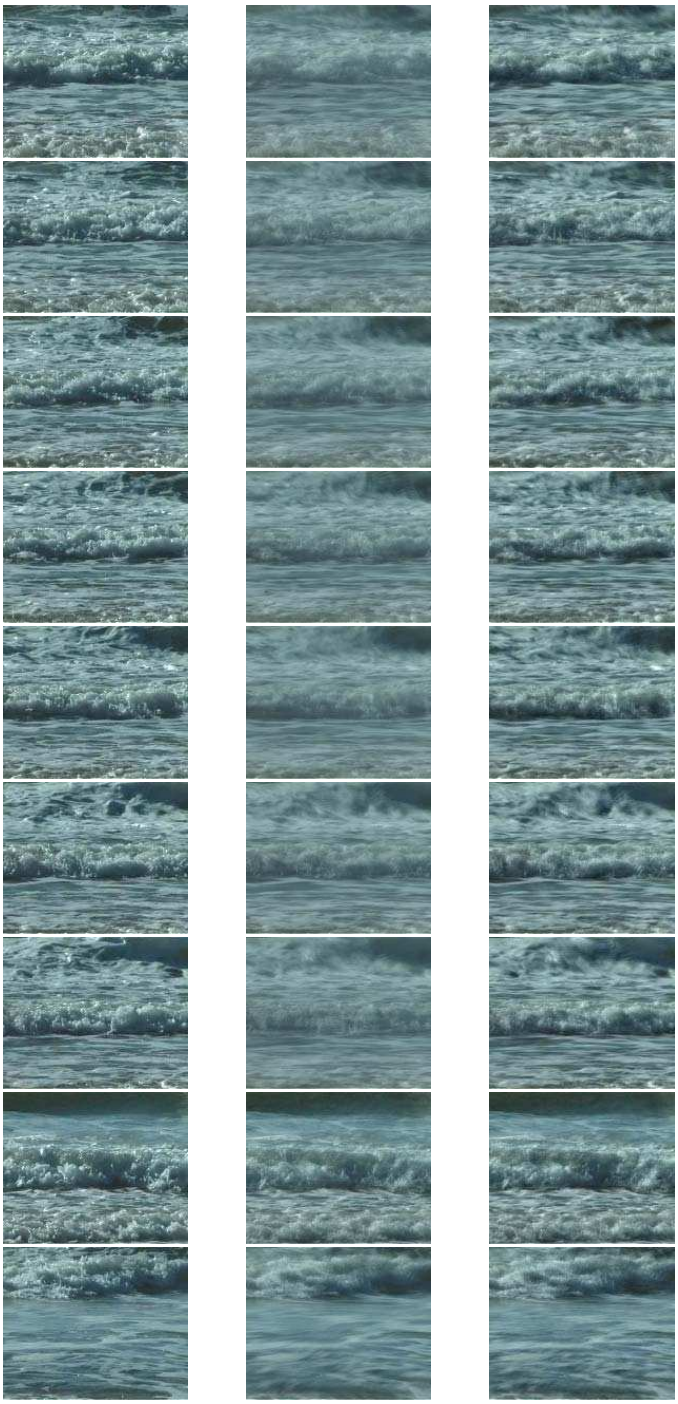


Figure 5.12: This figure shows reconstructed image sequence of waves. The first column represents the original reference frames. The second and third columns demonstrate the frames reconstructed by manifold charting based on MPPCA and matrix NG, respectively.

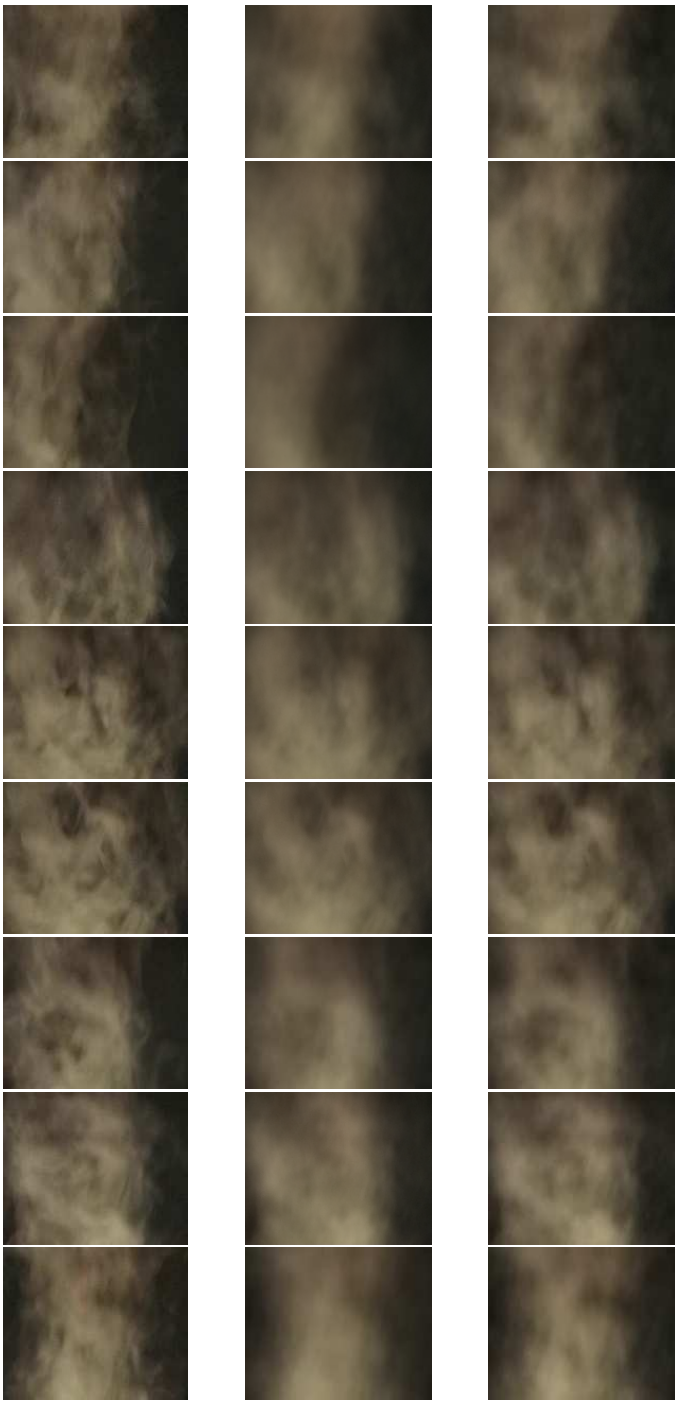


Figure 5.13: This figure shows reconstructed image sequence of smoke. The first column represents original reference frames. The second and third columns demonstrate the frames reconstructed by manifold charting based on MPPCA and matrix NG, respectively.

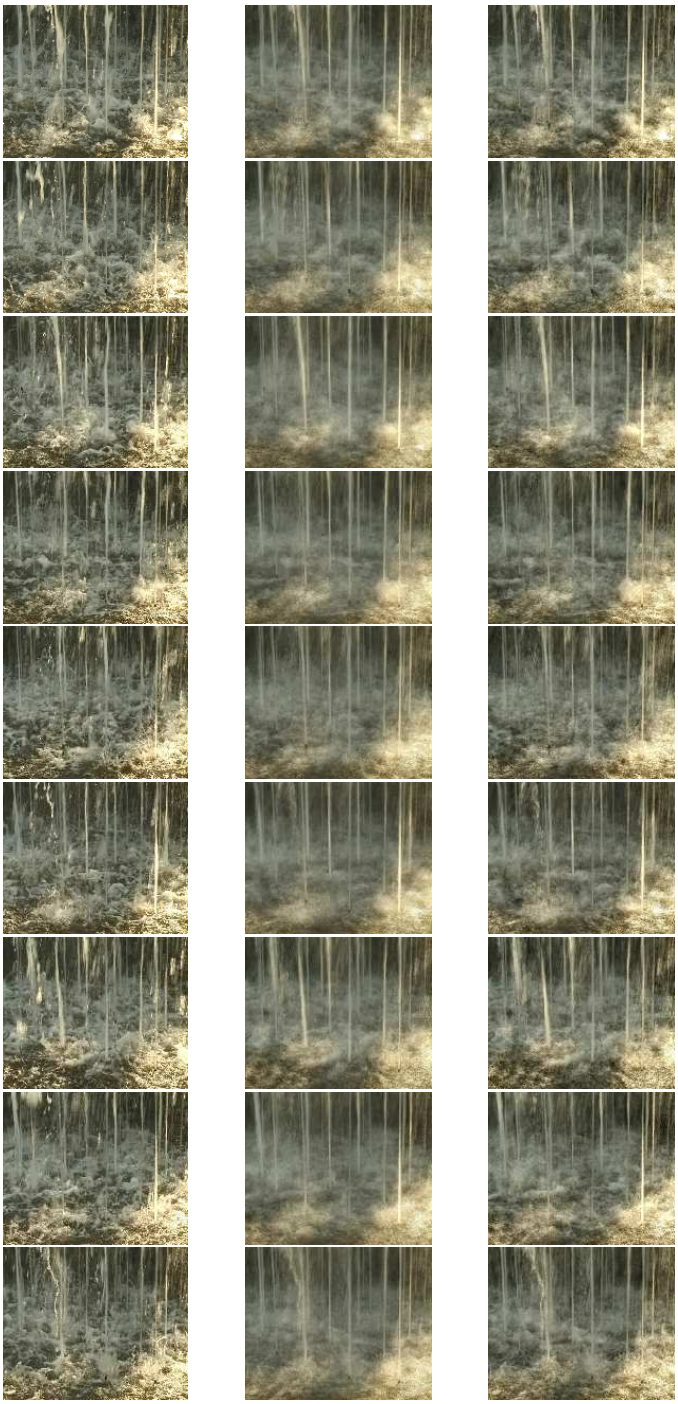


Figure 5.14: This figure shows reconstructed image sequence of fall. The first column represents original reference frames. The second and third columns demonstrate the frames reconstructed by manifold charting based on MPPCA and matrix NG, respectively.

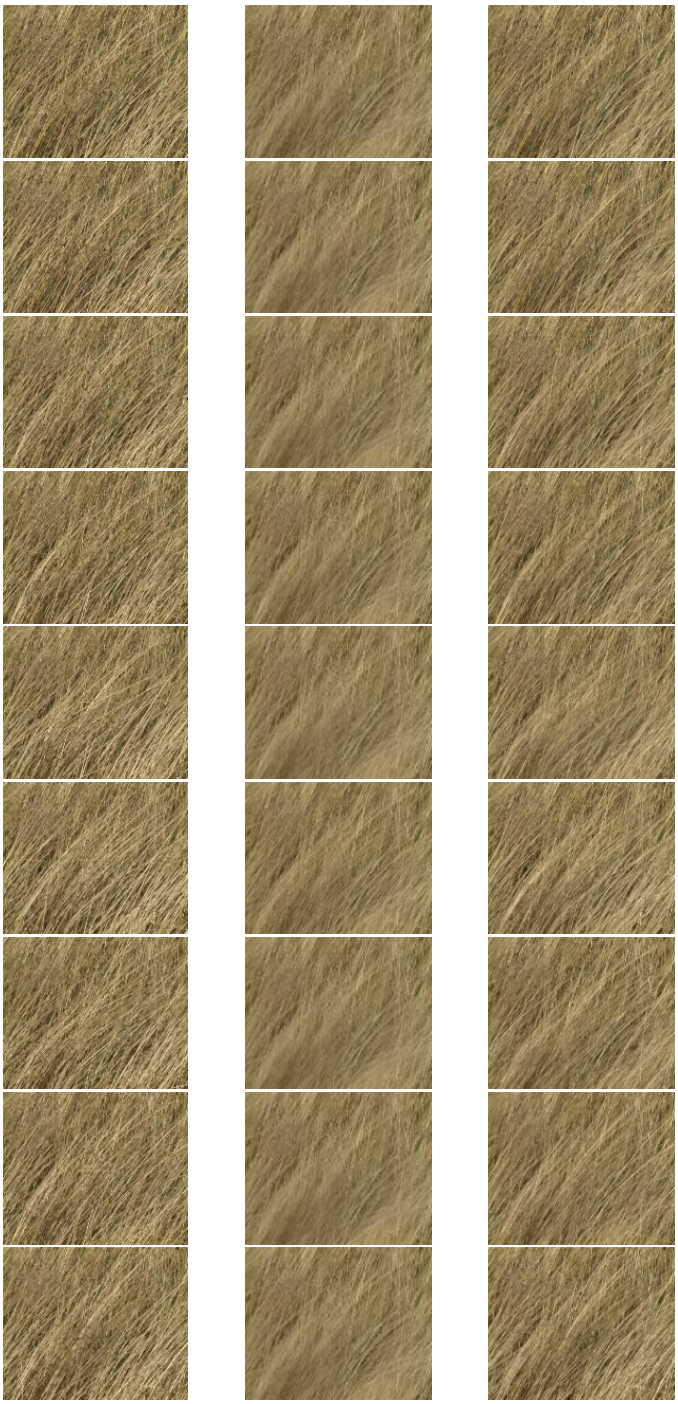


Figure 5.15: This figure shows reconstructed image sequence of straw. The first column represents original reference frames. The second and third columns demonstrate the frames reconstructed by manifold charting based on MPPCA and matrix NG, respectively.

5.5 Chapter summary

In this chapter, the focus was put on the possibility to further extend the resulting clustering to a global visualization and data projection method by combining these locally linear PCA projections with a global charting step. Using ideas of manifold charting, a global mapping of high dimensional data to lower dimensions can easily be achieved which preserves the original idea of prototype-based representation of data, i.e. it provides a data compression method besides visualization.

We evaluated the results of data visualization using a simple 1-NN classifier as well as a recently proposed trustworthiness measure, comparing matrix clustering on several benchmarks to standard methods including LLE, Isomap, and an alternative prototype-based method (mixtures of probabilistic PCA). Unlike Isomap and LLE, MNG and charting leads to a global embedding of data which is based on a locally linear representation of the data space by means of prototypes and clusters. Matrix neural gas turned out to be a very robust data projection method displaying reasonable results on both, artificial and real life data sets. Matrix clustering can also be used as the basis of other global coordination models. Nonlinear projection based on MNG is appropriate to be the internal representation of very high dimensional data. Thereby, MNG provides a natural clustering as well as data visualization.

Furthermore, charting based on MNG provides an explicit embedding function as well as its approximate inverse. The latter leads to ability of inference to original data space from low dimensional space. This advantage of approximate inverse is verified in generation of dynamic textures through system identification on embedding space. The reconstruction error by means of the approximate inverse shows good agreement to the original manifold also at the borders of the local linear pieces. We have demonstrated that MNG as basis of global coordination model, together with a traversing technique in low dimensional representation, can successfully be used in a

complex task from computer graphics, namely the synthesis of dynamic textures based on given image sequences.

CHAPTER 6

Conclusion and outlook

6.1 Thesis summary

In this thesis, we introduced an extension of topographic neural maps such as NG and SOM. Originally, standard NG and SOM use the Euclidean metric as distance measure. The resulting clusters from these methods are formed by isotropic shapes with spherical boundaries. By using Euclidean distance, the methods are unadjustable to fit complex and nonlinear data. To defeat this restriction, NG and SOM need to be improved by changing the underlying distance measure.

MNG as local PCA

We extended prototype-based clustering to adaptive matrix learning schemes. Unlike existing approaches in the literature, we developed our methods based on a cost function such that it was possible to also prove convergence of the algorithm. The resulting batch learning scheme makes it possible to link the method to local PCA such as proposed in the literature. We demonstrate the usefulness of the approach in a few clustering and classification benchmarks. Further, the method has been integrated into these applications

- matrix learning for image compression schemes,
- matrix learning in combination with manifold charting for manifold visualiza-

tion and dimensionality reduction,

- matrix learning and charting with approximate inverse maps as a manifold capturing mechanism which, together with temporal tracking can be used for dynamic texture synthesis.

MNG for image compression

Since digital images contain large amounts of information, image compression is essential to store and transmit images efficiently. Successful compression techniques aim at a preservation of as much information as possible while reducing the necessary number of bits to represent the images. Classical compression techniques can be based on vector quantization or transformation based codings based on PCA, for example. We proposed to use matrix clustering as a transformation based encoding which represents images in a compact way leading to reasonable reconstructing which are superior to single PCA or VQ.

MNG for data visualization

Since electronic data sets are increasing rapidly with respect to both, size of the data sets and data resolution, i.e. dimensionality, adequate data inspection and data visualization have become central issues of data mining. The task of data visualization is directly related to nonlinear dimensionality reduction (NLDR) techniques. Normally, NLDR can be classified into two main schemes: (1) manifold embedding and (2) mixture of local linear models. A variety of methods exists such as ISOMAP, LLE, LLC, MC, etc. These, however, partially have the drawbacks that no explicit embedding function is obtained, the methods are sensitive to initialization, or they display a large complexity. In this work, we demonstrated the applicability of MNG to

low-dimensional data embedding for data inspection. For this purpose, we extended matrix learing to obtain a mixture of overlapping local linear map and combined this with charting techniques. The evaluation on a few artificial and real life benchmarks revealed the universal suitability of the approach for difficult data sets.

MNG for dynamic texture synthesis

Dynamic texture synthesis is the process of producing an animation of dynamic textures which preserve the behavior of the system similar to its original appearance. As discussed in this thesis, the problem of dynamic texture synthesis can be converted into a problem of dimensionality reduction and system identification. Matrix learning and manifold charting does not only provide an explicit low dimensional embedding function but also its approximate inverse. This fact has been used in the thesis, together with a method of tracking the temporal dynamics in the low dimensional embedding space, to obtain an efficient and quite successful generator for dynamic texture synthesis.

6.2 Future work

The proposed thesis developed a method to incorporate matrix learing into unsupervised clustering, leading to powerful local PCA schemes which can be used for various tasks such as clustring, image compression, manifold visualization, and system identification. One key feature of the proposed model is its mathematical foundation by means of an underlying cost function and a formal convergence proof. A number of issues remain the subject of ongoing work, as already partially discussed in the thesis.

So far, the method relies on matrix inversion which is computationally intractable and numerically instable for large dimensionality. A possible remedy has already been

discussed in the thesis: a restriction to only the most important dimensions. It remains to demonstrate the usefulness of this approach in practical applications as introduced in this thesis. Further, since the number of essential dimensions is not clear a priori, it would be worthwhile to develop methods which define an appropriate number of essential dimensions during the algorithm. Another critical parameter is given by the number of prototypes, n. For Euclidean counterpart such as k-means, partial solutions have been proposed which automatically determine this number based on statistical considerations such as X-means. A transfer of these techniques to matrix learning seems promissing.

Further, as demonstrated in a few applications, matrix learning can be used at every position where a gaussian mixture model should be inferred from the data. This open the way towards further interesting application areas such as image segmentation or eigenface recognition. Unlike explicit statistical models, however, the exact statistical properties of matrix learning are not clear and an evaluation of important statistical characteristics such as the magnification factor remain the subject of future work.

REFERENCES

[1] B. Arnonkijpanich and C. Lursinsap, "Adaptive Second Order Self-Organizing Mapping for 2D Pattern Representation," *IEEE Int. Joint Conference on Neural Networks 2004*, pp. 775-780, 2004.

[2] Asuncion, A. and Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[3] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithm," *Neural Information Processing Systems 1994, (G. Tesauro, D.S. Touretzky, and T.K. Leen eds., MIT)*, pp. 585-592, 1994.

[4] R. Cappelli, D. Maio, and D. Maltoni, "Multispace KL for pattern representation and classification," *IEEE Trans. on PAMI*, vol. 23, no. 9, pp. 977-996, 2001.

[5] M. Cottrell, J.C. Fort, and G. Pagès, "Theoretical aspects of the SOM algorithm," *Neurocomputing*, vol. 21, pp. 119-138. , 1999.

[6] G. Cottrell, P. Munro, and D. Zipser, "Learning Internal Representations from Grey-Scale Images: An Example of Extensional Programming," *Ninth Annual Conference of the Cognitive Science Society*, pp. 462-473, 1987.

[7] M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann, "Batch and median neural gas," *Neural Networks*, vol. 19, pp. 762-771, 2006.

[8] R.D. Dony and S. Haykin, "Optimally adaptive transform coding," *IEEE Transactions on Image Processing*, vol.4, pp.1358-1370, 1995.

[9] R.O. Duda, P.E. Hart, and D.G. Storck, "Pattern Classification," *Wiley*, 2000.

[10] D. Francois, V. Wertz, and M. Verleysen, "Non-Euclidean metrics for similarity search in noisy datasets," *European Symposium on Artificial Neural Networks 2005*, pp. 339-344, 2005.

[11] J.C. Fort, P. Letrémy, and M. Cottrell, "Advantages and drawbacks of the Batch Kohonen algorithm," *European Symposium on Artificial Neural Networks 2002, (M. Verleysen ed.)*, pp. 223-230, 2002.

[12] D.E. Gustafson and W.C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," *IEEE Conference on Decision and Control 1979*, pp. 761-766, 1979.

[13] B. Hammer and A. Hasenfuss, "Relational Neural Gas," *KI 2007, (J. Hertzberg et al., eds.)*, pp. 190-204, 2007.

[14] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," *Neural Networks*, vol. 15, pp. 1059-1068, 2002.

[15] J.A.Hartigan, "Clustering Algorithms," *Wiley*, 1975.

[16] T. Heskes, "Self-organizing maps, vector quantization, and mixture modeling," *IEEE Transactions on Neural Networks*, vol. 12, pp. 1299-1305, 2001.

[17] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, "Fuzzy Cluster Analysis," *Wiley, Chichester*, 1999.

[18] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "MindReader: Querying Databases Through Multiple Examples," *24th Int. Conf. Very Large Data Bases*, pp. 218-227, 1998.

[19] A. Kambhatla and T.K. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, pp. 1493-1516, 1997.

[20] T. Kohonen, "Self-Organizing Maps," *Springer*, 1995.

[21] T. Kohonen, S. Kaski, and H. Lappalainen, "Self-organized formation of various invariant-feature filters in the Adaptive-Subspace SOM," *Neural Computation*, vol. 9, no. 6, pp. 1321-1344, 1997.

[22] K. Krishna, K.R. Ramakrishnan, and M.A.L. Thathachar, "Vector quantization using genetic K-means algorithm for image compression," *Int. Conf. on Information, Communications and Signal Processing*, vol. 3, pp. 1585-1587, 1997.

[23] E. López-Rubio, J. Muñoz-Pérez, and J.A. Gómez-Ruiz, "A principal components analysis self-organizing map," *Neural Networks*, vol. 17, no. 2, pp. 261-270, 2004.

[24] L.J.P. van der Maaten, E.O. Postma, H.J. van den Herik (2007). Matlab Toolbox for Dimensionality Reduction. MICC, Maastricht University. [http://www.cs.unimaas.nl/l.vandermaaten/Laurens_van_der_Maaten/Matlab_Toolbox_for_Dimensionality_Reduction.html].

[25] T. Martinetz, S. Berkovich, and K. Schulten, "'Neural gas' network for vector quantization and its application to time series prediction," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp. 558-569, 1993.

[26] T. Martinetz and K.J. Schulten, "Topology representing networks," *Neural Networks*, vol. 7, pp. 507-522, 1994.

[27] P. Meinicke and H. Ritter, "Topographic Local PCA Maps," *Proceeding of the Symposia on Neural Computation*, 2000.

[28] R. Möller and H. Hoffmann, “An Extension of Neural Gas to Local PCA,” *Neurocomputing*, vol. 62, pp. 305-326, 2004.

[29] D. Mochihashi, G. Kikui, and K. Kita, “Learning an optimal distance metric in a linguistic vector space,” *Systems and computers in Japan*, vol. 37, no. 9, pp. 12-21, 2006.

[30] E. Oja, “Simplified neuron model as a principal component analyzer,” *J. Math. Biol.*, vol. 15, pp. 267-273, 1982.

[31] E. Oja, “Neural networks, principal components and subspaces,” *Int. J. Neural Syst.*, vol. 1, pp. 61-68, 1989.

[32] J. Peltonen, A. Klami, and S. Kaski, “Improved learning of Riemannian metrics for exploratory analysis,” *Neural Networks*, vol. 17, pp. 1087-1100, 2004.

[33] J.W. Sammon, “A nonlinear mapping for data structure analysis,” *IEEE Transactions on Computers*, vol. 18, pp. 401-409, 1969.

[34] T.D. Sanger, “Optimal unsupervised learning in a single-layer linear feedforward neural network,” *Neural Networks*, vol. 2, no. 6, pp. 459-473, 1989.

[35] F.M. Schleif, T. Villmann, and B. Hammer, “Prototype based fuzzy classification in clinical proteomics,” *International Journal of Approximate Reasoning*, vol. 47, no. 1, pp. 4-16, 2008.

[36] P. Schneider, M. Biehl, and B. Hammer, “Relevance matrices in LVQ,” *European Symposium on Artificial Neural Networks 2007*, pp. 37-42, 2007.

[37] L.Z. Stones, Y. Zhang, and C.L. Jian, “Growing Hierarchical Principal Components Analysis Self-Organizing Map,” *Third International Symposium on Neural Networks 2006, Springer LNCS*, vol. 3971, pp. 701-706, 2006.

[38] M.E. Tipping and C.M. Bishop, “Mixtures of probabilistic principal component analyzers,” *Neural Computation*, vol. 11, pp. 443-482, 1999.

[39] M.E. Tipping and C.M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society, Series B*, vol. 61, no. 3, pp. 611-622, 1999.

[40] B.L. Zhang, Q. Huang, and T.D. Gedeon, “A mixture of local PCA learning algorithm for adaptive transform coding,” *International Conference on Neural Information Processing*, vol. 3, pp. 842-847, 1999.

[41] S. Zhong and J. Ghosh, “A unified framework for model-based clustering,” *Journal of Machine Learning Research*, vol. 4, pp. 1001-1037, 2003.

[42] Z.-Y. Liu and L. Xu, "Topological local principal component analysis," *Neurocomputing*, vol. 55, pp. 739-745, 2003.

[43] Y. Wang, M.I. Freedman, and S.-K. Kung, "Probabilistic principal component subspaces: a hierarchical finite mixture model for data visualization," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 625-636, 2000.

[44] H. Yin and N.M. Allinson, "Self-Organizing Mixture Networks for Probability Density Estimation," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 405-411, 2001.

[45] C. Bouveyron, S. Girard, and C. Schmid, "High-dimensional data clustering," *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 502-519, 2007.

[46] R. Cappelli, D. Maio, and D. Maltoni, "Multispace KL for pattern representation and classification," *IEEE Trans. on PAMI*, vol. 23, no. 9, pp. 977-996, 2001.

[47] C. Ambroise and G. Govaert, "Analyzing Dissimilarity Matrices using Kohonen Maps," *International Federation of Classification Societies*, vol. 1, pp. 425-430, 1996.

[48] T. Kohonen and P. Somervuo, "Self-organizing maps of symbol strings," *Neurocomputing*, vol. 21, pp. 19-30, 1998.

[49] T. Graepel, M. Burger, and K. Obermayer, "Self-organizing maps: generalizations and new optimization techniques," *Neurocomputing*, vol. 21, pp. 173-190, 1998.

[50] S. Seo and K. Obermayer, "Self organizing maps and clustering for matrix data," *Neural Networks*, vol. 17, pp. 1211-1229, 2004.

[51] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE transactions on Neural Networks*, vol. 16, no. 3, pp. 645-678, 2005.

[52] M. Teboulle, "A unified continuous optimization framework for center-based clustering methods," *Journal of Machine Learning Research*, vol. 8, pp. 65-102, 2007.

[53] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2210-2239, 1998.

[54] T. Martinetz, "Selbstorganisierende neuronale Netzwerkmodelle zur Bewegungssteuerung," *Dissertationen zur künstlichen Intelligenz*, Sankt Augustin, 1992.

[55] T. Heskes and B. Kappen, "Error potentials for self-organization," *IEEE International Conference on Neural Networks*, vol.3, pp. 1219-1223, 1993.

[56] S. Luttrell, "Self-organisation: A derivation from first principles of a class of learning algorithms," *IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 495-498, 1989.

[57] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: ordering, convergence properties, and energy functions," *Biological Cybernetics*, vol. 67, pp. 47-55, 1992.

[58] T. Kohonen, "Self-organizing maps: optimization approaches," *Artificial Neural Networks, T. Kohonen et al. (eds.), Amsterdam, North-Holland*, pp. 981-990, 1991.

[59] T. Geszti, "Physical models of neural networks," *World Scientific*, 1990.

[60] R. Möller and A. Könies, "Coupled Principal Component Analysis," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 214-222, 2004.

[61] Y. Feng and G. Hamerly, "Pg-means: learning the number of clusters in data," *Advances in Neural Information Processing Systems*, pp. 393-400, 2006.

[62] G. Hamerly and C. Elkan, "Learning the k in k-means," *Advances in Neural Information Processing Systems*, pp. 281-288, 2003.

[63] T. Villmann, R. Der, M. Herrmann, and T. Martinetz, "Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 256-266, 1997.

[64] Matthew Brand, "Charting a manifold," *Advances in Neural Information Processing Systems*, vol. 15, pp. 961-968, 2003.

[65] L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik, "Dimensionality Reduction: A Comparative Review," *Neurocomputing*, submitted.

[66] Sam Roweis and Lawrence Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323-2326, 2000.

[67] Yee Whye Teh and Sam Roweis, "Automatic alignment of local representations," *Advances in Neural Information Processing Systems*, vol. 15, pp. 841-848, 2003.

[68] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319-2323, 2000.

[69] B. Arnonkijpanich, B. Hammer, and A. Hasenfuss, “Local matrix adaptation in topographic neural maps,” *Neurocomputing*, submitted.

[70] Z. Gharamani and G. Hinton, “The EM algorithm for mixtures of factor analyzers,” *Technical Report CRG-TR-96-1, University of Toronto*, 1996.

[71] J. Venna and S. Kaski, “Visualizing gene interaction graphs with local multidimensional scaling,” *European Symposium on Artificial Neural Networks 2006*, pp. 557-562, 2006.

[72] P. Demartines and J. Herault, “Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets,” *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 148-154, 1997.

[73] L. Yang, “Sammon’s Nonlinear Mapping using Geodesic Distances,” *IEEE International Conference on Pattern Recognition*, vol. 2, pp. 303-306, 2004.

[74] M. Lennon, G. Mercier, M.C. Mouchot, and L. Hubert-Moy, “Curvilinear component analysis for nonlinear dimensionality reduction of hyperspectral images,” *Proceeding of SPIE*, vol. 4541, 2002.

[75] J.A. Lee, A. Lendasse, and M. Verleysen, “Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis,” *Neurocomputing*, vol. 57, pp. 49-76, 2004.

[76] T.F. Cox and M.A.A. Cox, “Multidimensional Scaling,” Chapman and Hall, 2001.

[77] J.A. Lee and M. Verleysen, “Nonlinear dimensionality reduction,” *Information Science and Statistics series*, Springer, 2007.

[78] S. Soatto, G. Doretto, and Y.N. Wu, “Dynamic Textures,” *IEEE International Conference on Computer Vision*, vol. 2, pp. 439-446, 2001.

[79] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, “Dynamic Textures,” *Int. Journal of Computer Vision*, vol. 51, no. 2, pp. 91-109, 2003.

[80] C.B. Liu, R.S. Lin, and N. Ahuja, “Modeling Dynamical Textures Using Subspace Mixtures,” *IEEE International Conference on Multimedia and Expo 2005*, pp. 1378-1381, 2005.

[81] C.B. Liu, R.S. Lin, N. Ahuja, and M.H. Yang, “Dynamic Textures Synthesis as Nonlinear Manifold Learning and Traversing,” *British Machine Vision Conference 2006*, vol. 2, pp. 859-868, 2006.

[82] B. Arnonkijpanich, A. Hasenfuss, and B. Hammer, “Local matrix learning in clustering and applications for manifold visualization,” *Neural Networks*, to appear.

[83] Y.Y. Chuang, D.B. Goldman, K.C. Zheng, B. Curless, D.H. Salesin, and R. Szeliski, “Animating pictures with stochastic motion textures,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 853-860, 2005.

[84] R. Peteri, M. Huiskes, and S. Fazekas. The DynTex database Homepage [http://old-www.cwi.nl/projects/dyntex/database.html]. Amsterdam, The Netherlands: Centre for Mathematics and Computer Science.

[85] M. Szummer and R.W.Picard, “Temporal texture modeling,” *IEEE International Conference on Image Processing*, vol. 3, pp.823-826, 1969.

[86] A. Treuille, A. McNamara, Z. Popovic, and J. Stam, “Keyframe control of smoke simulations,” *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 716-723, 2003.

[87] L.-Y. Wei and M. Levoy, “Fast texture synthesis using tree-structured vector quantization,” *Proceedings of ACM SIGGRAPH 2000*, pp.479-488, 2000.

[88] B. Arnonkijpanich and B. Hammer, “Global Coordination based on Matrix Neural Gas for Dynamic Texture Synthesis,” *IAPR International Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 84-95, 2010.

[89] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, pp. 386-408.

[90] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533-536, 1986.

[91] T. Kohonen, “Self-organizing formation of topologically correct feature maps,” *Biological Cybernetics*, vol. 43, pp. 56-69.

[92] T. Kohonen, “Analysis of a simple self-organizing process,” *Biological Cybernetics*, vol. 44, pp. 135-140.