Unsupervised Kernel Regression for Nonlinear Dimensionality Reduction

Diplomarbeit an der Technischen Fakultät
der Universität Bielefeld

Februar 2003

Roland Memisevic

Betreuer:

Prof. Helge Ritter
Universität Bielefeld
AG Neuroinformatik
Universitätsstr. 25
33615 Bielefeld

Dr. Peter Meinicke
Universität Bielefeld
AG Neuroinformatik
Universitätsstr. 25
33615 Bielefeld

Contents

1. Introduction

This thesis investigates the recently proposed method of *'Unsupervised Kernel Regression'* (UKR). The theoretical placement of this method as a means to pursue *Nonlinear Dimensionality Reduction* (NLDR) is analyzed, the technicalities involved with its practical implementation are inspected, and its applicability to real world problems is explored.

The UKR method stems from the area of *Machine Learning* which is concerned with the development of algorithms that discover *patterns in data*. More specifically, in relying on the key idea to let a system *learn from examples* what is important for a specific task, within this area methods are being developed that help to classify, detect, manipulate and produce patterns in a wide range of areas. These methods are not only of increasing practical interest, as they may be used to cope with the ever growing amount of data available in electronic form today, but also play an important role in the area of *Artificial Intelligence*, where they act as models of those mechanisms that lie at the heart of our own cognitive capabilities.

The methods developed in the area of Machine Learning can be divided into two broad classes. Those belonging to the first are concerned with *generalizing knowledge* presented by means of examples to new, unseen data. Inspired by biological learning, where any kind of knowledge residing in the examples needs to be pointed at by a teacher who corrects and thereby adjusts the learning system, the respective area is generally referred to as *Supervised Learning* (SL). It is contrasted and complemented by *Unsupervised Learning* (UL), which aims at developing systems that in the absence of prior knowledge automatically discover meaningful information hidden in the example

data. These methods thereby achieve the goal of accounting for the *variability* in the data and to provide *alternative representations*. Besides providing some kind of preprocessing which is often crucial to simplify a subsequent SL task these systems give rise to many further applications. Some of these will be sketched throughout. It is this second class of methods that UKR belongs to.

The two main tasks of UL can be defined as *dimensionality reduction* and *density estimation*. The UKR method may be cast in two distinct ways leading to a variant that pursues dimensionality reduction and a second variant that includes some kind of density estimation. This thesis examines only the first variant, and brief references to the closely related second method will be made at the appropriate places.

1.1. Dimensionality Reduction

Generally, methods in dimensionality reduction discover more compact representations of their input data, while at the same time they try to keep the usually resulting information loss at a minimum. These methods thereby minimize the required storage space or bandwidth capacity for saving or transmitting the data, which gives rise to some of their most widespread applications. Additionally, the resulting representations are often expected to capture the meaning inherent in the data more explicitly. This provides the basis for applications such as denoising and visualization and has also led to an increasing interest of the area in *computational neuroscience* for these methods, where the awareness that many higher level cognitive capabilities are not possible without some kind of dimensionality reduction is common grounds.

The basis for the reasoning adopted in a large body of methods in Machine Learning in general and in dimensionality reduction virtually exclusively is to represent the input and output data for these methods as sets of real valued vectors. For some given set of input vectors dimensionality reduction then amounts to computing an equally sized set of output vectors with lower dimensionality that fits the meaning of the input data set as closely as possible. Since the lower dimensional representations in many circumstances can be thought of as representing the 'real' or 'original' meaning of the data more closely than the (often noisy) input data, they will formally be denoted by the letter x, whereas

the vectors that are the input to the respective algorithm will be written y, although the converse notation can often be found in the literature.

For a given set of N input vectors $y_i \in R^d, i = 1, \ldots, N$, the problem of dimensionality reduction is then defined as that of finding a corresponding set of output vectors $x_i \in R^q, i = 1, \ldots, N$, a mapping $f : \Re^q \to \Re^d$ and a mapping $g : \Re^d \to \Re^q$ such that $\forall i = 1, ..., N$,

$$g(y_i) = x_i \tag{1.1}$$

$$f(x_i) = \hat{y}_i \approx y_i \tag{1.2}$$

(see e.g. [CG01]). The function f will also be referred to as 'forward mapping' or 'coding function' and g to as 'backward mapping' or 'decoding function.' The output variables are usually referred to as 'scores' or 'features' in the literature. Here, the latter will additionally be denoted 'latent space realizations,' or simply 'latent variables' with reference to chapter 2. Using the common Machine Learning jargon, the process of determining the scores and estimating models for the involved functions from the input data will be referred to as *training* and the set of input data to as *training data*. Depending on the application at hand, not all of the three sub-problems of dimensionality reduction necessarily need to be solved. If, for example, the aim of the dimensionality reduction task is to obtain a representation amenable to convenient *visualization* of a given dataset, only this representation – in other words, only a set of suitable scores – is needed. If the purpose is *noise reduction* or any kind of *pattern detection*, models for both the coding and decoding function become necessary in addition. Only the presence of a model for f is demanded if any kind of *pattern production* is aimed at.

In those cases in which a model for f or g is needed, the concept of *generalization* becomes crucial. The problem of generalization in general refers the fact that usually only a finite sized training data set is available for adapting a model, which shall afterwards be applied to *new* data not in this set. The optimization of the *expected* performance on the unseen data is the actual objective of the learning task. As a means to avoid adapting to noise present in the input data, which obviates good generalization and is usually referred to as *overfitting*, generally the 'flexibility' of the model is restrained

by some kind of *complexity control*. The determination of a suitable complexity control, referred to as *model selection* in the literature, is then conducted along with the training of the actual model and has to make use of the available data in some way. A widely used approach to model selection which plays an important role in this thesis is *cross validation*. It denotes the method of partitioning the available data into *training* and *test* sets and using the first for adapting a model using some specific complexity and the second to assessing the resulting performance and adjusting the complexity as required. An iteration of this procedure using different training set/test set partitionings may be used to improve reliability of the overall outcome. In the special case of the training set comprising $N - 1$ elements and test set 1 element giving rise to N train/test iterations this is referred to as *leave-one-out cross validation*.

Generally, the notion of generalization is closely connected to the notion of a test set and the error that a trained model gives rise to on this test set. Since in UL a test set only contains input elements, the term generalization here usually concerns some projection error in the input space. However, following [RS00], in this thesis the term generalization will be used in a somewhat broader sense and will denote the general problem of applying the forward or the backward mapping to new input or output elements.

1.2. Nonlinear Dimensionality Reduction / Overview

The oldest and best understood method for dimensionality reduction is *Principal Component Analysis* (PCA), which is based on the spectral decomposition of the data covariance matrix as described in detail below. As a *linear* model, PCA has some important advantages over many of the *nonlinear* models discussed in this thesis, in particular with regard to generalization.

Many *nonlinear* generalizations to PCA have been proposed. A broad class of these nonlinear models, that will be referred to as 'projection models' in the following and that in particular capture some of the generalization properties of PCA, can be represented in a unified way within the 'Generalized Regression Framework.' This framework also

builds the starting point for derivation of UKR. This general framework for UL shall be delineated in chapter 2. The second class of methods in UL – that include some kind of *density estimation* as described above – will be referred to as 'generative models' in the following. These models are also captured within this framework. Since the generative part of the generalized regression framework also builds the starting point for a generative variant of UKR (which is not subject of this thesis, however), it will also be sketched in chapter 2.

Besides these methods there is a second broad class of methods that have been proposed within the last years which are not captured within this general framework and might therefore be conceived of as *heuristic* methods. Virtually all of these rely on a spectral decomposition of some data proximity matrix that gives rise especially to an efficient computation of latent space realizations of the input data. As these *nonlinear spectral methods* in contrast to PCA do not allow for a straightforward estimation of the involved *mappings* they have specific problems with regard to generalization. These kinds of methods will be described in detail in chapter 3.

The UKR method for NLDR can be posed in two ways giving rise to models belonging to the 'projection' and to the 'spectral' class, respectively. Both will be described in chapter 4. Chapter 4 will also present practical considerations regarding their implementation in detail and illustrate this with several data sets. In addition, issues regarding generalization will be dealt with in some detail, and in particular a solution to combine the generalization capability of the projection models with the efficiency of spectral models will be proposed.

In chapter 5 some applications of UKR used as a method to perform nonlinear dimensionality reduction will be presented, where in particular the different prospects arising from latent space realization, applicability of f and applicability of g will be focused on.

Chapter 6 gives a review and summary of the issues dealt with in this thesis.

1.3. Conventions and Notation

In the following, scalar and vector valued variables will be denoted by lowercase italic letters, e.g. x. Matrices will be denoted by uppercase italic letters, e.g. X. Only real valued vectors and matrices will be used. The set of N d-dimensional input vectors y_i and the set of N q-dimensional output vectors x_i, $i = 1, \ldots, N$, will be represented by the $d \times N$ matrix Y and by the $q \times N$ matrix X, respectively.

A vector of ones will be denoted $\mathbf{1}$, a vector of zeros $\mathbf{0}$. In the cases where these notations are used, the dimensionality of these vectors will always be obvious from the context. The *mean* of some dataset represented for example by the matrix Y is defined as $Y\mathbf{1}$. The dataset Y will be said to be *mean centered* if it holds that $Y\mathbf{1} = \mathbf{0}$.

2. Unsupervised Learning as Generalized Regression

The UKR method investigated in this thesis arises naturally as a nonparametric instance from the 'Generalized Regression' framework for Unsupervised Learning[Mei00]. In this chapter this framework for Unsupervised Learning is be briefly reviewed, an overview over the two kinds of models it gives rise to is given, which are *Projection Models* and *Generative Models*, and it is shown how some of the known algorithms in Unsupervised Learning are reflected within this framework.

2.1. Conventional Regression

The purpose of regression is to model a functional relationship between random variables that is assumed to be made up of a systematic part and some unpredictable, additive noise. More precisely, let f denote a function that maps an input random vector $x \in \Re^q$ onto an output random vector $y \in \Re^d$ and let $u \in \Re^d$ denote some (zero mean) random vector that corresponds to the unpredictable noise. The relationship to be modeled is then assumed to be given by

$$y = f(x) + u, E(u) = 0. \tag{2.1}$$

To this end an approximation f^* to f is chosen from a set of candidate functions with respect to the objective to minimize the *expected prediction error*:

$$f^* = \arg\min_f \int \int \|y - f(x)\|^2 p(x, y) dx dy. \tag{2.2}$$

Under appropriate circumstances the function f^* that satisfies this demand can be shown to be given by the conditional expectation [HTF01]:

$$f^* = E[y|x] = \int y p(y|x) dy. \tag{2.3}$$

It is referred to as the *regression function* and its range as the *regression manifold*. In practice, f^* needs to be estimated from finite data sets for the input and output variables. One way this can be achieved is by replacing 2.2 with the *empirical prediction error*

$$E_N = \sum_{i=1}^{N} \|y_i - f(x_i)\|_2^2 \tag{2.4}$$

and minimizing this functional with respect to the function parameters. The input variables in this case are practically treated as fixed parameters instead of as random vectors.

Another option is to take into consideration a parameterized model $p(y|x; \theta)$ of the conditional probability density function of the output variables, given the input variables, for some parameter vector θ. Under the assumption of the sample elements to be drawn independently and all from the same distribution, the product density function specifies the sample probability density, so that the estimation of f can be realized by maximization thereof with respect to the function parameters. In practice equivalently the negative logarithm,

$$L_N = -\sum_{i=1}^{N} \log p(y_i|x_i; \theta), \tag{2.5}$$

is minimized, which gives rise to a learning principle generally referred to as *maximum likelihood*.

Yet another option that will not be used any further in this thesis, but which is mentioned here for the sake of completeness, is to regard the parameters as random variables themselves and learning as an update of their probability distribution, known as *Bayesian learning* [HS89].

2.2. Unsupervised Regression

The task of UL can be approached by utilizing a modified version of the regression model. As detailed in [Mei00] the difference then lies in the usage of the input variables. In the supervised case regression amounts to the estimation of a functional relationship utilizing a sample set for the input and their related output variable realizations. In UL the input variable realizations are conceived of as *missing* and therefore in the need to be estimated *together* with the functional relationship. The distinction can be expressed by referring to the input variables in an unsupervised setting as *latent* variables[1].

With regard to the practical learning task two important differences to the supervised case arise from the use of latent variables, the first of them affecting the definition of the learning problem: Since the input variables are not given in advance, one has to decide on a suitable *domain* for them. For that purpose several distinctions of the type of latent variables have to be taken into account, all leading to different types of regression manifolds. An important distinction is between *deterministic* and *random* latent variables leading to models referred to as *projection models* and *generative models*, respectively. Another distinction of the latent variable types is between *continuous* and *discrete* ones. Together with the option of choosing a class of candidate functions, where in particular the distinction between linear and nonlinear functions is of interest, the two dimensions along which a classification of the latent variables is possible, allow for the formulation of a wide spectrum of models of UL known from the literature. Some of these will be sketched in the subsequent sections.

The second novelty of unsupervised regression as compared to the supervised case

[1] The term *latent variable* has a longstanding history in statistical modeling and is closely related to the way it is used here. The informal definition given here is completely sufficient and self-contained with regard to the way this term will be used in the following.

regards the necessity to make use of some kind of *learning scheme*. Since the more ambitious goal of finding latent variable realizations *in addition* to parameters defining a suitable functional relationship needs to be tackled here, one has to conceive of a way to accomplish these tasks simultaneously. For deterministic latent variables a generally applicable approach to achieve this twofold objective is the 'Projection - Regression - Scheme.' It is obtained from an iteration of a 'Projection' - Step, used to find optimal values for the latent variables, given some values for the function parameters, and a 'Regression' - Step, used to re-estimate function parameters, while the values for the variables are kept constant. This optimization scheme can be thought of as a deterministic analog to the well known EM-algorithm, which can be applied in case of a generative model. Both, projection and generative models, the use of their respective learning schemes and examples of their applications will be described in detail below.

2.3. Optimization

As indicated above, within the Generalized Regression framework learning in general will be achieved by an iteration of a minimization of 2.4 or 2.5 with regard to the function parameters and an update of the latent variable realizations. However, the presence of a usually very large number of parameters in UL, owing to the fact that the latent variables need to be estimated here, as well, often cause the respective objective functions to be fraught with local minima. Therefore, unless a closed form solution for the concerned functions exists, the success of the learning task depends crucially on the *initialization* or - if no indication regarding auspicious areas in parameter space is available - on the use of an optimization *strategy* that helps to avoid or at least to diminish the chance of getting trapped in a local minimum.

The well-known method of *Simulated Annealing* tries to achieve this by allowing for random influences to appeal during the parameter update in an iterative optimization process. This alleviates the chance of getting stuck in a local minimum. Gradually reducing these random influences, called annealing by analogy to the temperature controlled process of crystal growing, can then result in the probability that the global minimum of the objective function is actually achieved to be asymptotically, in the limit

of an annealing schedule infinitely slow, to be equal to one.

An alternative strategy, which will be applied in this thesis, in particular for the approach described in section 4.2, chapter 4, is the method of *Homotopy*. This strategy is based on a set of transformations of the original error function into simpler or smoother functions with a smaller number of local minima. Minimization of the original function is then performed by starting with the most simple function present and gradually reducing the degree of smoothing during minimization until the original error function is received. Often, a suitable transformation arises automatically from the need to impose a complexity control. Homotopy in this case amounts to gradually releasing the constraints that the complexity control poses. This is in particular the case for the UKR model as shown later.

2.4. Projection Models

By using *deterministic* latent variables one obtains the class of models that is of particular concern with respect to *dimensionality reduction* and therefore of special interest for the methods described in this thesis. The latent variables are in this case treated formally as parameters that need to be estimated along with the function parameters. Since the backward mapping is modeled via some kind of projection, these models are generally referred to as *Projection Models*. In detail, this means that the score that corresponds to an observable data space element is given by its *projection index* which is formally defined as that latent space element that yields a minimal reconstruction error under f. The dependency on a particular model for f is often symbolized using the expression $s_{\mathbf{f}}$ for the backward mapping.

In the following the function class will be restricted to contain functions of the form:

$$f(x) = Wb(x), \tag{2.6}$$

with parameter matrix W and b being a vector of basis functions to be specified beforehand. The two aforementioned optimization steps (Projection- and Regression-Step) are

then given by:

$$\hat{x_i} = \arg\min_x \|y_i - Wb(x)\|^2, i = 1, \ldots, N, \tag{2.7}$$

and

$$\hat{W} = \arg\min_W \sum_i \|y_i - Wb(x_i)\|^2. \tag{2.8}$$

Note that in terms of the mappings f and g, involved in NLDR, optimization using this procedure only directly concerns f, while an optimal g is rather 'plugged in' instead of being adapted. In [Mal98] the importance of this proceeding is pointed out in a comparison of Principal Curves [HS89] which own this property, too, and so called Autoassociative Neural Networks (see [Kra91]) which do not. In particular, the presence of so called *ambiguity points* that cause the projection index defined as above to be a *discontinuous* function generally let the latter variant, where the backward mapping is a continuous function, fail to correctly approximate the given dataset. The importance of this finding for the UKR method resides in the fact that this method can be posed in two distinct ways. The first straightforwardly gives rise to a (nonparametric) projection model similar to those described in this chapter, in particular with the backward mapping defined as proposed above, while the second variant does not and can empirically shown to be flawed accordingly. The conclusions to be drawn from this finding will be detailed in 4.

As stated above, several methods for NLDR known from the literature may be formalized within the framework described in this section by varying the latent variable types and the class of candidate functions. In the following, some of the possible decisions on the latent variables and candidate functions and the algorithms they give rise to shall be sketched.

2.4.1. Principal Axes

By defining the latent variable domain to be $\Re^q$, i.e. using (deterministic) continues latent variables and restricting the function class to *linear* functions:

$$f(x) = c + Ax, c \in \Re^d, A \in \Re^{d \times q}, \tag{2.9}$$

the resulting learning method is essentially equal to *Principal Component Analysis*. Although the Projection-Regression-Scheme could be applied, the special, linear structure in this case gives rise to a closed form solution. Specifically, this is obtained through an eigenvalue decomposition of the input sample covariance matrix and therefore will be described in more detail in chapter 3, where also other, in particular recently developed *nonlinear* methods with this special property shall be delineated.

2.4.2. Principal Curves

Restriction of the latent variable domain to a closed interval $[a, b]$, on the real line, and f defined as

$$f(x) = Wb(x), W \in \Re^{d \times K}, \tag{2.10}$$

gives rise to nonlinear (one-dimensional) principal manifolds, or 'principal curves.' In fact, this way a generalization of the Principal Curves model proposed by [KKLZ00] is achieved, that are modeled by *polygonal* line segments there. The restriction of the latent variable domain is necessary here, as in this nonlinear case the absence of such a restriction would result in an interpolation of the training data points.

2.4.3. Principal Points

By using *discrete* latent variables and defining

$$f(x) = Wb(x), x \in \{1, 2, \ldots, K\}, W \in \Re^{d \times K} \tag{2.11}$$

$$b_j(i) = \delta_{ij}, \tag{2.12}$$

one straightforwardly obtains a learning method generally known as *Vector Quantization*. The columns of W then represent *prototype* or *codebook* vectors, the estimation of which using the general optimization scheme equals the well known K-means clustering algorithm.

2.4.4. Local Principal Axes

The use of a mixture of discrete and continuous latent variables together with a linear dependency on the continuous variables can be interpreted as a generalization of the vector quantization approach described above. The principal points are in this case replaced by continuous linear manifolds. This way one obtains a method for nonlinear dimensionality reduction that is known as 'Local PCA' in the literature and that can be used to model nonlinear relationships by residing to the assumption of local linearity.

The absence of a global coordinate system inherent to this approach, however, involves serious shortcomings as described, for example, in [TdSL00]. In fact, the LLE method, which will be introduced in 3.2.1 originally arose from a series of attempts to provide a *coordination* of locally linear models in order to overcome these shortcomings (see also [RS00] and [VVK02], e.g.).

2.5. Generative Models

The kinds of models in UL that include some kind of *density estimation* are referred to as *generative models* [Mei00]. These models arise automatically from the generalized regression framework by regarding the latent variables as random variables with non-trivial distributions. Since the UKR model can be formulated in a way to obtain also a generative variant, these kinds of models shall be sketched here in short.

If one uses random latent variables, optimization by minimization of 2.4 is no longer possible, as the interpretation of the latent variables as parameters no longer applies. Instead, the maximum likelihood approach (or some kind of Bayesian learning, which

is omitted in this thesis, as stated) needs to be used. Furthermore, the Projection-Regression optimization scheme is no longer applicable. It is replaced by an analogous scheme, known as EM-algorithm, as mentioned above. The detailed description of this algorithm shall be omitted here, as there exists a large body of literature on this topic (see, e.g. [Bil97] or [Cou96]). In short, the resulting optimization scheme closely resembles the PR-scheme with the update of latent variable realizations being replaced by an update of their *probability distribution* in this case. In the following a few models and some novelties that arise from the use of random latent variables will be sketched.

Assuming spherical Gaussian noise and a linear dependency on Gaussian latent variables one obtains a generative counterpart of the linear model given in 2.4.1. For a predefined latent space dimensionality q, the generative version yields an equal solution. However, as an extension to the projection model, by making use of the special role the noise variance plays in the generative case, a *dimensionality estimating* version can be obtained, by *predefined* the noise variance.

For *non-Gaussian* latent variables an estimation scheme for the well-known *Independent Component Analysis* ([Hyv99]) can easily be derived making use of the non-trivial latent variable distribution here by incorporating the assumption of their being statistically independent.

As a generative version of the principal points model (2.4.3) a method for density estimation generally known as 'mixture of Gaussians' arises straightforwardly from a (spherical) Gaussian noise assumption. In addition probabilistic versions of the clustering algorithm and of the local PCA model become possible, for example.

3. Spectral Methods for Dimensionality Reduction

One broad class of methods for dimensionality reduction that differs from most of the approaches delineated in the previous chapter is the class of *spectral methods*. The main difference is that these methods do not deploy any iterative optimization scheme. Instead, they rely on an objective function that has an efficiently computable global optimum. The by far most important and widely used instance is Principal Component Analysis. Recent developments regarding the applicability of spectral methods to nonlinear learning problems, however, has led to a current rise in their popularity, too.

The point of contact for practically all these methods is that they rely on an optimality criterion that can be posed as a *quadratic form*. Therefore, these methods rely on some variant of the *Rayleigh Ritz* theorem, which states in short, that for some quadratic form Q minimizer (maximizer) of $\mathrm{tr}(V^T Q V)$ with respect to the $q \times N$ matrix V, subject to $V^T V = I$, and $V\mathbf{1} = \mathbf{0}$, is the matrix V^{opt} containing the q eigenvectors corresponding to the q smallest (largest) eigenvalues of (symmetric) Q (see, e.q. [HJ94] or [Jol86]). In other words, all these methods rely on an *eigenvalue decomposition* (EVD) of some matrix Q, hence the name 'spectral methods.' A further commonness is that Q is defined as some kind of *data affinity matrix* of a given dataset in all cases, as recently pointed out by [BH02].

In the following the different spectral approaches to dimensionality reduction will be described and their specific characteristics, in particular in view of the spectral UKR variant to be introduced in 4.3, shall be pointed out, beginning with the linear variants

and PCA. A common drawback of the *nonlinear* generalizations of PCAis that, although these inherit the efficiency of their linear counterpart, they do not share the same merits in terms of *generalization*, as will be described in detail later. This entails particular problems with regard to potential applications as pointed out in chapter 1.

3.1. Linear Models

3.1.1. Principal Component Analysis

Principal Component Analysis (PCA) can be regarded as the oldest and most well-known method for dimensionality reduction[1]. It can be derived from the objective to maximize the variance of the projection of a given d-dimensional dataset onto a q-dimensional subspace. The quadratic form that preoccupies this objective is simply the (symmetric and positive definite) *sample covariance* matrix ([Jol86]). Precisely, assuming Y to be mean centered, let the spectral decomposition of YY^T be: $YY^T = U\Lambda U^T$ and let $U_{(q)}$ denote the matrix containing the normalized eigenvectors corresponding to the q largest eigenvalues as columns. The matrix X of the latent space vectors that meet the maximal variance objective is then given by $X = U_{(q)}^T Y$. Similarly, generalization to some new observable space element y is performed simply by left-multiplication with $U_{(q)}^T$, while the application of the forward mapping to some new latent space element x is performed by left-multiplication with $U_{(q)}$. In other words, here one has the unique case of a method that gives rise to the direct estimation of the involved *functions*, as latent space realizations are in fact obtained subsequently by *applying* the obtained model for f to the given input dataset.

3.1.2. Multidimensional Scaling

The method of *Multidimensional Scaling* (MDS) deviates somewhat from the other methods for dimensionality reduction exposed in this thesis, because it is not used as a method to determine a low dimensional representation of a high dimensional dataset,

[1] Although the derivation of PCA draws from much broader (but related) objectives, here it will be treated as a method for *dimensionality reduction* only.

but it obtains some dissimilarity measure as input instead. In fact, it thereby fails to meet the general definition given in chapter 1. It will be described here in short, nevertheless, as it is generally classified as a method for dimensionality reduction in the literature and makes up a crucial step of the Isomap algorithm portrayed below.

In detail, MDS addresses the problem of finding a (usually 'low' dimensional) data set from a set of pairwise dissimilarities, such that the distances between the resulting data points approximate the dissimilarities as closely as possible. Given a dissimilarity matrix[2] D with an EVD $D = V\Lambda V^T$ the optimal data set is given by $V\Lambda^{\frac{1}{2}}$. As stated, the abandonment of the need to use a data set from some euclidean space as input gives rise to applications that go beyond generic dimensionality reduction. Often, some kind of subjective similarity judgment is used as a basis to obtain D, which allows for the visualization of 'psychological spaces' [Krz96], for example. If D contains the pairwise euclidean distances of some real dataset, however, the solution is the same as that from PCA.

3.2. Nonlinear Models

3.2.1. Locally Linear Embedding

A method that incorporates an EVD in order to accomplish *nonlinear* dimensionality reduction and that has arisen a great deal of attention recently is the method of *Locally Linear Embedding* (LLE) [RS00]. It is essentially based on geometrical intuitions as it attempts to determine a lower dimensional embedding of a given dataset that *retains local neighborhood relations* between datapoints by making use of the following three-step algorithm:

1. for each y_i define I_i the index set of k nearest neighbors

[2] Generally, the matrix needs some simple preprocessing, which will not be detailed here.

2. set $w_{ij} := 0, \text{if} j \notin I_i$ and minimize with respect to w_{ij} the objective:

$$\sum_{i=1}^{N} \|y_i - \sum_{j \in I_i} w_{ij} y_j\|^2, \text{s.t.} \sum_{j \in I_i} w_{ij} = 1 \tag{3.1}$$

3. minimize with respect to x_i the objective:

$$\sum_{i=1}^{N} \|x_i - \sum_{j=1}^{N} w_{ij} x_j\|^2 \tag{3.2}$$

$$= \|XM\|_F^2, \tag{3.3}$$

with

$$M := \Big((w_{ij})_{ij} - I\Big)^T. \tag{3.4}$$

The purpose of the second step is to discover those weights that give rise to an optimal reconstruction of each observable space datapoint by its neighbors. This step requires to solve a constrained least squares fit and has a closed form solution. In the case of $k > d$ some kind of regularization heuristic is necessary, however. In the third step those latent variable realizations are then sought that minimize the average (latent space) error, if reconstructed from their neighbors with the same weights as their observable space counterparts. By writing 3.3 as $\text{tr}(XMM^TX^T)$ it is obvious that this step gives rise to a quadratic form in the latent variables. Therefore, by requiring $X\mathbf{1} = \mathbf{0}$ and $XX^T = I_q$, the solution is given by the eigenvectors belonging to the q (second to) smallest eigenvalues of $Q := MM^T$. Particularly advantageous here is the that Q is sparse, allowing for an efficient solution.

Overall, in tending to preserve the weights with which a datapoint is reconstructed from its neighbors under the sum-to-one constraint, the authors of the LLE algorithm state that it tends to preserve exactly those properties of each neighborhood that are invariant under *rescalings*, *rotations*, and *translations*. Hence, the algorithm provides

a mapping from observable to latent data space that tends to be *linear* for each neighborhood. In other words, LLE determines a low dimensional global representation of a manifold embedded in a higher dimensional space by assuming it to be arranged in linear patches. This assumption also defines the tender spot of the LLE algorithm, since a violation of it lets the algorithm fail. This is in particular the case, if *noise* is present in the data.

Generalization

The authors of the LLE algorithm propose two ways of generalizing a trained model to new latent or observable space elements. The first, non-parametric, approach is in principle a straightforward re-application of the main procedures that lie beneath the learning algorithm itself: for a new latent or observable vector generalization of f or g, is achieved by: (i) identifying the new datapoints' neighbors among the training or latent variable set, respectively, (ii) determining corresponding reconstruction weights, and (iii) evaluating the function as a linear combination of the found neighbors with their corresponding weights.

The theoretical justification for this kind of generalization lies in the same geometrical intuitions that the lle-algorithm itself is based on. In particular the assumption of local linearity is crucial for this approach to generalization to work properly, so that the dependency on noise-free data applies here in the same manner as above. The fact that the concept of generalization is most essentially based on the presence of noise, however, thereby represents a serious problem for this approach.

The second, parametric, approach to generalization that is proposed by the authors is to train a *supervised* model on the input-output data pairs that are available after the application of LLE. The problems affecting non-parametric generalization can be circumvented this way. The resulting overall algorithm consists of two separate parts - an unsupervised and a supervised part - with two unrelated objectives: optimal reconstruction of the latent data elements from their neighbors for the first and minimization of the expected prediction error for the second (see 2.1). A final model assessment is therefore based on the prediction error in the space of the observable variables. The problem is that it is not at all clear to what extend complying with the first objective is able to

preoccupy the second. This is in clear contrast to the projection models described in 2.4 that estimate latent variable realizations and a model for the forward mapping at the same time and both with the objective to minimize the observable space error.

3.2.2. Isomap

A method for NDLR that arose similar attention as LLE is the *Isomap* (isometric feature mapping) algorithm proposed by [TdSL00]. It can be regarded as a heuristic approach, too, but it is based on completely different intuitions than LLE. Isomap similarly emanates from the observable data being distributed along a low-dimensional manifold, but abstains from the assumption of linear patches. It seeks to find a low-dimensional embedding that preserves distances between datapoints *as measured along the manifold* - so called 'geodesic' (locally shortest) distances. The crucial point in this algorithm is therefore to derive these geodesic distances from the datapoints (more precisely, their *euclidean* distances) in an efficient way. To achieve this the authors propose the three step algorithm of (i) computing a topology-preserving network representation of the data, (ii) computing the shortest-path distance between any two points which can efficiently be done by using *dynamic programming*, (iii) determining the low-dimensional representation that preserves the computed distances as closely as possible using *Multidimensional Scaling* (MDS) on these distances.

With regard to generalization, Isomap shares the same shortcomings of LLE, because of its likewise rather heuristic quality. In fact, as in contrast to LLE not even a heuristic generalization of the involved mappings can be naturally derived from the intuitions the algorithm is based upon, the authors suggest to train a *supervised* model on the obtained completed dataset – resulting in the same shortcomings that hold above.

3.2.3. Kernel PCA

Another spectral approach to nonlinear dimensionality reduction, which at first glance resembles the UKR variant to be described in 4.3 with regard to its incorporation of both kernel and spectral methods, is *Kernel PCA*. A closer look reveals important differences, however.

Kernel PCA stems from a line of research on kernel based methods that is based on the idea to incorporate an implicit (usually highly nonlinear) mapping to some higher dimensional feature space by exploiting the finding that dot products in such a feature space can equivalently be computed using the original data space vectors alone through the help of kernel functions [Bur98], which is often referred to as 'kernel trick.' In order to apply this idea to Unsupervised Learning, the classical PCA approach may be recast in a form that makes use of inner products only. In [SSM99] the derivation of this formulation is given resulting in an algorithm to compute principal components in a higher dimensional feature space. Technically this amounts to performing the EVD of the matrix of pairwisely evaluated kernel function.

4. Unsupervised Kernel Regression

This chapter describes the method of Unsupervised Kernel Regression as recently introduced by [Mei03]. In a first step the concept of Nonparametric Regression is introduced. Then two distinct ways of extending this concept to Unsupervised Learning will be delineated in the subsequent two sections. In addition, a combination of the objective functions these two variants give rise to which has in particular proven to be useful with regard to practical considerations, will be proposed in the section that follows.

4.1. Unsupervised Nonparametric Regression

4.1.1. The Nadaraya Watson Estimator

Chapter 2 introduced the purpose of regression as that of modeling a functional relationship between variables by choosing that element from a parameterized set of candidate functions that minimizes the empirical prediction error on a training data set, which is asymptotically equivalent to taking the conditional expectation:

$$f(x) = E[y|x] = \int yp(y|x)dy \text{ (repeated)}. \tag{4.1}$$

In contrast to 2.1, where the approximation of the regression function has been realized by minimization of the empirical prediction error or maximization of the data log-likelihood, one obtains a *nonparametric* variant of the regression estimation, if one aims directly at modeling the conditional expectation. This can be achieved by considering

a non-parametric estimator $\hat{p}$ of the joint probability density function $p(x, y)$ of the involved variables, taking into account that 4.1 can be relocated to yield:

$$f(x) = \frac{\int yp(x, y)dy}{\int p(x, y)dy}. \tag{4.2}$$

By utilizing the multivariate *Kernel Density Estimator* (KDE) to model the joint density [Sco92]:

$$\hat{p}(x, y) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{K}_q(x, x_i)\mathcal{K}_d(y, y_i), \tag{4.3}$$

with $\mathcal{K}(.,.)$ being multivariate Kernel functions and $K(.,.)$ denoting the unnormalized portions thereof, so that with the normalization constants $\mathcal{N}_d$ and $\mathcal{N}_q$

$$\int \mathcal{K}_q(., z)dz = \int \mathcal{N}_q K(., z)dz = 1 \tag{4.4}$$

and

$$\int \mathcal{K}_d(., z)dz = \int \mathcal{N}_d K(., z)dz = 1 \tag{4.5}$$

hold, the estimate of the regression function becomes[1]

$$f(x) = \frac{\sum_{j=1}^{N} K(x, x_j)y_j}{\sum_{k=1}^{N} K(x, x_k)}, \tag{4.6}$$

[1]The symbol K will be overloaded to denote both the observable space and latent space Kernel functions, and later on also the matrix of kernel functions. Since the dataspace kernel functions cancel out in 4.6, here they are only specified with latent space arguments; later on, however, they will be used to accommodate data space arguments analogously.

which is known as the *Nadaraya Watson Estimator* [Bis96]. In this thesis only the spherical Gaussian

$$K(x_i, x_j) = \exp(\frac{1}{2h^2}\|x_i - x_j\|^2) \tag{4.7}$$

and the Epanechnikov kernel function

$$K(x_i, x_j) = \begin{cases} 1 - \frac{1}{h^2}\|x_i - x_j\|^2, & \text{if } \|x_i - x_j\|^2 < 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

will be deployed and both denoted by $K(.,.)$. The function used will be indicated at the respective places.

The function parameter h determines the *kernel bandwidth* and provides a means to adjust the model complexity for the Nadaraya Watson Estimator.

4.1.2. Unsupervised Nonparametric Regression

As in the parametric case, the transition to unsupervised regression is made by regarding the regressors as *latent*. In contrast to the parametric case, however, that poses the twofold objective of finding suitable latent variable realizations along with parameters defining a functional relationship, here both objectives are achieved at the same time by merely taking care of finding suitable latent variable realizations, because of the nonparametric nature of the problem. This way the 'double burden' that problems in UL hitherto gave rise to is eliminated, resulting in methods that resemble those from SL, because they depend on the estimation of only one class of parameters.

As pointed out in [Mei03], in the unsupervised case the Nadaraya Watson Estimator may be deployed in two ways. The first is to treat the latent variables in 4.6 as *parameters* to be estimated. By measuring the observable space error one obtains the objective function dealt with in detail in the next section. The second way is to compute the latent variable realizations by simply *applying* the Nadaraya Watson Estimator to the observed variables, that are regarded as input in this case. In other words, this variant amounts to computing the nonparametric regression function in the opposite direction.

The objective function for this approach is obtained by measuring the latent space error. In that case, a nontrivial coupling of the resulting latent variable realizations has to be accounted for, a problem that can be solved efficiently by a spectral decomposition as described in 3. This variant will be described in 4.3.

4.2. Observable Space Error Minimization

To obtain a loss function for learning suitable latent variable realizations one might conceive of 4.6 as being parameterized by the latent data matrix X and measure the mean square reconstruction error on the observed variables[Mei03]. The resulting objective function of this UKR variant, denoted oUKR in the following, is given by:

$$E(X) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \frac{\sum_{j=1}^{N} K(x_i, x_j) y_j}{\sum_{k=1}^{N} K(x_i, x_k)}\|^2 \tag{4.9}$$

$$= \frac{1}{N}\|Y M\|_F^2 \tag{4.10}$$

with

$$M := \Big(\frac{K(x_i, x_j)}{\sum_{k=1}^{N} K(x_i, x_k)}\Big)_{ij} - I.$$

Since the effect of any variation of the kernel bandwidth could be equivalently caused by a change in the average scale of the latent variable realizations, in the following, if not stated otherwise, the kernel bandwidth will be conceived of as being constant ($h = 1.0$) and any influence on the model complexity will be accounted for or effected by the latent variable norms only.

X – and by virtue of 4.6 at the same time f – will then be estimated by minimizing 4.10. Since a minimization without further restrictions on the objective function would drive the latent variable scales to infinity, however, with the reconstruction error for the training data set at the same time approaching zero, it is obvious that some kind of complexity control has to be imposed in order for the optimization problem to be well

defined [Mei03]. Practical considerations regarding both minimization and restriction of the model complexity will be delineated in the next section.

The generalization of a trained model to new latent space or observable space elements is straightforward. Since a model for f is learned along with suitable latent variable realizations, the oUKR method closely resembles the projection models described in 2.4. The only difference is that no iterative training procedure is necessary here, because no parameters need to be estimated. Application of the regression function to new latent space elements is therefore possible simply by plugging these into 4.6. And the backward mapping g can be defined analogously to 2.7:

$$g(y) := \arg\min_x \|y - f(x)\|_2^2. \tag{4.11}$$

This requires to solve a nonlinear optimization problem, that should be initialized suitably. A straightforward choice as an initialization is

$$x_0 := \arg\min_{x_i} \|y - f(x_i)\|_2^2, i = 1, \ldots, N, \tag{4.12}$$

requiring a search over the number of training data points.

From the reasoning laid out so far a generative model is obtained simply by regarding the latent variables as *random* variables as in 2.5 and maximizing the conditional log-likelihood (see chapter 2) which can be derived straightforwardly (see [Mei03] for details).

4.2.1. Optimization

The objective function is nonlinear with respect to the latent variables and needs to be minimized iteratively. Since it is possible for the gradient of the objective function to be computed analytically, some gradient based optimization scheme may be used. For the partial derivatives with respect to the latent variables it holds:

$$\frac{\partial E(X)}{\partial x_{ij}} \propto \sum_{k=1}^{N} \sum_{l=1}^{d} y_l^T M^k y_l^T \frac{\partial M^k}{\partial x_{ij}}, \tag{4.13}$$

with M^k denoting the k^{th} column of M.

Since there are N latent vectors with q components each, it is obvious that the time complexity for computation of the gradient amounts to at least $O(N^2dq)$. This is indeed the complexity class for computation of the gradient, if one pre-computes $y_l^T M^k$ and $y_l^T \frac{\partial M^k}{\partial x_{ij}}$. Computation of these terms gives rise to costs of $O(N^2 d)$ and $O(N^2 dq)$, respectively. While this is clear for the first expression, the second one deserves special attention. It holds:

$$\begin{aligned}
A_{ij}^{kl} &:= y_l^T \frac{\partial M^k}{\partial x_{ij}} \\
&= \sum_{m=1}^{N} y_{lm} \frac{\partial}{\partial x_{ij}} \Big(\frac{K(x_k, x_m)}{\sum_{t=1}^{N} K(x_k, x_t)} - \delta_{km} \Big) \\
&= \sum_{m=1}^{N} y_{lm} \frac{1}{\big(\sum_{t=1}^{N} K(x_k, x_t)\big)^2} \\
&\quad \cdot \Big(\frac{\partial K(x_k, x_m)}{\partial x_{ij}} \sum_{t=1}^{N} K(x_k, x_t) - K(x_k, x_m) \big(\delta_{ik} \sum_{t=1}^{N} \frac{\partial K(x_i, x_t)}{\partial x_{ij}} + \frac{\partial K(x_k, x_i)}{\partial x_{ij}} \big) \Big) \\
&= \frac{1}{\sum_{t=1}^{N} K(x_k, x_t)} \sum_{m=1}^{N} y_{lm} \frac{\partial K(x_k, x_m)}{\partial x_{ij}} \\
&\quad - \frac{1}{\big(\sum_{t=1}^{N} K(x_k, x_t)\big)^2} \big(\delta_{ik} \sum_{t=1}^{N} \frac{\partial K(x_i, x_t)}{\partial x_{ij}} + \frac{\partial K(x_k, x_i)}{\partial x_{ij}} \big) \sum_{m=1}^{N} y_{lm} K(x_k, x_m).
\end{aligned} \tag{4.14}$$

Therefore

$$A_{ij}^{kl} = \frac{1}{O^k} P_{ij}^{kl} - \frac{1}{(O^k)^2} \big(\delta_{ik} Q_{ij} + \frac{\partial K(x_k, x_i)}{\partial x_{ij}} \big) R^{kl},$$

with

$$O^k := \sum_{t=1}^{N} K(x_k, x_t),$$

$$P_{ij}^{kl} := \sum_{m=1}^{N} y_{lm} \frac{\partial K(x_k, x_m)}{\partial x_{ij}} = y_{li} \frac{\partial K(x_k, x_i)}{\partial x_{ij}} + \delta_{ik} \sum_{m=1}^{N} y_{lm} \frac{\partial K(x_i, x_m)}{\partial x_{ij}},$$

$$Q_{ij} := \sum_{t=1}^{N} \frac{\partial K(x_i, x_t)}{\partial x_{ij}},$$

$$R^{kl} := \sum_{m=1}^{N} y_{lm} K(x_k, x_m).$$

The terms O, P, Q, R may be pre-computed, which amounts to a time complexity of $O(N^2)$, $O(N^2dq)$, $O(N^2q)$ and $O(N^2d)$, respectively.

Throughout this chapter only the Gaussian kernel function will be used. In this case the derivative is given by

$$\frac{\partial K(x_i, x_k)}{\partial x_{ij}} = \frac{\partial K(x_k, x_i)}{\partial x_{ij}} = \frac{1}{h^2} K(x_i, x_k)(x_{ij} - x_{kj}).$$

To further reduce time complexity for computation of the objective function as well as its gradient, a sparse structure of the kernel matrix might be induced by residing to a kernel function with finite support, such as the Epanechnikov kernel or the differentiable 'quartic kernel' [Sco92]. An exploration of this approach, however, is not within the scope of this thesis.

Ridge Regression

The above mentioned requirement to restrict the model complexity can be met by adding a regularization term $P(X)$ to the objective function that constrains the scale of latent variable realizations in some way. This is is referred to as 'ridge regression' or 'weight decay' in the literature (see [HS89]). The resulting penalized objective function then

reads

$$E^P(X) = E(X) + \lambda P(X). \tag{4.15}$$

The parameter $\lambda \in \Re$ functions as a means to control the influence of the regularization term and thereby provides a way to control the model complexity. The matrix of suitable latent variable realizations is now determined by minimizing the penalized objective function[2]:

$$X^{opt} = \arg\min_X E^P(X). \tag{4.16}$$

A straightforward choice for the regularization term is one that equally restricts the euclidean latent variable norms, which can be achieved by defining

$$P(X) := \|X\|_F^2. \tag{4.17}$$

Other regularization terms are conceivable, however. In fact, these might even be desirable in specific circumstances, since they could provide a facility to include some kind of top-down knowledge by imposing constraints upon the latent variables and their relations. This finding will be evaluated in more detail in section 4.2.2. In the following, if not stated otherwise, the variant given in 4.17 will be used only.

To deal with the presence of local minima, an optimization strategy such as the homotopy scheme described in 2.3 needs to be applied for minimization of 4.15. Here, this practically amounts to slowly decreasing λ during optimization. In order to avoid overfitting, the error that the projection of an independent test set onto the resulting manifold gives rise to might be tracked. As an illustration, the approximations of the two-dimensional 'noisy S-manifold,' embedded in a three-dimensional space, which are depicted in 4.1, have been determined this way (see Appendix A for details on the generation of the toy datasets used here in the following). Here, in particular the importance of a well chosen annealing strategy becomes obvious. While the two panels to the right

[2] Note that for optimization by gradient descent the gradient given in 4.13 needs to be modified by adding the derivative of the respective penalty term with respect to the latent variables.

visualize the progression of the latent variable realizations obtained from a reasonably chosen annealing strategy, the two panels to the left show the effect of too fast annealing and the result of getting stuck in a local minimum leading to an only suboptimal final solution. The out-most illustrations have been rescaled to accommodate visualization of the latent variable realizations, while the innermost panels show the same results in a consistent scale for each of the two progressions, making visible the enlargement of the latent realizations throughout the annealing progress. It is obvious that, while the solution to the left after 20 annealing steps spans an area that is approximately five times larger than the solution to the right after 350 steps, the final solution using 'slow' annealing clearly captures the structure of the original data set far better than the left one.

A general drawback of using ridge regression and homotopy that is related to the necessity of a suitably 'slow' annealing schedule is *efficiency*. Datasets containing a thousand or more elements in a few hundred dimensions have shown to be problematic to handle and therefore might ask for alternative strategies. One option will be exposed in the following.

Constrained Optimization

While adding a penalty term as described above can be interpreted as imposing a soft constraint on minimization of the objective function by including the tendency to favor small norm solutions, it is also possible to use a strictly constrained optimization algorithm instead by re-defining the optimization problem as

$$X^{opt} = \arg\min_{X} E(X), \tag{4.18}$$

$$\text{subject to } c(X) <= 0, \tag{4.19}$$

with c defining some nonlinear constraint. In analogy to 4.17 one might set

$$c(X) = \left(\|X\|_2^2 - a \right), \tag{4.20}$$

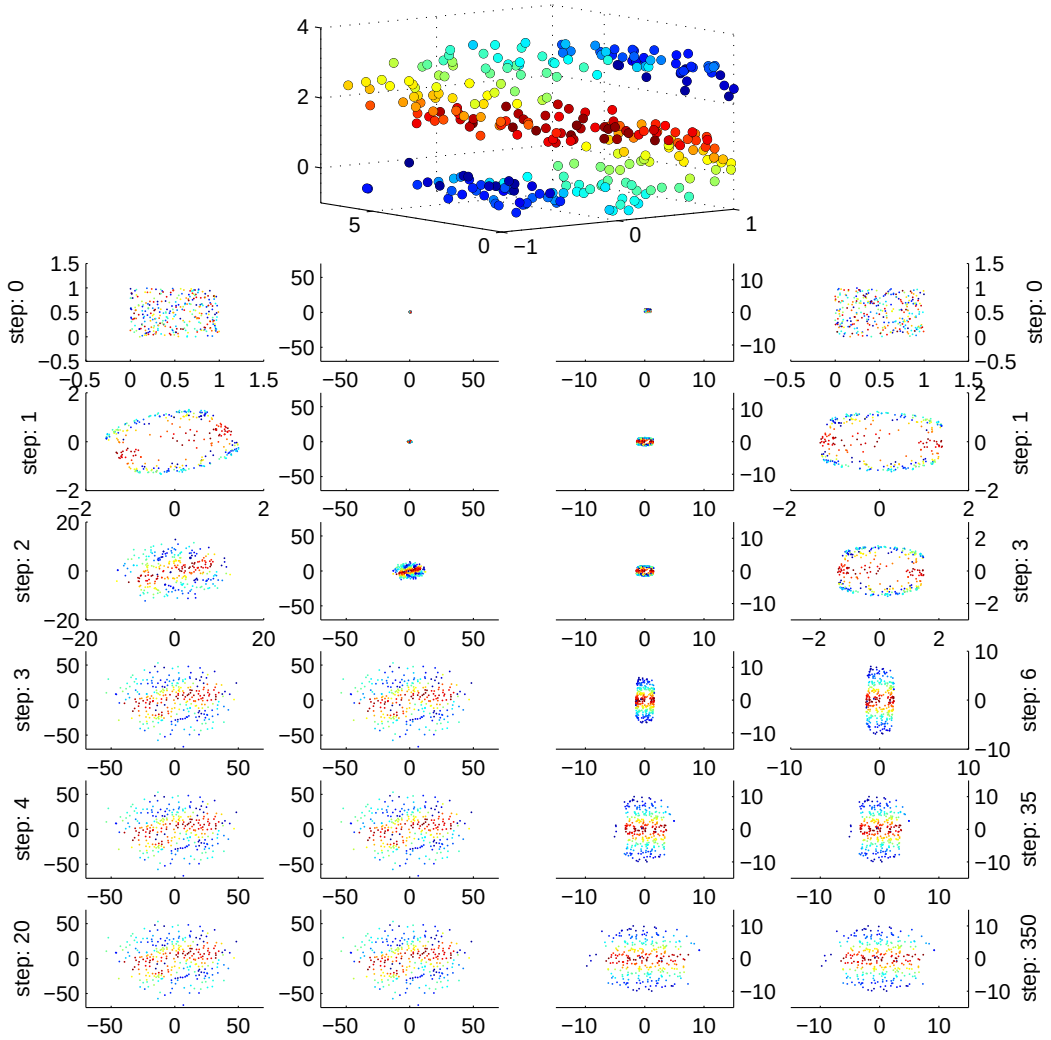

Figure 4.1.: The effects of different annealing strategies. The dataset on the top, consisting of $N = 300$ datapoints sampled from the two-dimensional 'S-manifold' distribution with spherical Gaussian noise ($\sigma^2 = 0.5$) has been approximated using: (left) 20 annealing steps, with λ being declined geometrically with factor 0.1 after each step and (right) 350 annealing steps, with λ being declined with factor 0.9. Start value for λ was 1.0 in all cases. The latent variables have been initialized randomly from a uniform distribution over the unit square. Note the tendency of the latent space realizations to arrange spherically provoked by using the Frobenius norm as regularization.

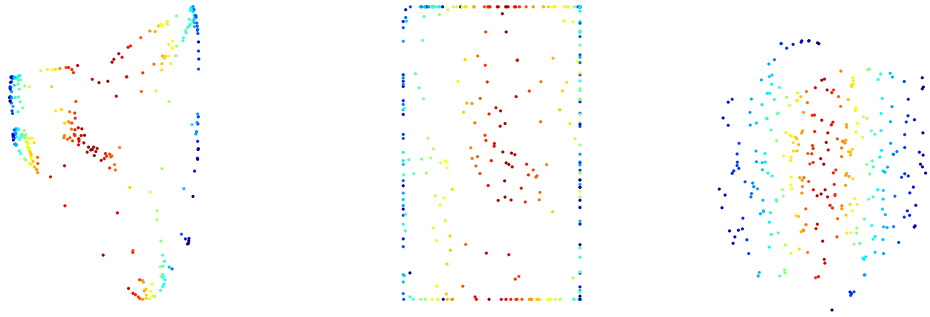

Figure 4.2.: The effects of different optimization constraints using the two-dimensional 'S-manifold'. The left plot shows the solution obtained from lUKR with kernel bandwidth h set to 1.0 (see section 4.3). The solution was used as an initialization for oUKR using bound constraints as defined in 4.21 in one case (middle plot) and the nonlinear constraint as defined in 4.20 in another (right plot).

for example, in order to restrict the average latent variable norms. Alternatively, simple bound constraints be applied. For this means, 4.19 may be simplified to yield

$$L <= X <= U, \tag{4.21}$$

with L and U being matrices of lower and upper bounds, respectively.

Making use of the homotopy strategy here then amounts to gradually releasing the constraints by directly decreasing or increasing the entries of L or U, respectively, or by increasing a. However, an important difference to ridge regression is that, if a latent matrix initialization X^{init} is available, one may abstain from homotopy and derive suitable constraints from the initialization instead, for example by setting $L := -abs(X^{init})$ and $U := abs(X^{init})$ or by setting $a = \|X^{init}\|_2^2$, with c defined as above. The question that remains, of course, is where to obtain such an initialization owning the required property of being correctly scaled so that suitable constraints can be derived in some way. The solutions that the LLE algorithm yields, for example, are not useful here as they are arbitrarily scaled as described in 3. Nevertheless, there is a way to obtain such a suitably scaled initialization, as will be described in 4.4.

In anticipation of the results given there, an example of the constrained (re-)optimization of such an initialization is depicted in figure 4.2. A two-dimensional approximation of a dataset consisting of 500 elements of the noise-free ($\sigma = 0$) 'S-manifold' was determined using the lUKR variant that will be described in section 4.3. The obviously sub-optimal solution shown to the right was then used as an initialization X^{init} for the oUKR method. Constrained optimization was used in two different ways on this initialization. In the first case, bound constraints were used with L and U defined as above. In the second, a nonlinear constraint (4.20) was used with a defined as above. The figure shows that both strategies are able to improve the sub-optimal initialization. In addition, the influence that a constraint has on the final solution is visible. This influence is obviously stronger than it has been using ridge regression.

'Built in' Cross Validation

An alternative means of regularization that abandons the need to estimate any hyperparameters and the associated necessity to embed the minimization of 4.15 in an embracing cross validation loop can be obtained using a cross validation mechanism 'built into' the objective function. This is done by utilizing only a subset of the training data *at every function evaluation*, excluding in particular the data point to be approximated in the current evaluation step, so that an *external* validation set becomes over-due. Using 'leave-one-out' cross validation, where the only vector excluded is the one to be approximated, gives rise to the modified objective function:

$$E^{\text{cv}} = \sum_{i=1}^{N} \|y_i - f^{-i}(x_i)\|^2 \tag{4.22}$$

$$:= \sum_{i=1}^{N} \|y_i - \frac{\sum_{j\neq i} K(x_i, x_j) y_j}{\sum_{k\neq i} K(x_i, x_k)}\|^2 \tag{4.23}$$

$$= \|Y\tilde{M}\|_F^2 \tag{4.24}$$

with

$$\tilde{M} = \Big((1 - \delta_{ij}) \frac{K(x_i, x_j)}{\sum_{k \neq i} K(x_i, x_k)} \Big)_{ij} - I.$$

Then, an optimal embedding is given by

$$X^{opt} = \arg\min_X E^{\mathrm{cv}}(X). \tag{4.25}$$

While normally the 'leave-one-out' strategy is often problematic, because it is the computationally most expensive cross validation variant, the 'built in' alternative adopted here provides a convenient 'trick' to exploit and utilize the coupling of all datapoints and the related $O(N^2)$ complexity each function evaluation of the UKR method, being a kernel based method, gives rise to anyway. Thereby, leave-one-out cross validation becomes possible without any additional computational cost.

Applying homotopy is still possible using the built in cross validation variant, since one may still add a penalty term to 4.24 or constrain the solutions accordingly. In fact, one might even prefer the modified objective function over 4.10 for this means in order to forgo the tracking of some test error. But alternatively, if a suitable initialization is available, the built in cross validation mechanism also allows for some direct optimization. A well chosen initialization is vital in this case, however, because of the highly nonlinear structure of the objective function. An illustration is depicted in figure 4.3: A random initialization, as depicted at the top in (a), is not leading to a satisfactory result as can be seen at the bottom in (a). An only suboptimal solution that the LLE algorithm might yield, which can happen in particular in the presence of noise or because of a badly chosen neighborhood size (see 3.2.1), is seen at the top of (b) and provides the grounds for this method to find an appropriate embedding (see bottom of (b)).

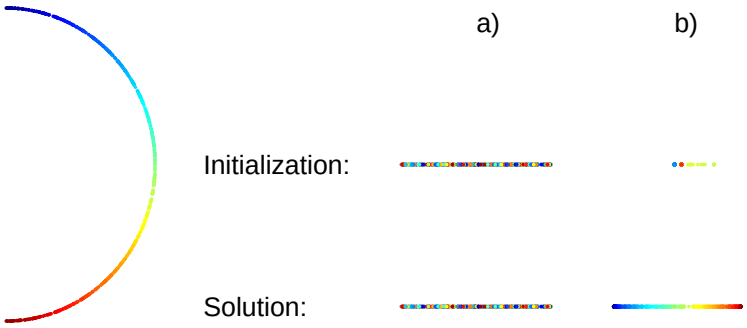

Figure 4.3.: Illustration of the importance of a suitable initialization for the built in leave-one-out cross validation variant. Noisefree two-dimensional 'halfcircle' dataset to the left and one-dimensional UKR approximation to the right. While UKR fails to find a suitable approximation for randomly initialized latent variables as can be seen in (a), an only suboptimal LLE-solution provides a starting point to obtain a satisfactory result, however, as shown in (b).

4.2.2. Experiments

Visualization of the regression manifold

As stated above, since the oUKR method determines a model for f along with suitable latent variable realizations, the application of the regression function to new latent space elements is straightforward. This makes it possible to visualize the the learned regression manifold (for up to three-dimensional embedding spaces) by sampling the latent space and 'plugging' the obtained latent space elements into the learned model for f. As shown in figure 4.2.2 this procedure has been used to visualize the effects that a *rescaling* of the latent data matrix X has on the resulting regression manifold. It is obvious that for a rescaling factor $s = 0.0$ the regression manifold degenerates to the *sample mean*. The result that the rescaling factor $s = 1.0$ gives rise to is shown in the rightmost plot in the top row.

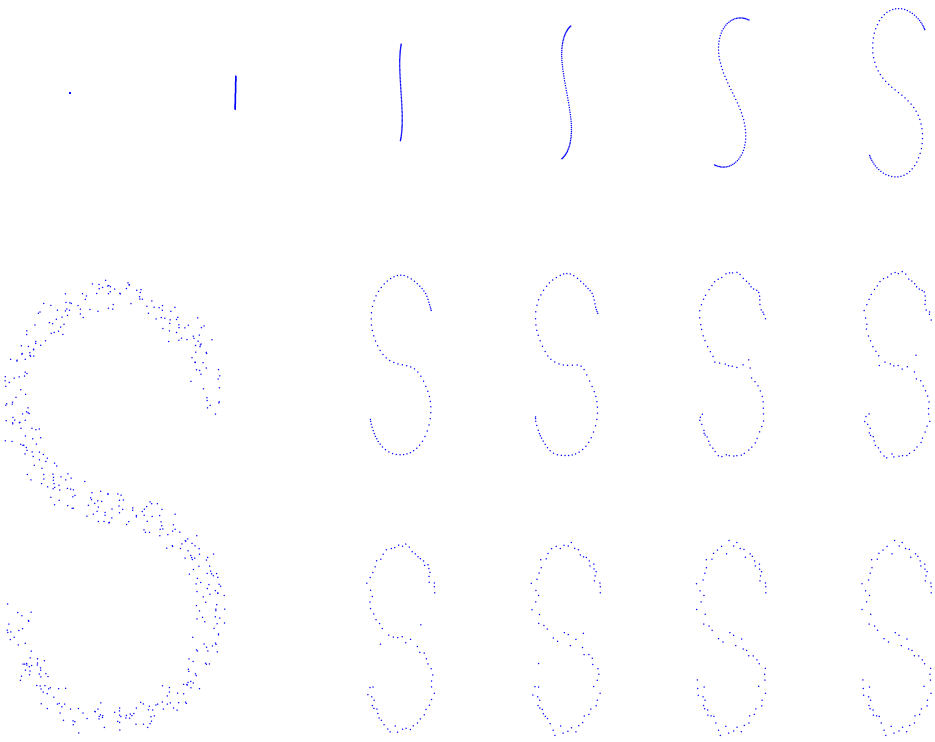

Figure 4.4.: Visualization of the different regression manifolds resulting from rescaling the latent data matrix. For the one-dimensional 'noisy S-manifold' dataset shown in the bottom left corner a low-dimensional representation was obtained using UKR with homotopy. The plots show in successive order from left to right and from top to bottom the visualization of the regression manifold obtained from sampling latent space along a regular grid after rescaling the latent data matrix with factors 0.0, 0.1, 0.2, 0.3, 0.5, 1.0, 2.0, 3.0, 8.0, 15.0, 30.0, 10^2, $2 \cdot 10^3$, and 10^5.

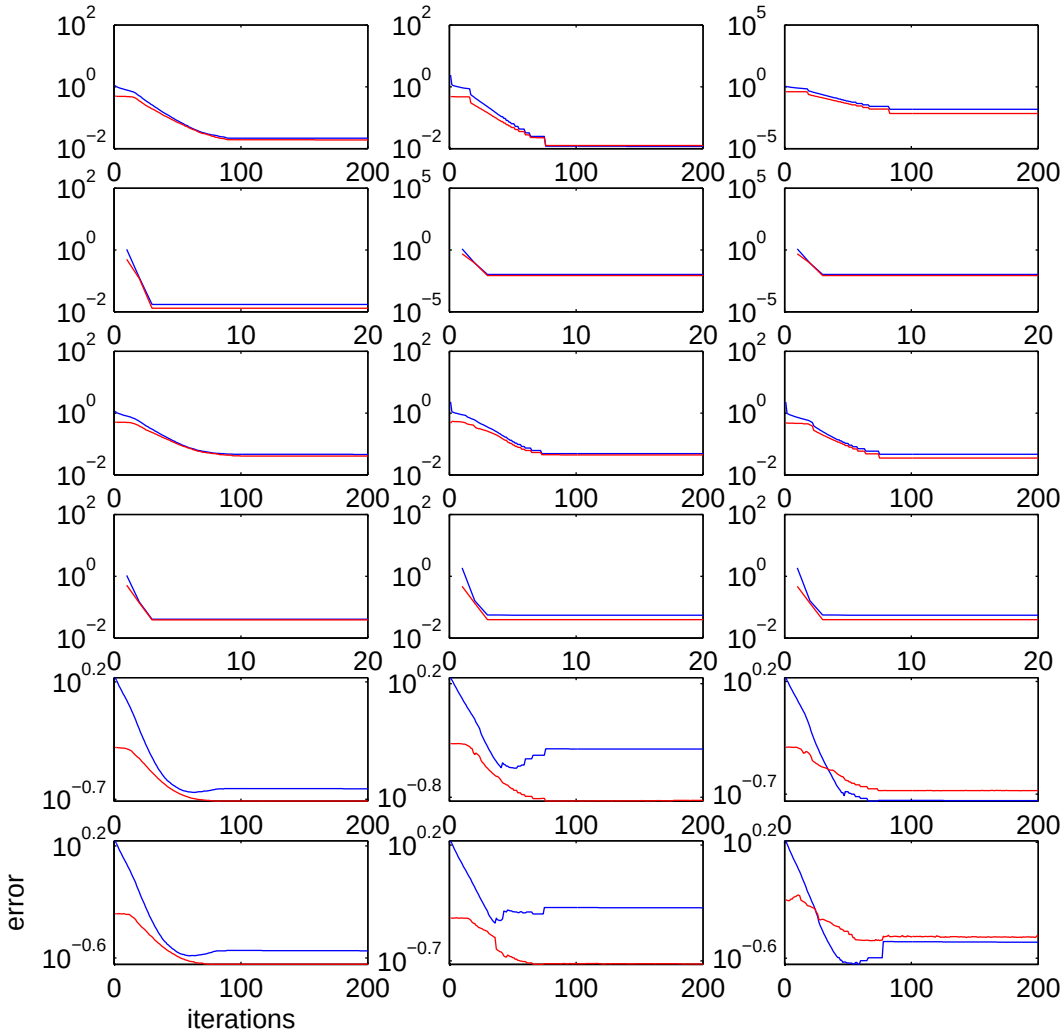

Figure 4.5.: The built in cross validation error E^{cv} (depicted in blue) compared to the error on an independent test set E^{test} (depicted in red). The left column shows the error progressions averaged over 50 runs, the middle and right column show out of the 50 runs only those with the largest deviations between the two error progressions according to the L_2 norm and to the L_{inf} norm, respectively. In the first and third row the progressions for the one-dimensional 'noisy S-manifold' with $\sigma = 0.0$ and $\sigma = 0.5$, respectively, are depicted. The second and forth row show the respective progressions resulting from a faster annealing schedule. The two bottom rows show the error progressions for the two-dimensional 'S-manifold', again for $\sigma = 0.0$ and $\sigma = 0.5$, respectively.

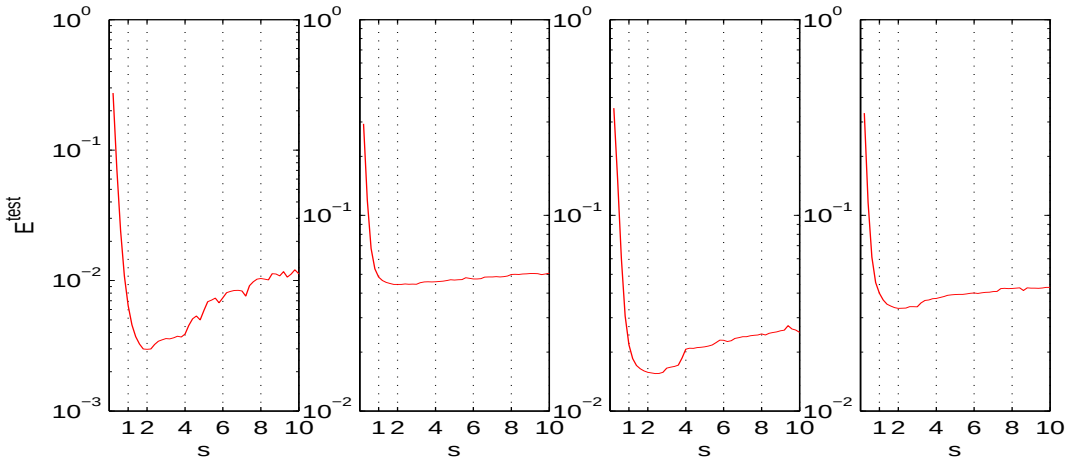

Figure 4.6.: The effect that a rescaling of the latent data matrix obtained from using the built in leave-one-out cross validation variant of oUKR has on the error that projection of an independent test set gives rise. The plots in the succession from left to right correspond to the settings described in the caption of figure 4.5 from top to bottom for the four top plots.

Built In Cross Validation vs. Homotopy

As stated, the built in cross validation error criterion is a promising computational shortcut for solving the problem of model selection. In order to provide empirical evidence that this criterion is of real practical value, an investigation of the distribution of the resulting error (E^{cv}) as compared to the average error on an independent test set (containing elements $y_i^{\text{test}}, i = 1, \ldots, N^{\text{test}}$),

$$E^{\text{test}} := \sum_{i=1}^{N^{\text{test}}} \| y_i^{\text{test}} - f(g(y_i^{\text{test}})) \|_F^2,$$

has been undertaken. To this end, the UKR model has been trained on the one-dimensional and on the two-dimensional 'S-manifold' datasets, comprising 50 elements each, in one setting without noise and in another with spherical Gaussian noise ($\sigma = 0.5$). The homotopy scheme has been applied using the penalized objective function (4.15) with regularization parameter λ starting with 1.0 in all cases and being annealed with factor 0.9 in one setting and for the one-dimensional case with factor 0.1 in another.

The error on a test set, consisting of 500 datapoints in each setting, has been tracked throughout and its average over 50 runs for each setting is depicted together with the cross validation error in 4.5 (first column). Overall, a strong correlation between these quantities can be noticed in all settings. They even converge as the number of training steps increases. In addition, the largest deviations along the 50 runs between the two error progressions according two the L_2 norm, as well as to the L_{inf} norm, are depicted in columns 2 and 3, respectively. They show significant differences only on the outset of the learning process, revealing the actual reliability of the cross validation criterion.

Another finding that becomes obvious from the plots is that a rise of the test error hardly ever occurs. Even for the settings in which the rather 'radical' annealing factor 0.1 has been applied, the test error monotonously decreases. This gives rise to the assumption that a change in λ rather leads to a slight deformation of the error surface than to a complete restructuring of it, so that the homotopy method causes a local optimizer to *track* a once reached local minimum during the overall optimization process.

The built in cross validation error, on the other hand, as can be seen in the plots for the two-dimensional datasets, does happen to rise. The plots in the last row show that this can even be the case when the test error is obviously still falling. From this observation one may draw the conclusion that an application of the built in cross validation criterion will rather give rise to underfitting than to overfitting. In order to collect more evidence for this conclusion, a further experiment has been conducted. The final solutions for the one-dimensional datasets have been used as an initialization for an iterative minimization of E^{cv}. Then the effect that a rescaling of the resulting latent data matrix with some rescaling factor s has on the test error has been measured. If training by minimization of E^{cv} leads to an over-regularization, rescaling with a factor $s > 1$ greater than one should result in a *decreasing* test error. Figure 4.6 shows that this is the case, indeed. While downsizing the latent variable scale leads to an increasing test error, enlarging leads to a decreasing error. The optimal rescaling factor with respect to the test error is approximately 2 in all settings. This indicates that built in cross validation indeed tends to a slight over-regularization. Since the reason to deploy this error criterion is to circumvent *overfitting* and since the increase of the test error that this over-regularization gives rise to is small as can be seen in the plots it can be concluded

that the built in leave-one-out criterion is indeed a practical alternative to other, external, model selection strategies.

Modeling Top-down influences: 'Implicit Clustering'

In 4.2.1 the effects that different optimization constraints and regularization terms have on the resulting latent space realization are mentioned, suggesting to use these in order to include some kind of top-down knowledge into this unsupervised method. In fact, the introduction of top-down influences into methods from UL is an important and emerging topic. In particular research in cognitive science, especially cognitive modeling, is concerned with questions regarding the integration of bottom-up and top-down processing. The importance of this is increasingly agreed upon and the occurrence of this in human brain functioning is common grounds (see [GSSSK00] or [SKK], for example). In the context of Unsupervised Learning, constraining of the learning process by introduction of some kind of *knowledge* provides a step towards this direction.

Although an evaluation of the potential of the UKR method in this context goes beyond the scope of this thesis, an example of the facility to include top-down knowledge through the use of appropriately tailored optimization constraints shall be given here: Figure 4.7 shows a dataset consisting of two 'half circles.' Fitting a model to such a disrupted manifold is not trivially possible for the known algorithms for dimensionality reduction. [RS00] e.g. state that it is an open question, how to deal with *non-uniformly* sampled data. However, including knowledge of the partitioning structure through the optimization constraints into the learning procedure, might simplify the task. This has been tried by using constrained optimization (see 4.18), with nonlinear optimization constraint

$$c(X) = -\left(\|X\|^2 - a\right)^2, \tag{4.26}$$

forcing the latent variable values to assemble in two groups. Parameter a has been initialized to 0.1 and has been increased by factor two after every update of the latent variable realizations. The second to right plot shows the effect of generalizing g to new observable space vectors from the same distribution and the rightmost plot the

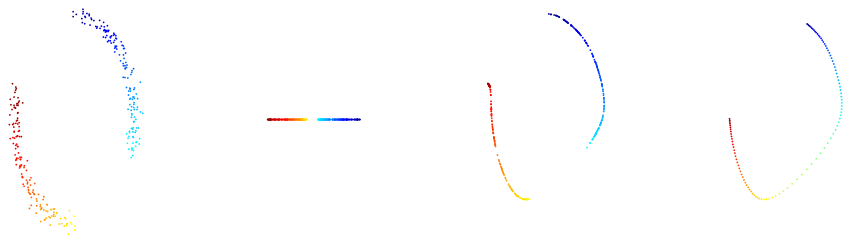

Figure 4.7.: 'Implicit Clustering' (From left to right): Partitioned dataset; latent space realization; projection of a test data set onto the resulting manifold; visualization of the regression manifold obtained from sampling latent space.

effect of generalizing f to new latent space elements, showing in particular the effect of sampling along the 'gap,' leading to an approximately linear connection of two adjacent ends of the two 'half-circles.' In forcing the latent vectors to assemble in two groups the procedure may be interpreted as an 'implicit' *latent space clustering*.

4.3. Latent Space Error Minimization

By turning the roles of the latent and the observable variables and regarding the first as outputs and the second as regressors the learning problem can also be defined as finding a suitable latent variable realization for every given data vector by deploying the Nadaraya Watson Estimator in this case in order to *directly compute the regression of* x *onto* y [Mei03]:

$$x = g(y) := \frac{\sum_{j=1}^{N} K(y, y_j) x_j}{\sum_{k=1}^{N} K(y, y_k)}. \tag{4.27}$$

Measuring the error in latent data space in this case leads to the objective function

$$E^{\text{lat}}(X) = \frac{1}{N} \sum_{i=1}^{N} \| x_i - \frac{\sum_{j=1}^{N} K(y_i, y_j) x_j}{\sum_{k=1}^{N} K(y_i, y_k)} \|^2, \tag{4.28}$$

$$= \frac{1}{N}\|XM_Y\|_F^2, \tag{4.29}$$

with

$$M_Y := \Big(\frac{K(y_i, y_j)}{\sum_{k=1}^{N} K(y_i, y_k)}\Big)_{ij} - I. \tag{4.30}$$

which defines a quadratic form in the latent variables and can therefore be solved by a spectral decomposition, as detailed below. The resulting UKR variant will be denoted lUKR in the following.

Note, that while the UKR variant described in the previous section is based upon the simultaneous – and in fact associated – determination of latent variable realizations and a model for f, here determination of the latent data matrix is connected to, and therefore leading to, an implicit estimation for the backward mapping g, instead. This has got important implications regarding generalization: Since no model for f is learned, generalization of the regression function to new *latent space elements* drops out. As a consequence, generalization of the backward mapping to new *observable space elements* by projection as in 4.11 is not possible, either. The implicitly learned model for the backward mapping therefore remains the only alternative. However, as indicated in 2.4, this variant can be problematic, because it is a continuous function. Overall, it is obvious that a major shortcoming of this UKR variant regards generalization.

4.3.1. Optimization

An optimal latent data matrix is given by

$$X^{opt} = \arg\min_X E^{\text{lat}}(X). \tag{4.31}$$

Relocating 4.28 to yield

$$E^{\text{lat}}(X) = \text{tr}(XM_YM_Y^TX^T) \tag{4.32}$$

shows that the problem defines a quadratic form $Q := M_Y M_Y^T$ and may be solved by an EVD as described in 3. In other words, constraining the solutions by requiring

$$X1 = 0 \text{ and } XX^T = I_q, \tag{4.33}$$

optimal latent space realizations are simply given by the eigenvectors corresponding to the q (second to) smallest eigenvalues of Q [Mei03].

While in 3.2.1 sparsity of Q was provoked automatically through the use of small neighborhood sizes ($K < N$), here the same effect needs to be achieved by using a bandwidth limited kernel function such as the Epanechnikov Kernel (4.8). In contrast to the previous section, here no compensation of the choice of a kernel bandwidth h by automatic adjustment of the latent variable norms takes place[3]. Therefore the application of this method relies on a predefined observable space kernel bandwidth. In fact, without a reasonable choice for h, no good solutions can be expected, as detailed below

The problems of lUKR with regard to generalization are visualized in figure 4.3.1. Although the embedding correctly captures the low dimensional structure (a), the ambiguity points are mapped onto scores in a wrong way so that the model fails to preserve the structure as shown in plot (c). The inappropriate regression manifold due to inappropriate scaling of the latent matrix that the EVD gives rise to is also visualized (b).

4.3.2. Choosing the Observable Space Kernel Bandwidth

The quality of the solution that the lUKR variant yields depends crucially on the choice of a suitable kernel bandwidth h. This dependency is illustrated in figure 4.9 where latent space solutions for several datasets and varying kernel bandwidths are shown. Samples from the 'halfcircle', the one-dimensional 'S-manifold', the 'spiral', and the two-dimensional 'S-manifold' were used. The noise variance was varied in three steps, using $0.0/0.1/0.2$, $0.0/0.2/0.8$, $0.0/0.05/0.08$, and $0.0/0.6/0.8$ for the respective datasets. The respective sample sizes were $N = 100$, $N = 300$, $N = 600$, and $N = 600$, for each

[3] Note that the symbol h which has beed used to denote the latent space kernel bandwidth before here denotes the observable space bandwidth. In all that follows h will denote the observable space kernel bandwidth only.

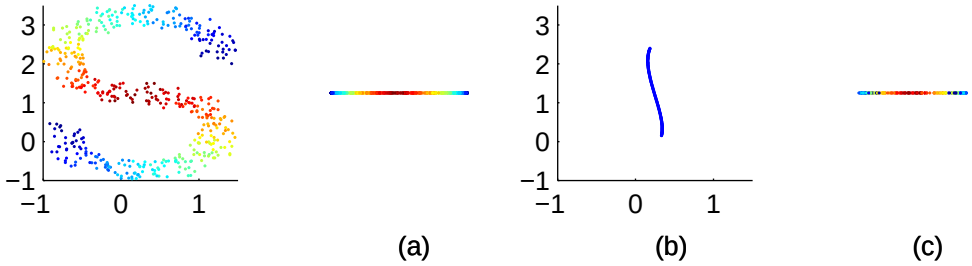

Figure 4.8.: Visualization of the inappropriate scaling and of the flawed generalization of the backward mapping for a solution that lUKR gives rise to. lUKR has been applied with kernel bandwidth $h = 0.8$ on a sample of 500 elements from the 'noise S-manifold' with $\sigma = 0.5$.

of the noise variance choices. For each dataset in addition one sparsely sampled dataset was used. The dataset sizes in that case were $N = 30$, $N = 50$, $N = 80$, and $N = 150$.

From the illustration it can be observed that those bandwidths that give rise to a solution which obviously reflects the low-dimensional structure present in the dataset reside in an interval whose width is dependent on the 'complexity' of the manifold. For the 'half-circle' dataset the (visual) quality of a solution is indifferent with regard to h. As the complexity is increased by turning to the 'noisy S' and later to the 'spiral' dataset, the interval of optimal solutions gets smaller. Note, that the absence of an observable space error criterion calls for such a visual quality measure. In 4.4 a quantitative assessment will be given.

The dependency on a parameter defining a neighborhood size in some way is characteristic of the spectral methods for NLDR in general. For the LLE algorithm, for example, (see 3.2.1) it is given as the number of nearest neighbors, K. The theoretical connection between this neighborhood parameter and appropriate latent solutions remains an open question. But some hints that at least help restrict the search space for suitable bandwidths may be derived from theoretical considerations. Some of these shall be sketched in the following.

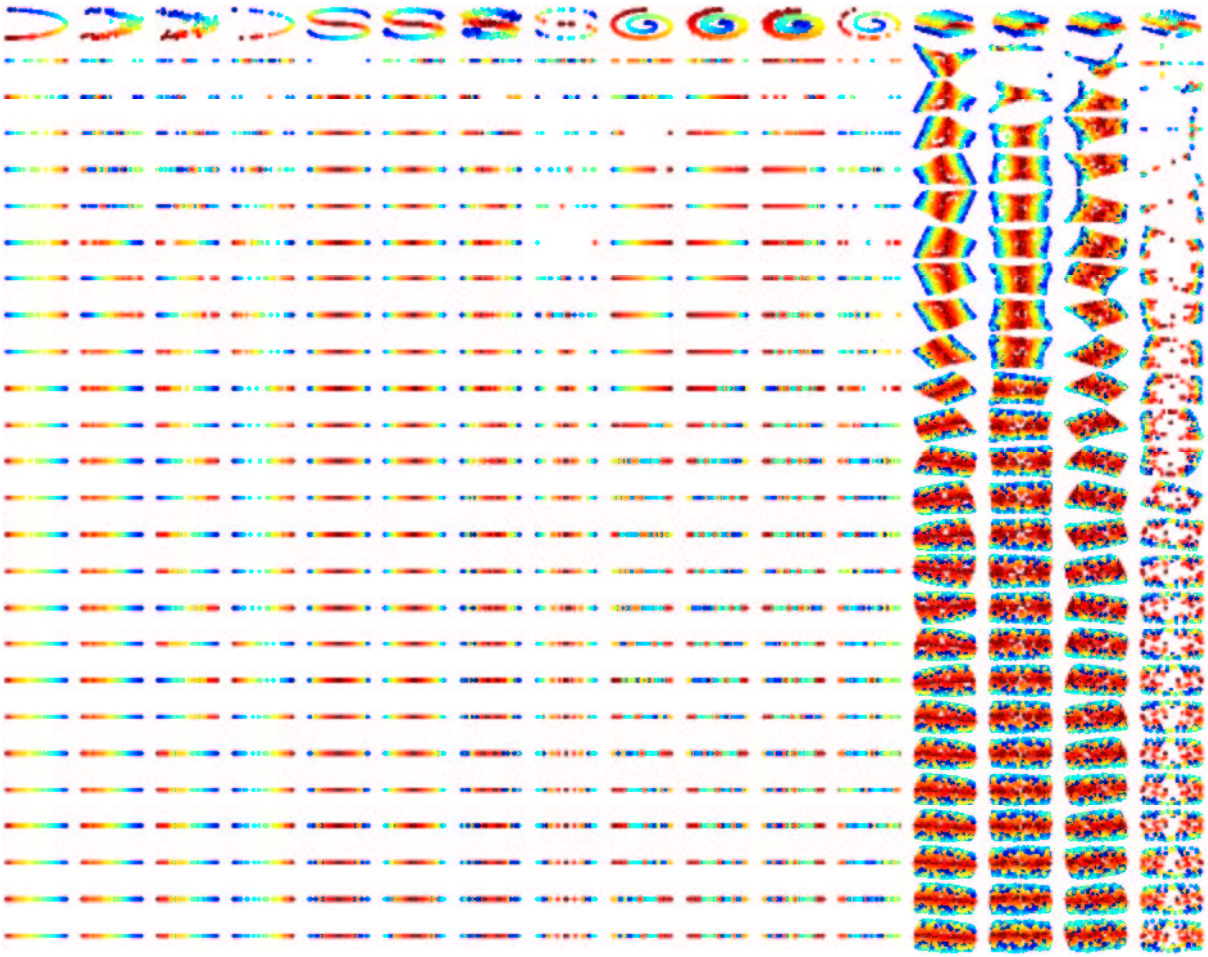

Figure 4.9.: Dependency of the lUKR solutions on the choice of the observable space kernel bandwidth. lUKR was applied with various kernel bandwidths on the different datasets visualized in the top row. The latent space realizations for increasing h are visualized beneath the visualization of each respective dataset. The range in which h has been varied is the same within each of the four classes of datasets. In detail, the ranges are $[0.14, 2.00]$, $[0.16, 4.00]$, $[0.11, 2.15]$, and $[0.61, 7.10]$ for the 'halfcircle', 'one-dimensional S-manifold', 'spiral', and, 'two-dimensional S-manifold' dataset, respectively.

Kernel Density Estimation

The most obvious solution to the bandwidth estimation problem is to take that kernel width that gives rise to an optimal *kernel density estimator*. This approach, which may be motivated with reference to the assumption that the 'real' bandwidth that the underlying density gives rise to, should be the right one this situation, too, is flawed, however.

A commonly used and simple approach to bandwidth estimation is by maximizing the *cross validation log likelihood* [Sco92]

$$L^{\text{cv}}(h) = \sum_{i=1}^{N} \log \frac{1}{N-1} \sum_{j \neq i} \mathcal{K}(x_i, x_j), \tag{4.34}$$

For the Epanechnikov kernel, this function is monotonously rising within the whole range of values for h that gave rise to the plots in figure 4.9, showing that this approach to finding a suitable bandwidth to apply lUKR has to be dismissed.

Connected Graph

A strict lower bound beyond which no suitable solutions are to be expected can be derived from an approach to dimensionality reduction referred to as *Laplacian Eigenmaps*([BN]). In drawing from Spectral Graph theory, the authors propose an algorithm that is similar to the lUKR method and is based on an EVD of a matrix derived from a kernel matrix, with the Epanechnikov Kernel replaced by a bandwidth limited Gaussian Kernel. Though the theoretical background is quite different from the regression background adopted by UKR, it suggests a lower bound for h that is straightforwardly transferable to this situation. In particular, interpreting the kernel matrix as a graph *connectivity matrix* gives rise to a lower bound as the smallest value for h that yields a fully connected graph. Values beyond this connectivity threshold yield an eigenvalue problem that is numerically unstable and that therefore needs to be dismissed. The latent space realizations that result from a too small kernel bandwidth can be seen, for example, in the top row of figure 4.9. Obviously they are not better than any randomly initialized latent data matrix.

Practically it appeared that the strategy to choose a value for h just above the connec-

tivity threshold – which has to be determined by some search procedure – yields good results the toy datasets most of the time. There have been cases, however, where this criterion did not yield appropriate results, and where a larger value for h was needed.

Observable Space Noise Variance

Another approach to the direct estimation of h may be derived from the theory of Noisy Interpolation. In [Web94] a solution is proposed to the (supervised) objective to generalize a mapping implicitly given by a set of (noisefree) input/output variables to new, unseen samples, where the new *input* data might have been corrupted by noise. For a spherical input space noise model with variance σ^2, minimization of the output space error leads to a nonparametric function equal to the Nadaraya Watson Estimator (4.6), with the kernel bandwidth h being replaced by σ^2.

This viewpoint provides a re-interpretation of the 'reversed' regression approach that the lUKR variant adopts, where the latent variables which are treated as being noiseless in the standard regression model given in 2.1 (which might still provide the conceptual background of the overall situation) are regarded as outputs, and the observables which are regarded as being corrupted by noise are treated as inputs. Thereby this viewpoint additionally provides a hint towards which values for h might prove useful by suggesting the observable space noise variance.

The visual assessment in figure 4.9 seems to contradict such an approach at first sight, since the introduction of noise obviously does not lead to a displacement of the area of optimal values for h towards larger values. Instead, rather a displacement towards smaller values seems to be the case. However, the theory of Noisy Interpolation is based on the assumption of noise-free training data, which does not hold here, of course. In order to obtain an approximation to this noise-free input data set one may use some 'data condenser' as for example the 'reduced set density estimator' proposed by [GH02]. The reduced data set together with the original data set could be used to estimate the noise variance, for example by performing local PCA on the reduced data set and estimating the variance of the original data set in the space orthogonal to the space spanned by the eigenvectors from the PCA solution. Then lUKR can be applied using the reduced data set alone as input and the estimated noise variance as kernel

bandwidth. This approach may point towards a promising direction for future research. It will not evaluated further here.

Iteratively

A way to find a suitable kernel bandwidth that is more expensive, but safer, than the direct estimation approaches is to iterate the lUKR method and choose that bandwidth that yields the best solution. The 1d-search space, that the use of spherical kernels gives rise to let this approach still be tractable for large samples sizes. For this method to be applicable, a criterion to assess the optimality of the solutions for different kernel widths needs to be defined.

One approach is to measure the resulting latent space error (4.28) after application lUKR. Empirically, this approach has not proven useful, as E^{lat}, similar to 4.34, increases monotonously within the range of useful kernel bandwidths. However, a variant that showed to be leading to satisfactory results can be obtained by a slight modification of E^{lat}. Analogous to the built in cross validation criterion described in 4.2.1 one may use a latent space built in cross validation criterion by modifying 4.28 and defining:

$$E^{\text{lat}}_{\text{cv}}(X) := \frac{1}{N}\|X\,\tilde{M}_Y\|^2_F \tag{4.35}$$

with

$$\tilde{M}_Y = \Big((1-\delta_{ij})\frac{K(y_i,y_j)}{\sum_{k\neq i}K(y_i,y_k)}\Big)_{ij} - I.$$

By choosing the bandwidth that minimizes 4.35 one can indeed obtain results that are good with regard to the visual assessment. However, as in the case of the 'connectivity criterion' described above, there are cases, where this criterion obviously underestimates the correct bandwidth, which leads to solutions that resemble the 'random' solutions depicted in the top rows of figure 4.9.

Several alternative approaches to assess the optimality of the solutions for different kernel bandwidth that rely on the *rescaling* of the latent data matrix can be suggested. One may, for example, rescale the matrix with the objective to approximate an estima-

tion of the probability density function of the observable variables by an estimation of the probability density function in latent space. Alternatively, one may first compute a local PCA solution in observable data space and then rescale the latent data matrix with the objective to approximate the local distances of the latent variable realizations obtained from PCA by the ones obtained from lUKR. The first of these approaches led to good results, practically. For the second it appears that the preprocessing by some data condensers as mentioned above is advisable. However, all these approaches shall not be described here in more detail, because there is one alternative approach that has shown to be superior to all the others. If one is willing to conduct a procedure with an iterated rescaling of the latent data matrix anyway, one may perform the rescaling with the objective to minimize the *observable space error*. This approach allows to profit from the advantages that the oUKR method has because of its observable space error criterion and has in particular proven to be more reliable than the other iterative bandwidth estimation procedures. It gives rise to a combination of the latent space and the observable space error criteria and will be described in detail in the next section.

4.4. Combination of Latent and Observable Space Error Minimization

As shown in 4.2 the mean square reconstruction error provides a reasonable and stable error criterion for learning a low-dimensional approximation to a high-dimensional data set. The price for the application of this criterion in 4.10, however, is its high computational complexity and an objective function that depends on an extremely large number of parameters and that is fraught with local minima. Minimization of this function has therefore shown to be relying crucially on a reasonable initialization for the latent data matrix. The latent space error criterion deployed in 4.3, on the other hand, while leading to an objective function amenable to optimization through an efficient and global eigenvalue decomposition, has shown to be inapplicable in determining a suitable data space kernel width, which this variant crucially depends on, however.

A way out of this dilemma is pointed out by a combination of the latent space and

observable space error criteria. Taking advantage of the computational efficiency of the spectral UKR method, we can embed this into a 1d-search to find that kernel bandwidth that gives rise to an optimal eigen solution – with optimality being defined by an observable space error criterion. In detail, for every lUKR solution that results from a specific kernel bandwidth h the observable space error criterion given in 4.10 can be determined by computing the nonparametric regression on every vector from the obtained latent data matrix. The resulting observable space reconstruction error then provides a means to assess the quality of the received eigen vector solution associated with h. This way we receive a efficient algorithm to compute a (nonparametric) principal manifold relying on a purely data driven error criterion.

Because of the arbitrary scaling of the lUKR solution, however, direct application of 4.10 would not make sense, as has been indicated above. For this error criterion to still become applicable the lUKR solution might be adapted by *minimizing 4.10 with respect to X first.* This way one receives an overall optimization scheme in which the application of lUKR might be interpreted as merely providing an *initialization* for the oUKR method. A computationally more feasible alternative would be to simply repair the actual main defect of the lUKR solution, which is indeed the inappropriate scaling, by *minimizing 4.10 with respect to the latent data matrix scaling* alone. Details for this procedure are given below. The application of the combined error criterion using rescaling alone can also be interpreted as an iteration of the computation of a lUKR solution followed by the application of a *supervised* nonparametric regression model on the resulting complete dataset, since rescaling the latent variable realizations is equivalent to finding a suitable latent space kernel bandwidth. Using a different rescaling factor for every latent space dimension, as described in the next section, is that way equivalent to using structured kernels.

Additionally applying the oUKR method in each iteration gives rise to the interpretation of the overall method providing a way to tackle the problem of global vs. local optimization by replacing the high-dimensional search space for suitable initializations for the application of the oUKR method by a one-dimensional search space. This makes an exhausting, global optimization scheme more tractable and avoids the need to reside to methods like simulated annealing, for example.

Regardless of which strategy – using the latent matrix as an initialization for further training using oUKR or just adapting scale factors – is chosen, presence of the observable error criterion makes it even more so possible to assess the goodness of different solutions *quantitatively* and thereby provides an important supplement to the mere visual assessment adopted in the previous section. Especially the different direct approaches to estimating h can be compared appropriately with this criterion, which will be done in 4.4.2.

4.4.1. Optimization

In summary, for the combined error criterion the learning task can be carried through using the following algorithm:

for $h = hmin : increment : hmax$
use lUKR with kernel width h to compute latent data matrix X
determine diagonal rescaling matrix S by minimizing (a penalized variant of) $E^{\text{obs}}(SX)$
set $X := SX$
(optional:) additionally apply observable space UKR on X
if$E^{\text{obs}}(X) < minerr$
$X^{opt} := X, minerr = E^{\text{obs}}(X opt)$
optimal embedding resides in X^{opt}

Several hints for a suitable range to search for h might be derived by the reasoning delineated in the previous section. This is crucial, since for large datasets even the EVD becomes too expensive to embed into a large number of iterations.

Rescaling of the Latent Data Matrix

Rescaling of the latent variable values can be practically realized using a scalar rescaling factor s globally applied to the latent data matrix or by determining a suitable factor s_i

for every latent space dimension i:

$$X^{resc} := SX, \text{with} S := diag(s, \dots, s) \text{ or } S := diag(s_1, \dots, s_q).$$

As stated above, to determine suitable rescaling factors the observable space error criterion can again be applied, in this case being a function of the rescaling matrix S:

$$E(S) = \sum_{i=1}^{N} \| y_i - \frac{\sum_{j=1}^{N} K(Sx_i, Sx_j) y_j}{\sum_{k=1}^{N} K(Sx_i, Sx_k)} \|^2 \tag{4.36}$$

The rescaling of the latent data matrix by minimization of 4.36 poses a nonlinear optimization problem similar to the one described in 4.2. In contrast, however, here the parameters reside in an only q-dimensional space making the optimization problem more tractable. As in the case of the observable space UKR method some gradient based optimization scheme may be applied, with the gradient of the objective function taking a similar form as in 4.2.1. It contains subtle differences, however, and shall therefore be given here in detail, as well. With M^k defined as above, it similarly holds:

$$\frac{\partial E(S)}{\partial s_j} \propto \sum_{k=1}^{N} \sum_{l=1}^{d} y_l^T M^k y_l^T \frac{\partial M^k}{\partial s_j}, \tag{4.37}$$

and here again, pre-computing particular terms can help to restrain the computational costs. In particular, by setting analogously

$$\begin{aligned}
\hat{A}_j^{kl} &:= y_l^T \frac{\partial M^k}{\partial s_j} \\
&= \sum_{m=1}^{N} y_{lm} \frac{\partial}{\partial s_j} \Big(\frac{K(x_k, x_m)}{\sum_{t=1}^{N} K(x_k, x_t)} - \delta_{km} \Big) \\
&= \frac{1}{\sum_{t=1}^{N} K(x_k, x_t)} \sum_{m=1}^{N} y_{lm} \frac{\partial K(x_k, x_m)}{\partial s_j} \\
&\quad - \frac{1}{\big(\sum_{t=1}^{N} K(x_k, x_t)\big)^2} \Big(\sum_{t=1}^{N} \frac{\partial K(x_i, x_t)}{\partial s_j} \Big) \Big(\sum_{m=1}^{N} y_{lm} K(x_k, x_m) \Big) \\
&= \frac{\hat{P}_j^{kl}}{\hat{O}^k} - \frac{(\hat{Q}_j^k)\hat{R}^{kl}}{(\hat{O}^k)^2},
\end{aligned} \tag{4.38}$$

computation of the gradient amounts to an overall complexity of $O(N^2 dq)$, with

$$\hat{O}^k := \sum_{t=1}^{N} K(x_k, x_t),$$

$$\hat{P}_j^{kl} := \sum_{m=1}^{N} y_{lm} \frac{\partial K(x_k, x_m)}{\partial s_j},$$

$$\hat{Q}_j^k := \sum_{t=1}^{N} \frac{\partial K(x_k, x_t)}{\partial s_j},$$

$$\hat{R}^{kl} := \sum_{m=1}^{N} y_{lm} K(x_k, x_m),$$

which amount to costs of $O(N^2)$, $O(N^2dq)$, $O(N^2q)$, and $O(N^2d)$, as before. A gain in efficiency here, due to a smaller number of different terms computed, gets compensated by the fact, that there is no large number of derivatives becoming zero, as was the case before. As before, Gaussian kernels may be used. The derivative of these with respect to the rescaling factors is given by

$$\frac{\partial K(Sx_k, Sx_t)}{\partial s_j} = -\frac{1}{h^2} K(Sx_k, Sx_t) \left(x_{kj} - x_{tj}\right)^2 s_j.$$

The need make use of a way to control the model complexity applies in the same way as in 4.2. Here, the built in leave-one-out cross validation variant (see 4.2) is particularly recommendable, because it avoids the need to use an external test set. This would be especially undesirable here, since usually the latent matrix rescaling will already be embedded into a search for a suitable kernel bandwidth.

4.4.2. Experiments

lUKR initialization vs. LLE initialization

Instead of using lUKR as an initialization for rescaling or for the application of oUKR, one may also use the solution that another spectral method or any other heuristic method for dimensionality reduction gives rise to, of course. Therefore it is natural to ask, how the potential of other methods to provide a suitable initialization compares to that of lUKR. A comparison between the potential of lUKR and LLE is given in table 4.1. Depicted is for different datasets and on the average for ten runs the final error E^{cv} obtained from using lUKR and LLE as an initialization for rescaling with the objective to minimize E^{cv} and additional application of oUKR. For LLE for every run the best rescaling error has been chosen that could be obtained for all neighborhood sizes K ranging from $Kmin := q$ to $Kmax := N - 1$. For UKR the best kernel bandwidth has been determined analogously by choosing the bandwidth in the range between that value that yields a 'connected graph' (see 4.3.2) and that value that causes at least one neighborhood to cover the whole dataset, using the Epanechnikov Kernel.

The results show that lUKR clearly outperforms LLE with regard to its potential

	halfcircle		scurve		spiral	
σ^2	0.0	0.2	0.0	0.5	0.0	0.05
LLE + rescaling	0.00060	0.0735	0.3279	0.4677	0.2364	0.2453
lUKR + rescaling	0.00035	0.0472	0.1107	0.1911	0.1971	0.2057
LLE + oUKR	0.00059	0.0300	0.0790	0.0725	0.0582	0.0400
lUKR + oUKR	0.00035	0.0218	0.0481	0.0685	0.0319	0.0369

Table 4.1.: Comparison of LLE and UKR as a means to provide an initialization for subsequent minimization of E^{cv} with regard to the latent variable scale only (two top rows) and with regard to the latent variables (bottom two rows.)

as an initialization for rescaling and also for subsequent application of oUKR. With regard to mere rescaling the resulting error is approximately half as high for lUKR as compared to LLE on the average for all datasets. The performance gain is not as high but still obvious regarding subsequent application of oUKR. Here, interestingly an extreme general performance gain – independent of the initialization – as compared to mere rescaling is obvious for all datasets except for the 'halfcircle.'

Additionally, it can be observed that after mere rescaling there is no significant difference in the performance gain with regard to presence or absence of noise. But after applying oUKR the performance gain that lUKR gives rise to is smaller as compared to mere rescaling in the presence of noise than without noise. In other words, oUKR is able to 'repair' to some extend the defect in the spectral solutions that noise gives rise to and this capability of repairing is stronger for LLE initializations than for lUKR initializations.

Comparison of different Bandwidth Estimation Strategies with regard to Observable Space error

Section 4.3.2 has examined the dependency of the lUKR solutions on the choice of a suitable observable space kernel bandwidth and proposed different strategies for the selection of a suitable bandwidth. The absence of a quantitative performance measure demanded a visual assessment of the different solutions. The presence of the observable space error criterion now allows for a *quantitative* investigation of this dependency. The results of such an investigation are depicted in figure 4.10. In extension of the visual

evaluation illustrated in figure 4.9, here the different bandwidths and estimation strategies are related to E^{cv}, using the same toy datasets. It is obvious that the observable space error clearly reflects the findings that were derived from the visual presentation. The fact that the range for suitable solutions is dependent on the complexity of the manifold is reflected in the graphs, as is the observation that for the 'halfcircle' dataset the quality of the lUKR solution is indifferent with regard to h. The numerical instability for small values for h is reflected in a very large value for E^{cv} that obviously abruptly drops when h exceeds some critical value. In the plots that show E^{cv} also the threshold values that yield a connected graph are shown in form of a green 'x'. The values were obtained by a search procedure: h was initialized by that value that gives rise to at least one non-zero kernel function value, if evaluated on every datapoint and its corresponding nearest neighbor using the Epanechnikov kernel and was increased with factor 1.01 until the graph connectivity criterion (see 4.3.2) was met. Generally, the connectivity threshold corresponds quite well to the critical value that yields the obviously stable solutions. In some cases, however, it underestimates the critical value, confirming the aforementioned findings. The dependency of $E^{\text{lat}}_{\text{cv}}$ on h is depicted beneath each plot for E^{cv}. Analogous conclusions can be drawn from these: The minimum of $E^{\text{lat}}_{\text{cv}}$ corresponds to the minimum of E^{cv} quite well, too, but the tendency to underestimate – also mentioned before – is obvious here, as well. Overall, it becomes obvious that both criteria described may be used as a lower bound to search for a value for h with the objective to minimize E^{cv}, but they are not useful to directly derive such a value. The evaluation of the other criteria for determining a suitable value for the kernel bandwidth that could be rejected already (such as the monotonously rising log-likelihood, see 4.3.2) shall not be described in detail, since a comparison to E^{cv} has only been able to confirm these. Overall, it can be concluded that none of the approaches to bandwidth selection proposed in 4.3.2 is safe enough to represent a serious alternative the strategy to use the observable space cross validation error and choose a suitable kernel bandwidth iteratively.

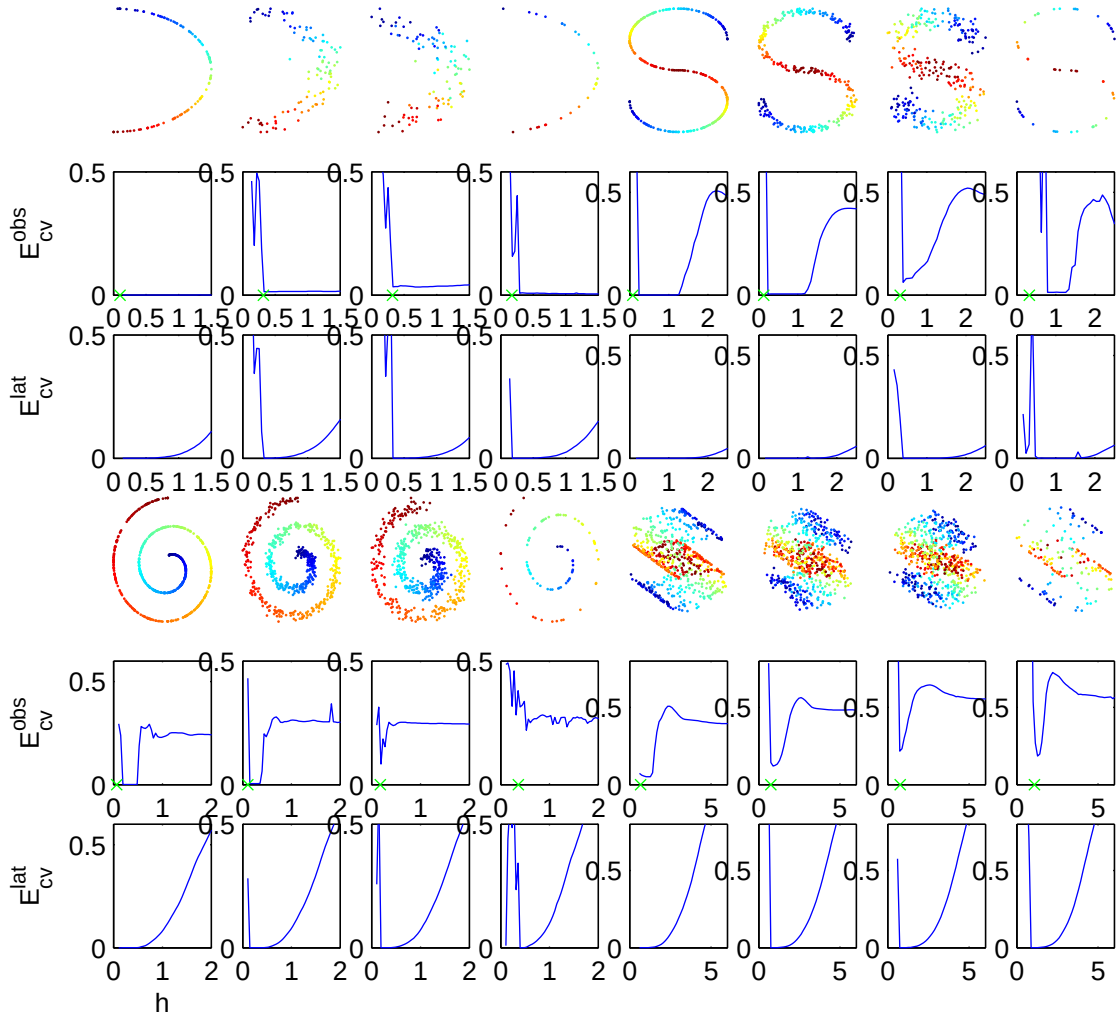

Figure 4.10.: Dependency of the observable space and the latent space built in leave-one-out cross validation errors on h. Beneath the scatter plot for each dataset E^{cv} and $E^{\mathrm{lat}}_{\mathrm{cv}}$ are depicted for varying kernel bandwidths. In addition, the 'connectivity threshold' is marked in the plots for E^{cv}.

	iris		glass		diabetes	
q	1	2	1	2	1	2
GTM	2.7020	0.9601	8.0681	1.9634	6.7100	1.8822
PPS	2.5786	0.8757	7.9465	1.8616	6.5509	1.8202
lUKR	2.1164	1.2667	7.6977	6.1930	6.4130	4.9041
lUKR + oUKR	1.2017	0.5961	5.8665	4.3678	5.8018	3.9466
oUKR HOM	0.9414	0.5651	4.7515	3.6402	6.2488	3.9619

Table 4.2.: Evaluation of the test error that UKR gives rise to as compared to the test error that PPS gives rise to on several benchmark datasets. Key: GTM (Generative Topographical Mapping), PPS (Probabilistic Principal Surface), lUKR (UKR using latent space risk minimization), oUKR (UKR using observable space risk minimization), lUKR + oUKR (oUKR with initialization provided by lUKR), oUKR HOM (oUKR with homotopy scheme).

Fitting 'Nonparametric Principal Surfaces' to Real World Data

Since the use of the observable space error criterion gives rise to an interpretation of the UKR model as a nonparametric variant of a principal curve or surface, an evaluation of the performance of this model as compared to a parametric counterpart on some benchmark datasets suggests itself. For this end the UKR model was trained on three publicly available datasets from the 'UCI Machine Learning repository': 'iris,' 'glass' and 'pima indians diabetes,' which consist of 150, 214, and 768 elements in 4, 9, and 8 dimensions, respectively (see [CG01] for more details). The test conditions that have been used are similar to those used in [CG01], where a new approach to principal surfaces named 'Probabilistic Principal Surfaces' (PPS) is evaluated using these same datasets. In detail, the PPS model is a modified version of the 'Generative Topographic Map' (GTM) (see, e.g. [BSW98]) and degenerates to this for specific (hyper-) parameter settings. In table 4.2 the performances of UKR are shown along with the results the authors obtained using (generic) GTM, as well as PPS. One important difference regarding the coming off of the results that should be noted, is that the results concerning UKR have been obtained by training and adjusting hyper-parameters on a training set only. The shown error represents the resulting error on an independent test set. The columns for PPS and GTM, in contrast, show the test set error, where the model hyper-parameters have been adjusted using *this same test set*.

In detail, the UKR model was trained, using the following test conditions: for a total of 25 runs, each (pre-whitened) dataset has been partitioned randomly into training and test set of the same size. Then, for latent space dimensions $q = 1$ in one setting and for $q = 2$ in another, scores have been computed using the algorithm given in 4.4.1, where the observable space error has been used for *rescaling* only. The error that projection of the corresponding *test* set on the resulting manifold gives rise to, averaged over all 25 runs, is shown in table 4.2, third row. The results that a further training on the resulting scores using oUKR (4.2) gives rise to are shown below, and the results obtained using oUKR alone, by applying the homotopy scheme with a penalized built in leave-one-out cross validation objective function, in the last row.

From the table it becomes obvious that in the case of $q = 1$ the UKR results clearly outperform those obtained from GTM and PPS, despite the parameter optimization on the test set that gave rise to these results as mentioned above, suggesting UKR to be a reasonable alternative to parametric models, as a first conclusion. While for $q = 2$ UKR is not able to keep up with the extreme performance gain on the 'glass' and the 'diabetes' dataset, which may be probably due to test set optimization scheme, on the 'iris' dataset the best UKR results outperform the other two methods here, as well.

More interestingly, it can be observed that, although mere application of lUKR yields decent results already, additional training using oUKR is able to dramatically further improve these. On the 'iris' dataset the average error is even minimized approximately by factor 0.5 for both latent space dimensionalities. The superiority of the oUKR variant with regard to the observable space test error is further backed up by the results obtained using this method alone. Here, a further improvement can be observed in all but one setting, which is the one-dimensional approximation of the 'diabetes' dataset, where in the two-dimensional case, however, this method is even able to come close to the GTM/PPS results.

Overall, the results obtained by these experiments abet the conclusion that UKR is reasonable choice indeed when it comes to approximating a high dimensional dataset by some lower dimensional manifold. The much better results of the oUKR variant clearly compensate for the higher computational effort. In a practical situation the observable space dimensionality and data set size may demand to weigh up the the computational

resources available against the generalization performace needed by the application at hand. Here, the combination of lUKR and oUKR might provide a reasonable compromise.

5. Applications

In the following some applications of NLDR as achieved by using UKR shall be demonstrated. It has been argued before that different kinds of applications demand a solution of the different facets of the overall problem of dimensionality reduction. The applications demonstrated here have been chosen to reflect this categorization. They have therefore been drawn from a uniform domain in order to abet a convenient comparison of the different facets of dimensionality reduction. In particular, a *visual* (image processing) domain has been chosen where each observable space element represents a black-and-white image by holding its pixel intensities encoded as real numbers. The advantage of such a visual domain is that it allows for a particularly straightforward application of the involved methods and thereby prevents the essential concepts from being obscured by technical details. Nevertheless, it should be noted that these applications are transferable to any other domain where some vector valued data representation is possible.

5.1. Visualization

An application of dimensionality reduction that does not necessarily require an estimation of the coding or decoding function, since it gives rise to methods that only depend on the presence of latent space realizations, is *visualization*. Being a regularly used instrument in the area of 'datamining,' visualization provides a means to discern patterns and structures in complex data that can be used to to gain insights and draw inferences about this data. It is regularly applied in many areas where there are large amounts

of data that need to be made sense of in some way. A common procedure used in visualization is to provide a representation for a given dataset that facilitates the perception of neighborhood relations or the clustering structure of the data. In order to make such properties of the data accessible to human perception, a *lower* dimensional representation is particularly helpful and gives rise to the deployment of methods for dimensionality reduction.

Examples of the visualization of a dataset consisting of *face images* are shown in figures 5.1 and 5.2. The dataset consists of 400 grayscale images taken from the Cambridge Olivetti Research Lab (ORL) database of faces, cut down in size to 64×64 pixels. There are 10 images of each of 40 different persons. First lUKR has been applied using the combined error criterion described in section 4.4. The optimal kernel bandwidth $h = 3.1127 \cdot 10^3$ with regard to the observable space error was obtained by using 30 search steps. Then the resulting latent data matrix has been used as an initialization for oUKR, that was applied using the built in leave-one-out cross validation criterion. While the two-dimensional representation obtained from applying lUKR (figure 5.1) and the one that further training with oUKR gives rise to (figure 5.2) obviously both capture the structure of the data quite well, it is also obvious that they are quite different. Not only is the average scale of the latent space realizations obtained from training with oUKR approximately ten times higher than the lUKR initialization, but the datapoints are also spread out much more homogeneously for the first. In addition it is obvious that the grouping of the datapoints with regard to the different subjects is reflected in both low-dimensional representations, but much more clearly for the oUKR solution. For the lUKR solution, on the contrary, the tendency of the scores of some subjects to spread out along the horizontal axis is notable.

The degree of freedom within the dataset that is captured in the solution this way corresponds to head rotations. This becomes clear from figure 5.3, in which the original images of some of the subjects, sorted with regard to the score in the first latent space dimension of the lUKR solution, are depicted. This *automatic indexing* that the lUKR solution thereby gives rise to points towards another major application area of NLDR in general. Interestingly, the tendency of oUKR to distribute the data more homogeneously, while at the same time concentrating more on grouping structure, in this case

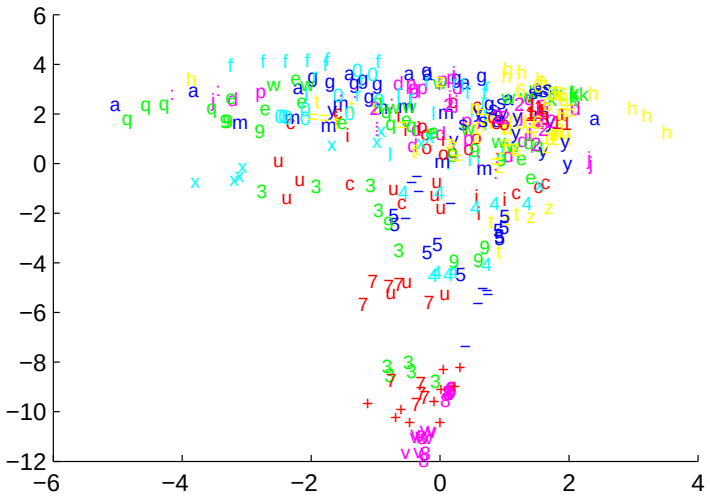

Figure 5.1.: Two-dimensional representation of the whole 'olivettifaces' dataset, using lUKR. Each of the 40 distinct persons is represented by one symbol (the letters [a-z], digits [0-9] and in addition the symbols [+], [-], [:], [=], and [*] have been used). In addition, for convenient visualization the symbols have been colored randomly, but consistently for each person.

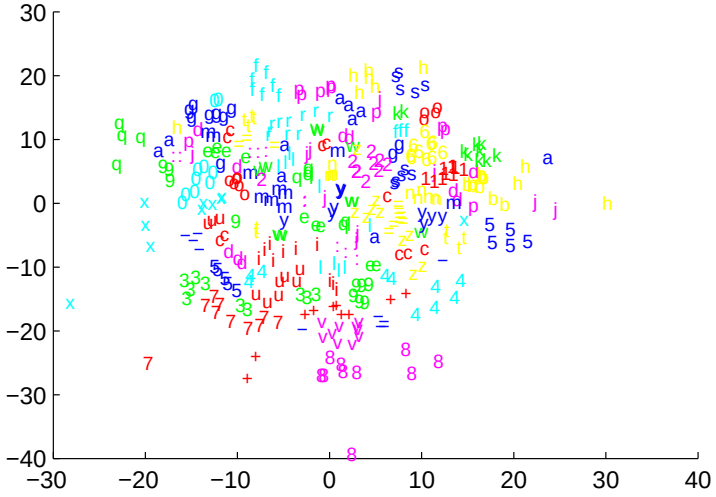

Figure 5.2.: Two-dimensional representation of the 'olivettifaces' dataset obtained by using oUKR with the lUKR solution depicted in figure 5.1 used as an initialization.

Figure 5.3.: Automatic indexing: Seven faces from the olivettifaces dataset sorted according the first eigenvalue of the lUKR solution.

destroys this specific capacity. Experiments showed that it is not possible to achieve this kind of indexing using the oUKR representation. On the other hand, this variant would be much more useful as a kind of preprocessing for some kind of *classification* with regard to the different subjects, for example. In general, it becomes obvious that despite the advantages that the observable space error criterion has, there are cases in which the spectral UKR variant is able to capture some properties of the data distribution in a way that may be more valuable for some applications.

An example of the visualization of a dataset of grayscale images of handwritten digits is shown in figures 5.4 and 5.5. The images were taken from the National Institute of Standards and Technology (MNIST) database of handwritten digits. A subset of 1500 images was randomly chosen from the database. The subset thus contains approximately equally sized sets of images of the ten represented classes (digits [1] to [10]). From figure 5.4, in which the two-dimensional representation of a subset containing the [0]s, [1]s, and [7]s is depicted, it is obvious that not only a grouping with respect to the different classes takes place, but also the inner-class variability is captured. This effect becomes even more obvious in figure 5.5, where the two-dimensional representation of a subset containing only the [8]s is shown, which was obtained by also using only this class for training.

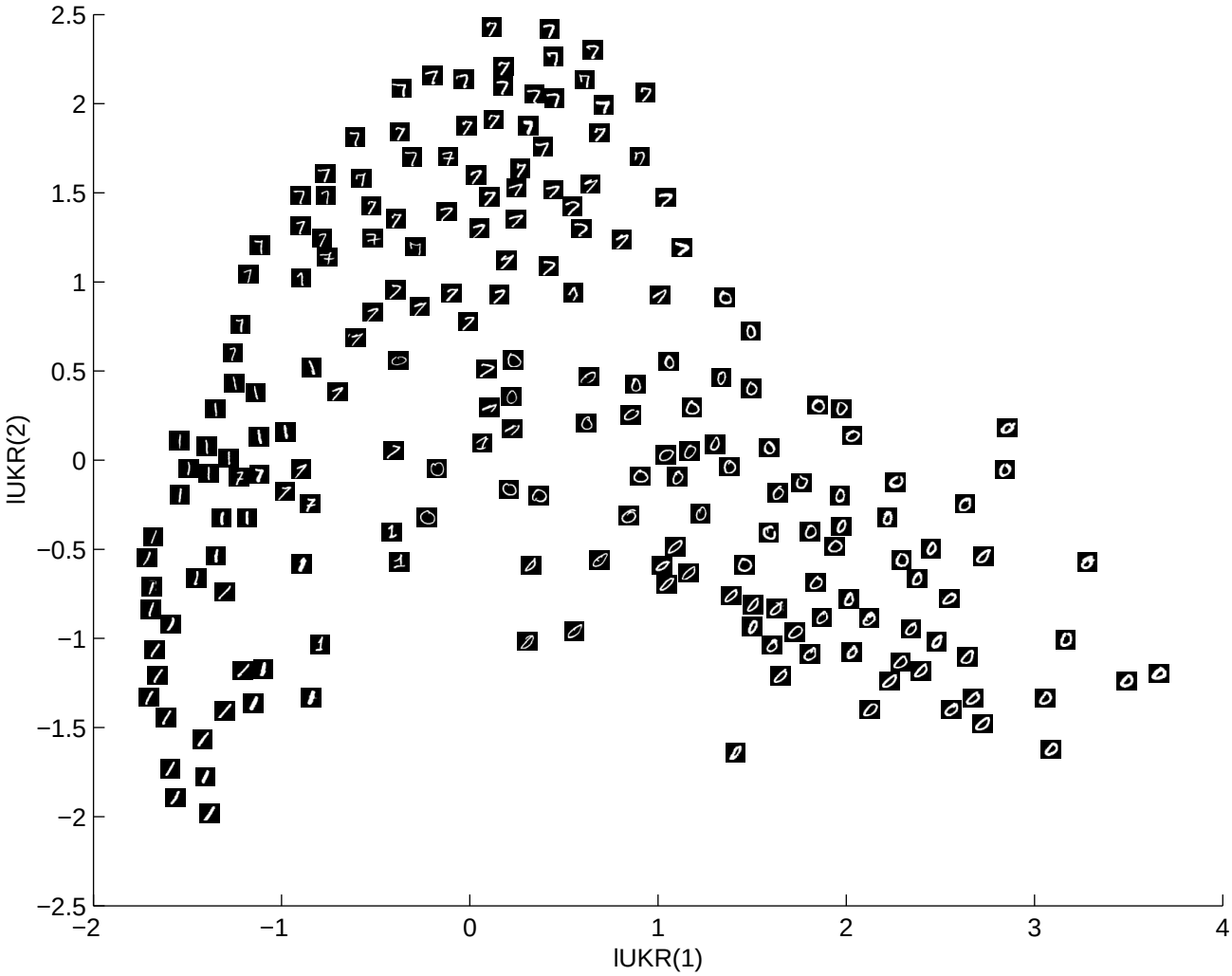

Figure 5.4.: Visualization of a subset of the MNIST digits dataset using the original images. A two-dimensional representation of a randomly chosen $N = 1500$ elements comprising subset of the whole dataset was computed using the combined error criterion described in section 4.4 with 20 iterations to search for an optimal h. Depicted are randomly chosen subsets of the elements belonging to the classes [0], [1], and [7]. The visual representation was created by randomly choosing latent space elements and displaying the corresponding observable space element as a grayscale image at the latent space coordinates only in the case that no other element within a pre-specified area around the chosen element has already been chosen before. Otherwise the latent space element has been rejected.

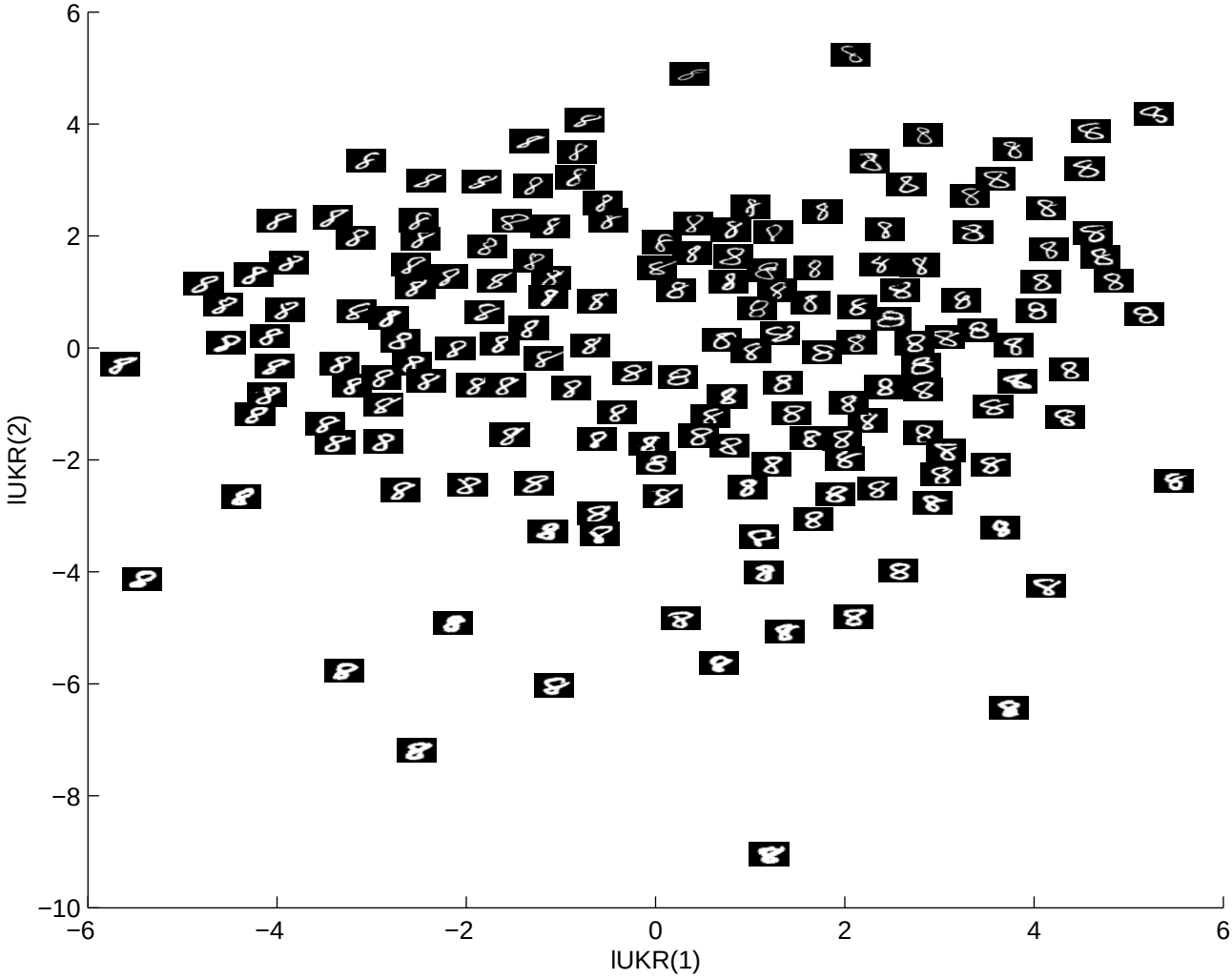

Figure 5.5.: Visualization of a subset of the MNIST digits containing only [8]'s. lUKR was used with $h = 3.2 \cdot 10^3$. The visualization strategy described in the caption of figure 5.4 has been used here, too.

5.2. Pattern detection

If models for both the coding and decoding function are available, some kind of *pattern detection* becomes possible. The term pattern detection here refers to the problem of discovering those data instances within a large dataset that have specific predefined properties. It can be interpreted as a special binary classification problem where the membership to a specific class of objects vs. non-membership shall be determined. Since the training data in such a setting only comprises the positive examples, it is often referred to as 'one-class' problem in the literature.

Dimensionality reduction can be used for the purpose of detection by measuring the error $E^{\mathrm{proj}}(y) := \|y - f(g(y))\|^2$ that the projection of a data vector y onto the manifold obtained by training on class-members only gives rise to and then, subject to some predefined threshold parameter α, asserting class membership, iff $E^{\mathrm{proj}} <= \alpha$ holds.

Here, this strategy has been used for the detection of handwritten digits from the MNIST-dataset described above. More specifically, the problem of detecting instances that represent an [8] has been tackled. For this purpose UKR has been deployed to train a manifold using 80 randomly chosen pictures that represent an [8]. Then another subset (disjunct to the training set) comprising 1000 randomly chosen elements was used to assess the detection performance. The results are shown in figure 5.6 in the form of ROC plots. ROC ('Receiver-Operating-Characteristics') plots provide a straightforward, visual way to assess the quality of a pattern-detection system. In their most frequently used variant they display the (normalized) ratio of class members that have been correctly classified as such (called 'true positives') and non-members that have wrongly been classified as class members (called 'false positives') for various threshold values.

Figure 5.6 shows the ROC plots that result from using the two different models for g – the projection model (4.11) and the 'feed forward' model (4.27) – in the top and in the bottom row, respectively. The latent space dimensionalities $q = 1$ and $q = 2$ were used. For a latent data matrix obtained from using lUKR and another matrix obtained from further training with oUKR the ROC plots are depicted. In addition the ROC plot that results from a nearest-neighbor approach to detection is shown as

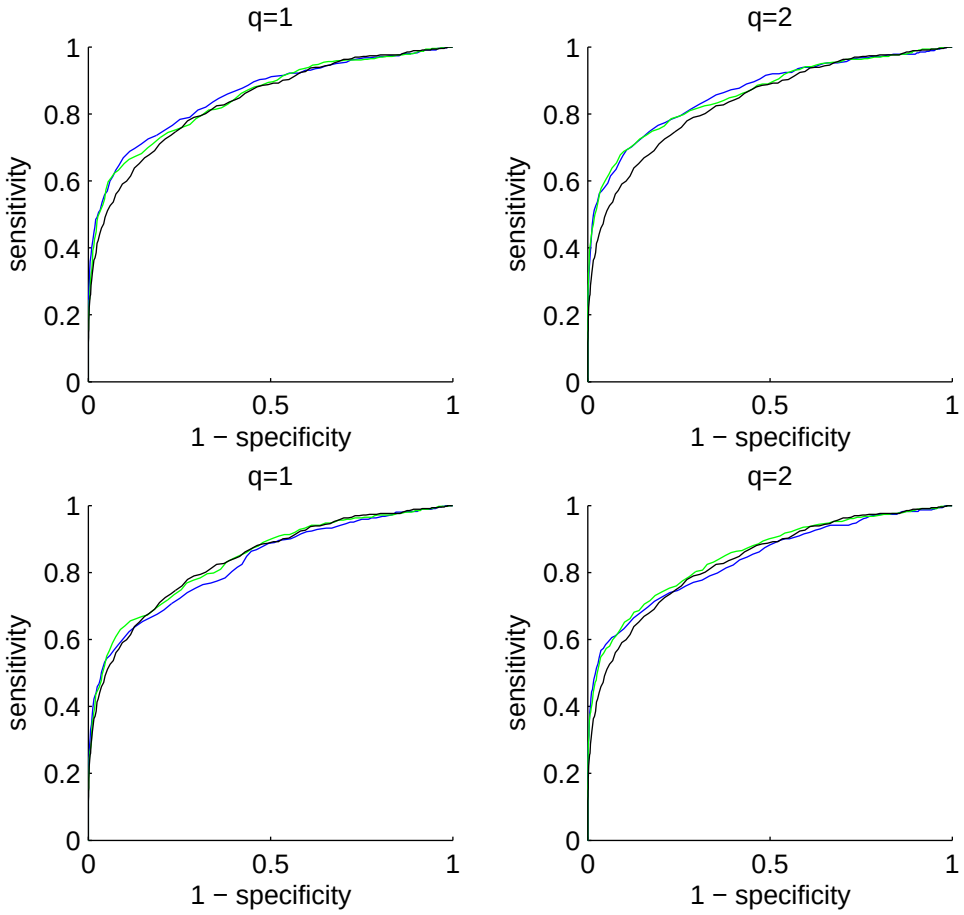

Figure 5.6.: ROC plots to assess the performance of UKR used as a method for pattern detection. The axes labeled ('1 - specificity') represent the relative number of false positives, the axes labeled ('sensitivity') the relative number of true positives. The top two plots show the performance using UKR with g modeled by projection, the bottom two plots the performance that results from modeling g in a 'feed forward' fashion. The results for lUKR are depicted in blue, the results for oUKR in green and as a reference the results for a nearest neighbor approach to detection are depicted in black in each plot.

a reference. The nearest-neighbor method amounts to simply replacing $f(g(y))$ in the definition of E^{proj} by the nearest neighbor of y among the training data. The plots reveal a slight gain in performance for UKR as compared to the nearest-neighbor approach in the case that g is modeled by projection. In the other case no performance gain can be observed. It is also obvious that the improvement is more significant for $q = 2$ than for $q = 1$, indicating that further raising the latent space dimensionality may lead to a further improvement. Interesting is that the lUKR approach used alone is able to slightly outperform the oUKR approach in the case where g is modeled by projection. To answer the question if the detection application represents another example where the oUKR method is not superior to mere application of lUKR (with subsequent rescaling) would demand a further investigation, however.

5.3. Pattern production

Once the degrees of variability in a given dataset have been revealed it is possible to create *new* elements belonging to the same kind by producing observable space vectors that obey the respective constrictions of variability. If a model for f is present, this is achieved straightforwardly by producing latent space elements and 'plugging' these into the learned model. This way simple linear operations in latent data space give rise to possibly highly nonlinear operations in observable space. Above and beyond this, some kind of *interpolation*, *extrapolation* or *analogy making* with regard to the training data elements can be achieved, for example, as pointed out in [TdSL00].

An example of obtaining new data elements by interpolation using the face images-dataset from above is given in figure 5.3. The top row shows the effect of interpolating in observable space between two pictures of the same person where the viewing direction is different (left-most and right-most picture), resulting in a simple cross-fade. The second row shows the result of interpolation in latent space which corresponds to a non-linear transformation in observable space that obviously captures the head-rotation that naturally connects the two face examples far better. The two rows below analogously show the effect of interpolating between face images of different *persons*. Here, again the superiority of latent space interpolation is visible.

Figure 5.7.: Interpolation in observable space vs. interpolation in latent space.

An additional example for the production of new patterns on the grounds of a set of examples is shown in figure 5.8. The example was produced by combining interpolation with *similarity based retrieval*, another important application of NLDR that can only be touched on here. The figure depicts the detailed screens of a movie that show the gradual deformation of a face, also known as 'morphing.' The face deformations movie starts with a woman's face (top left corner) which is gradually deformed to yield the face of another person, which is deformed again, and so on, until after 13 stages the first picture is obtained again.

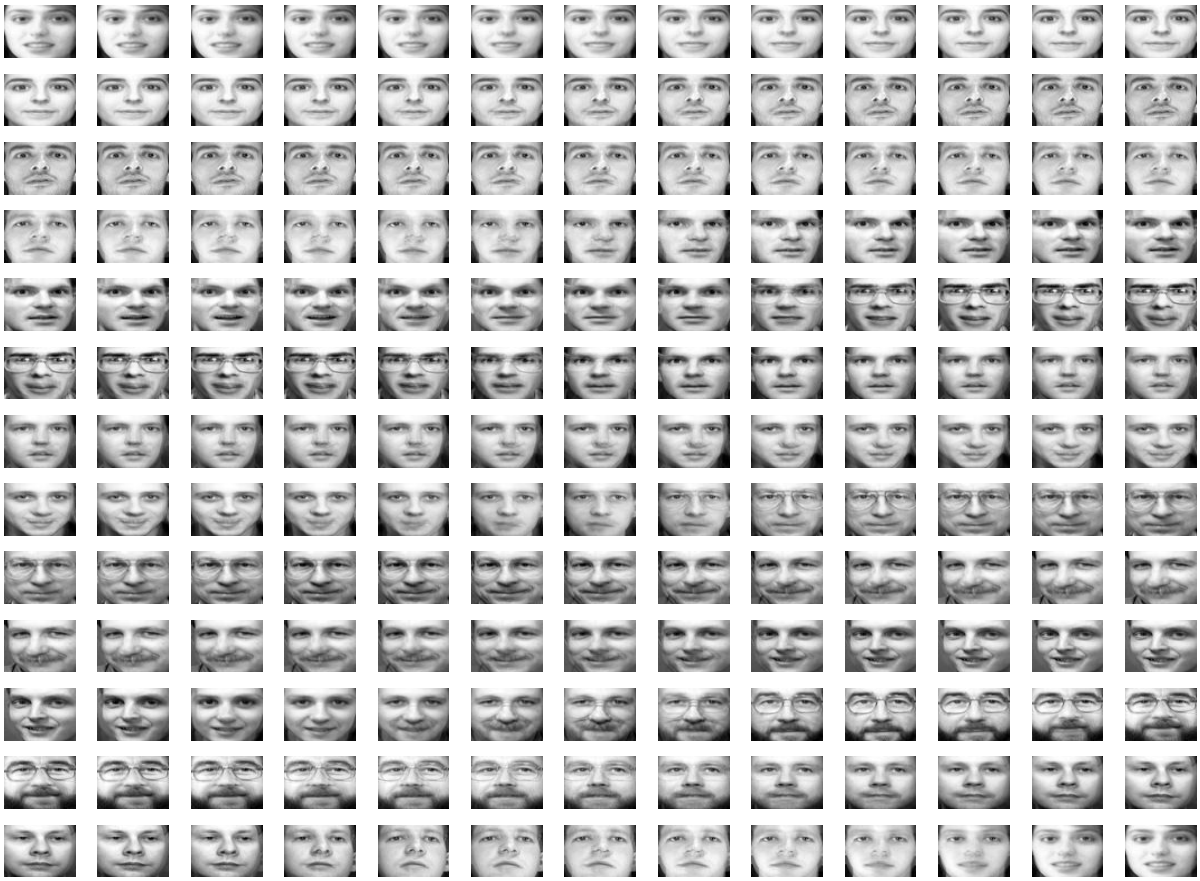

Figure 5.8.: ’Face Deformations’

6. Conclusions

This thesis evaluated the potential of Unsupervised Kernel Regression as a method for Nonlinear Dimensionality Reduction.

NLDR was exposed to be consisting of the problems of (i) determining a low-dimensional dataset, (ii) determining a nonlinear latent to observable space mapping f, and (iii) determining a nonlinear observable to latent space mapping g, where depending on the application different combinations of (i), (ii) and (iii) might be demanded. Furthermore it has been argued that in case a model for f or g is needed, the problem of *generalization* becomes a subject matter of crucial importance.

The two methods the UKR framework gives rise to, termed oUKR and lUKR, and technical issues regarding their implementation have been investigated, such as nonlinear optimization and complexity control for the first, and kernel bandwidth selection for the second. In addition, the specific characteristics of the two approaches with regard to generalizing the involved mappings to unseen data have been pointed out. Specific shortcomings of the two methods, such as the computational burden for oUKR, and for lUKR the dependency on a suitably chosen observable space kernel bandwidth and the lacking capacity for generalization led to considerations regarding a consolidation of both. The resulting algorithm and the UKR approach generally have been proven to be profitable in several applications.

As the restricted scope of this thesis admitted only a brief early investigation of the potential of the UKR approach, some issues have only been touched and others tackled rudimentarily, here. Among the issues that are in need of further investigation are in particular (i) the use of bandwidth limited kernel functions to restrain the computational

burden of oUKR, as pointed out in 4.2, (ii) particularly suited nonlinear optimization strategies in general to further cope with the computational complexity of this UKR variant, and (iii) the different bandwidth selection strategies for lUKR, such as the use of the observable space noise variance in cooperation with some 'reduced set density estimators,' as described in 4.3.

In addition, several promising variations and extensions of the UKR method certainly deserve a further exploration. Among the topics that point towards promising directions for future research are in particular (i) the generative variant of oUKR pointed out in 4.2, (ii) the introduction of top-down influences that the oUKR method allows for by making use of an accordingly customized complexity control referred to in the same section, and (iii) the recently introduced third UKR version, that supplements oUKR and lUKR by performing 'Feature Space Error Minimization' in some higher dimensional feature space [Mei03] through the use of the 'kernel trick' (see 3.2.3).

Overall, it became clear that UKR is a valuable method for NLDR. There has been a longstanding and strong demand for such methods, that hitherto could not be met with complete satisfaction, because of the aforementioned shortcomings of the methods available today. Therefore, especially the abundance of applications, that NLDR gives rise to ask for a further investigation and point towards far-reaching developments to be made in the future.

Bibliography

[BH02] Matthew Brand and Kun Huang. A unifying theorem for spectral embedding and clustering. Technical report, 2002.

[Bil97] J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models, 1997.

[Bis96] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1996.

[BN] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering.

[BSW98] Christopher M. Bishop, Markus Svensén, and Christopher K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10(1):215–235, 1998.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CG01] KuiYu Chang and Joydeep Ghosh. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):22–41, 2001.

[Cou96] C. Couvreur. The em algorithm: a guided tour, 1996.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[DLR77] A. P. Dempster, N. Laird, and D. Rubin. Maximum likelihood for incomplete data via the EM algorithm. *J. of the Royal Statistical Society, ser. B*, 39:1–38, 1977.

[GH02] Mark Girolami and Chao He. Probability density estimation from optimally condensed data samples. Technical report, 2002.

[GSSSK00] R.L. Goldstone, M. Steyvers, J. Spencer-Smith, and A. Kersten. Interactions between perceptual and conceptual learning. In E. Dietrich and A. B. Markman, editors, *Cognitive Dynamics: Conceptual Change in Humans and Machines*. Lawrence Erlbaum and Associates, 2000.

[HJ94] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, New York, NY, USA, 1994.

[HS89] T. Hastie and W. Stuetzle. Principal curves. *JASA*, 84:502 – 516, 1989.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer, 2001.

[Hyv99] A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[Jol86] I. T. Jolliffe. *Principal Component Analysis*. Springer, 1986.

[KKLZ00] Balazs Kegl, Adam Krzyzak, Tamas Linder, and Kenneth Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[Koh95] Teuvo Kohonen. *Self-organizing maps*. Springer, Berlin, 1995.

[Kra91] M.A. Kramer. Non linear principal components analysis using autoassociative neural networks. *AIChe*, J 37:233–243, 1991.

[Krz96] W. J. Krzanowski. *Principles of Multivariate Analysis. A User's Perspective*, volume 3 of *Oxford Statistical Science Series*. Oxford University Press, Oxford, 1996.

[Mal98] E.C. Malthouse. Limitations of nonlinear pca as performed with generic neural networks. *IEEE Transactions on Neural Networks*, 9(1), 1998.

[Mei00] P. Meinicke. *Unsupervised Learning in a Generalized Regression Framework*. PhD thesis, Bielefeld University, 2000.

[Mei03] P. Meinicke. Unsupervised kernel regression, 2003. submitted.

[MN88] J.R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, New York, 1988.

[MS94] Thomas M. Martinetz and K. J. Schulten. Topology representing networks. *Neural Networks*, 7(2):507–522, 1994.

[MS00] Todd K. Moon and Wynn C. Stirling. *Mathematical methods and algorithms for signal processing*. Prentice Hall, 2000.

[Pap91] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 3rd edition, 1991.

[Rip96] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

[RS00] S.T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.

[Sco92] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, New York, 1992.

[Sea82] Shayle R. Searle. *Matrix Algebra Useful for Statistics*. John Wiley & Sons, New York, NY, USA, 1982.

[SKK] Markus Siegel, Konrad P. Körding, and Peter König. Integrating top-down and bottom-up sensory processing by somato-dendritic interactions.

[SSM99] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.

[TdSL00] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2000.

[VVK02] J.J. Verbeek, N. Vlassis, and B. Kröse. Coordinating Principal Component Analyzers. In J.R. Dorronsoro, editor, *Proceedings of International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science, pages 914–919, Madrid, Spain, August 2002. Springer.

[Web94] A. R. Webb. Functional approximation by feed-forward networks: A least-squares approach to generalization. *IEEE Transactions on Neural Networks*, 1994.

A. Generation of the Toy Datasets

The preceding chapters used several two- and three-dimensional toy datasets for illustration purposes. In the following the generation of these datasets will be described.

All datasets have been generated by a random vector y or are given by the union of two sets generated by two random vectors y_1 and y_2. For the different datasets used, the random vectors are defined as follows:

- 'halfcircle:'

$$y = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix} + u, \tag{A.1}$$

- '2 halfcircles:' The same as the 'halfcircle' dataset, but with a different interval for ϕ (see below).

- 'one-dimensional S-manifold' or 'S-curve'

$$y_1 = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix} + u, y_2 = \begin{bmatrix} -\cos(\phi) \\ 2 - \sin(\phi) \end{bmatrix} + u, \tag{A.2}$$

- 'two-dimensional S-manifold[1]:'

[1]The definition for the S-curve dataset is the same that has been used by [RS00] for illustration purposes.

$$y_1 = \begin{bmatrix} \cos(\phi) \\ \chi \\ \sin(\phi) \end{bmatrix} + u, y_2 = \begin{bmatrix} -\cos(\phi) \\ \chi \\ 2 - \sin(\phi) \end{bmatrix} + u, \tag{A.3}$$

- 'spiral:'

$$y = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix} \mathrm{diag}(\frac{\phi}{4\pi} + 0.2) + u, \tag{A.4}$$

In all cases, the distributions of the random variables ϕ and χ are uniform on an interval that depends on the dataset. The intervals used for ϕ for the different datasets are (in the same order as the definition of the random vectors): $[0, \pi]$, $[0, 0.6\pi] \cup [\pi, \pi + 0.6\pi]$, $[-0.75, 0.75]$, $[-0.75, 0.75]$, $[0, 4\pi]$. The interval for χ is $[0, 6.0]$.

The random vector u is spherically Gaussian with standard deviation σ in each direction.

Versicherung

Versicherung gemäß Paragraph 19, Absatz 9 der Diplomprüfungsordnung für den Studiengang Naturwissenschaftliche Informatik an der Technischen Fakultät der Universität Bielefeld vom 10. Januar 1996.

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig erarbeitet habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Bielefeld, 12. Februar 2003

Roland Memisevic