Principal Surfaces from Unsupervised Kernel Regression

Peter Meinicke, Stefan Klanke, Roland Memisevic and Helge Ritter

pmeinic@gwdg.de, {sklanke,helge}@techfak.uni-bielefeld.de, rmemisev@cs.toronto.edu

Abstract

We propose a nonparametric approach to learning of principal surfaces based on an unsupervised formulation of the Nadaraya-Watson kernel regression estimator. As compared with previous approaches to principal curves and surfaces the new method offers several advantages: first of all it provides a practical solution to the model selection problem, because all parameters can be estimated by leave-one-out cross-validation without additional computational cost. In addition, our approach allows for a convenient incorporation of nonlinear spectral methods for parameter initialization, beyond classical initializations based on linear PCA. Furthermore, it shows a simple way how to fit principal surfaces in general feature spaces, beyond the usual data space setup. The experimental results illustrate these convenient features on simulated and real data.

1 Introduction

The close relation between supervised and unsupervised learning of functions[1] has been indicated by many researchers within the field of machine learning. While in the supervised case one is usually interested in the relationships between two kinds of observable variables, in the unsupervised case one is interested in the relationships between the observable data variables in a d-dimensional space and some unobservable *latent* variables in a q-dimensional space. In both cases most existing approaches are based on functions of the form

$$\mathbf{f}(\mathbf{x}; \mathbf{W}) = \mathbf{W}\mathbf{b}(\mathbf{x}), \quad \mathbf{W} \in \mathbb{R}^{d \times M} \tag{1}$$

with $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^M$ being a vector of M basis functions and $\mathbf{W}$ denoting a matrix of weights for linear combination of the basis functions. In supervised learning a large variety of different realizations of (1) has been proposed and only a few shall be mentioned here. In neural networks, multilayer-perceptrons (MLP) and radial basis function (RBF) networks are prominent examples which utilize functions of the above form. Usually, these functions involve many additional free parameters for the realization of the nonlinear basis functions, which in the RBF-case specify the location, shape and size of the basis functions and in the MLP-case may include several layers of nested functions, that is the basis functions themselves are linear combinations of nonlinear functions. In contrast to the more complex neural

---

[1]here in the sense of vector-valued multivariate functions according to general mappings from one multidimensional space to another

network basis functions recent kernel-based approaches to function learning are usually based on simple basis functions with only a very small number of free parameters (e.g. the kernel bandwidth) and incorporate all flexibility into the weights. This strategy is also followed by most unsupervised approaches to function learning and, depending on the choice of the latent variable domains and the associated basis functions, a variety of unsupervised methods, ranging from principal component analysis to vector quantization, may all be realized by flexible linear combinations of fixed basis functions [1]. In particular most approaches to Principal Curves and Surfaces [2] can be stated in terms of fitting a specific instance of (1) with fixed basis functions to multivariate data: Polygonal Lines [3] arise from piecewise linear basis functions according to degree one B-splines with the columns of $\mathbf{W}$ representing the vertices of the polygonal curve. In a similar way, Adaptive Principal Surfaces [4] can be specified using products of one-dimensional degree one splines as basis functions together with some constant function for compensation of the bias. Regularized Principal Manifolds [5] can be represented using general kernels and a constant term as basis functions. Probabilistic Principal Surfaces [6] have been proposed as an extension to the Generative Topographic Mapping [7] using a more general noise model but the same manifold parametrization according to Gaussian and constant basis functions. With some extensions to the original formulations, Self-Organizing Maps [8, 9] can be viewed as discrete approximations of principal manifolds using piecewise constant basis functions according to products of degree zero B-splines, in which case the columns of $\mathbf{W}$ represent the supporting points of the discrete approximation. This has been generalized in the Parametrized Self-Organizing Maps [10], where the piecewise constant functions were replaced by a set of orthogonal polynomials and where the supervised learning of mappings between different subsets of the observable data variables is viewed as the construction of a single data space manifold indicating the close relation between supervised and unsupervised learning of functions.

Although most approaches to principal surfaces are actually based on some realization of (1), in practice that function model implies some inherent difficulties which arise from the necessity to provide viable solutions to the *model selection* problem. First the model requires an a priori specification of the M basis functions which should somehow cover the latent space in a sufficient way. Using localized basis functions, like Gaussians or splines, the locations in latent space have to be specified. Often, a regular square grid is imposed in order to restrict the range of possible models to some tractable set of candidates. In general, this choice cannot be expected to be optimal and for higher dimensionalities of the latent space such an approach becomes impractical. In addition, certain types of basis functions also require to specify an additional smoothness parameter, e.g. the bandwidth in the Gaussian case. If the number of basis functions is not restricted, i.e. it may even be higher than the number of data points, then some regularizer for the corresponding weights has to be introduced in order to make learning feasible. In that case a sensitive regularization parameter has to be adjusted to control generalization.

For the Regularized Principal Manifolds the model has been restricted to a regular square grid for the experimental results reported in [5]. The width of the Gaussian basis functions was chosen in an ad hoc manner and the regularization parameter was adjusted manually for the reported results. For the Adaptive Principal Surfaces an iterative model refinement for successively adding new basis functions has been proposed in [4]. The stopping criterion for that scheme depends on a regularization parameter, which has been fixed at a certain

value. Although the authors tried to motivate their particular choice for that parameter as a general rule, for some of the datasets considered in the paper, actually different values were used to achieve the reported results. For the Probabilistic Principal Surfaces [6] candidate models were restricted to a regular square grid in the experiments. The width of the Gaussian basis functions and a regularization parameter were fixed in an ad hoc manner. Then the authors just compared several choices for a continuous noise model parameter and two discrete complexity parameters for the resulting projection error on hold-out data. With respect to Self-Organizing Maps only probabilistic formulations like the Generative Topographic Maps [7] have actually faced the model selection problem. In [11] a scheme for Bayesian inference with regard to the regularization parameter and the bandwidth of the Gaussian basis functions has been proposed. For the experimental results on synthetic data the authors utilize a regular square grid with an ad hoc choice of the number of basis functions and the number of latent sample points.

In this paper we introduce an approach to principal surfaces based on an alternative class of functions together with an associated learning scheme, which constitutes an unsupervised version of classical kernel regression. The resulting *unsupervised kernel regression* (UKR) approach aims at overcoming the before-mentioned shortcomings of previous approaches to principal surfaces. Thereby the focus of this paper is not on theoretically achievable learning rates[2] but on motivation and demonstration of the practical advantages of the UKR approach.

In the following section, UKR is derived as the unsupervised counterpart of the Nadaraya-Watson kernel regression estimator. After that, a general UKR learning scheme is proposed in terms of minimization of some cross-validated reconstruction error. Next, by means of the kernel trick [12], the concept of UKR is extended to general feature spaces. In the subsequent section, examples on synthetic and real-world data illustrate some of the convenient features of the UKR approach in practice. In the discussion, we focus on the relations to other approaches to nonlinear dimensionality reduction, especially on the pros and cons with regard to spectral embedding methods. In the last section we draw some final conclusions.

2   Kernel Regression on Latent Variables

The basic idea of the UKR approach is to generalize the Nadaraya-Watson kernel regression estimator [13, 14] to the unsupervised case of function learning. In the supervised case with multivariate inputs and outputs, given a sample $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ from the joint distribution of both variables, classical kernel regression is based on an estimate of the conditional density $p(\mathbf{y} \,|\, \mathbf{x}) = p(\mathbf{x}, \mathbf{y})/p(\mathbf{x})$ using kernel density estimators [15] of the joint and marginal densities, respectively. Then the corresponding conditional mean $\langle \mathbf{y}|\mathbf{x} \rangle$ yields an estimator of the regression function $\mathbf{f}(\mathbf{x})$. In order to derive an unsupervised counterpart of a given method for supervised function learning, we follow the Generalized Regression Framework [1], which simply suggests to choose the same functional form of the corresponding estimator, but to treat the missing input data as *parameters*. These parameters $\mathbf{x}_i \in \mathbb{R}^q$ serve as lower-dimensional latent representations of the data points $\mathbf{y}_i \in \mathbb{R}^d$ and in general some suitable constraints on these parameters have to be introduced in order to make learning possible. For reasons which will be indicated below we here assume the latent locations to be restricted

---

[2]the study of learning rates is nevertheless important and will be the subject of future work.

to some compact subset $\mathcal{X} \subset \mathbb{R}^q$. In the following, these latent points are the columns of the parameter matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, while the actual data points are the column vectors of the matrix $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$ according to an iid sample from the unknown data space distribution. With that choice the UKR function model takes the form

$$\mathbf{f}(\mathbf{x}; \mathbf{X}) = \sum_{i=1}^{N} \frac{K(\mathbf{x} - \mathbf{x}_i)}{\sum_{j=1}^{N} K(\mathbf{x} - \mathbf{x}_j)} \mathbf{y}_i = \mathbf{Y}\mathbf{b}(\mathbf{x}; \mathbf{X}), \tag{2}$$

where the N-vector $\mathbf{b}(\mathbf{x}; \mathbf{X})$ contains the kernel-based latent basis functions, normalized according to a unit sum of its components $b_i(\mathbf{x}; \mathbf{X})$. The density kernels $K(\mathbf{v})$ are assumed to have a continuous first derivative and may be chosen from a wide range of possible functions (see e.g. [15]). Usually, the argument of a multivariate density kernel is subject to a linear transformation by means of an invertible smoothing matrix $\mathbf{H}$. In that way the common notation $K(\mathbf{H}^{-1/2}\mathbf{u})$ implies a prior scaling of the data dimensions and obviates the incorporation of any smoothing parameters into the kernel. Here we follow this convention but we drop the smoothing matrix because the scale of the latent points themselves can be used to control the degree of smoothing.

In general, the objective of unsupervised function learning is to find a suitable realization of the mapping $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$ together with an associated latent representation [16]. Remarkably, in the UKR case that objective can be stated as the problem of finding a suitable latent mixture density

$$p(\mathbf{x}; \mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i), \tag{3}$$

which takes the form of a kernel density estimator, given the latent points as "data". For notational convenience, in the following the dependency on the parameters $\mathbf{X}$ shall be omitted from references to p. With that latent density model, the UKR-function can be completely specified without any further parameters, except for some suitable restriction of the latent domain $\mathcal{X}$. Each basis function response $b_i(\mathbf{x}; \mathbf{X})$ is then derived from the latent mixture model as the posterior probability that $\mathbf{x}$ has been generated by the i-th mixture component. As compared with function (1), the UKR-function (2) is also realized by linear combinations of nonlinear latent basis functions. However, the corresponding weight matrix is now replaced by the data matrix and all flexibility is incorporated into the N basis functions, which are realized by normalized kernels with locations as free parameters. Thereby, each of the N basis functions is associated with a data point such that the response of the i-th basis function determines the weight of $\mathbf{y}_i$ within a convex combination of data points.

In the following, we will show how the latent variable domain $\mathcal{X}$ may be restricted in order to control the complexity of the *UKR-manifold*

$$\mathcal{M} = \{\mathbf{y} \in \mathbb{R}^d \mid \mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{X}), \ \mathbf{x} \in \mathcal{X}\}, \tag{4}$$

which contains all possible images of the UKR-function. According to the UKR function model (2), $\mathbf{f}$ can be viewed as an estimator based on a weighted average of data points with the kernel defining the effective size of a latent neighborhood, which in turn determines the weights of the data points. From this formulation it is clear that for a sufficiently smooth

UKR-manifold q-dimensional points which are close in latent space are associated with d-dimensional points on the manifold which are close in data space. Therefore, each point on the UKR-manifold has to be the result of local averaging with respect to a non-vanishing latent neighborhood in order to provide generalization, i.e. a sufficient smoothness of the manifold. On the other hand, a necessary condition for generalization is that the UKR-manifold has to be restricted in terms of an explicit or implicit upper bound on its spatial extension. Otherwise, as with any other realization of principal curves or surfaces, a nonlinear manifold may interpolate the data to yield a trivial solution for any approximation criterion based on minimization of some empirical risk or reconstruction error.

For $q = 1$ the spatial extension can be measured by the curve-length

$$\int_{\mathcal{X}} \|\mathbf{Y}\mathbf{b}'(x)\| dx \ \leq \ \|\mathbf{Y}\|_2 \int_{\mathcal{X}} \|\mathbf{b}'(x)\| dx \ \leq \ \|\mathbf{Y}\|_2 \max_{x \in \mathcal{X}} \|\mathbf{b}'(x)\| \int_{\mathcal{X}} dx, \tag{5}$$

where $\mathbf{b}'(x)$ denotes the vector of all derivatives at x with components

$$b_i'(x) = \frac{K'(x - x_i)}{Np(x)} - \frac{K(x - x_i)p'(x)}{Np^2(x)}. \tag{6}$$

Thus, for compact support $\mathcal{X}$ and continuously differentiable kernels the length of the curve is upper bounded if the density is lower bounded by some positive nonzero constant within the latent domain. In a similar way the argument applies to manifolds with $q > 1$ (see Appendix 7.1).

Recently, for principal curves within an empirical risk minimization framework [3] it has been shown that an upper bound on the curve length is also *sufficient* for generalization. In [17] it has been shown that alternatively also a bounded total turn of the curve provides a sufficient condition for generalization. This formulation also includes the first principal axis of a distribution as a special case. An extended analysis which directly applies to general multidimensional manifolds has been provided in [5].

The above considerations imply that an upper bound on the spatial extension of the manifold can be realized by defining the UKR-functions only on latent regions where the density $p(\mathbf{x})$ is sufficiently high. Then, the size of the latent domain will always be bounded while at the same time the magnitude of the derivatives of the basis functions will be bounded. From (5) and (6) it follows that an upper bound on the manifold's spatial extension can be directly controlled by a lower bound on the latent density, which determines both the maximum possible magnitude of the derivatives and the size of the latent domain. Therefore, in order to control the complexity of the UKR-manifold, it makes sense to define the latent domain $\mathcal{X}$ by means of a density constraint

$$\mathcal{X}_\eta = \left\{\mathbf{x} \in \mathbb{R}^q \mid p(\mathbf{x}) \geq \eta K(\mathbf{0})\right\}, \quad 0 < \eta < 1. \tag{7}$$

As the manifold does not exist for regions of low density, the above constraint can give rise to perforated or even disconnected manifolds. Although to our knowledge such manifolds have not been considered by previous approaches to principal surface learning so far, for a good low-dimensional representation it makes sense to define the approximating manifold only in regions which are actually supported by the data space distribution, i.e. where it can be estimated from the data.

In the following three subsections, we propose a UKR learning scheme for estimation of the latent points.

2.1 Data Space Reconstruction Error

After having defined the UKR function model, some learning scheme for selecting a certain instance from the set of possible functions has to be specified. Analogous to the supervised case where the regression function minimizes the mean square error (MSE), for the unsupervised case we shall now consider the error which occurs if the data points are reconstructed from their latent representations. Given N d-dimensional data points $\mathbf{y}_i$ (the columns of $\mathbf{Y}$) which are reconstructed from N q-dimensional latent variable vectors $\mathbf{x}_i$ (the columns of $\mathbf{X}$), the reconstruction error of the data can be defined in terms of the Frobenius-norm

$$R(\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i; \mathbf{X})\|^2 = \frac{1}{N} \|\mathbf{Y} - \mathbf{Y}\mathbf{B}(\mathbf{X})\|_F^2, \tag{8}$$

where $\mathbf{B}(\mathbf{X}) = [\mathbf{b}(\mathbf{x}_1; \mathbf{X}), \ldots, \mathbf{b}(\mathbf{x}_N; \mathbf{X})]$ is the matrix of latent basis function responses. Following the arguments in [3] we restrict the set of candidate models according to UKR-manifolds with a bounded spatial extension. Using the density constraint (7) learning requires to solve the constrained optimization problem:

$$\text{minimize} \quad R(\mathbf{X}) \quad \text{subject to} \quad \forall_i \; \mathbf{x}_i \in \mathcal{X}_\eta. \tag{9}$$

For the limiting case $\eta \to 1$ the kernel centers are forced to coincide in order to reach the maximum possible overlap, which in turn lets the UKR-manifold shrink towards the sample mean of the data. Unfortunately, prior specification of the parameter η would be difficult without any knowledge on the optimal latent density. Considering η as a hyperparameter, several values have to be compared for the generalization of the resulting UKR-manifolds. As a convenient alternative, in the following we shall consider a learning objective for implicit restriction of the latent variable domain, which does not require a prespecified density threshold.

2.2 Leave-One-Out Cross-Validation

With respect to the model selection problem, a striking and very attractive feature of the UKR-approach is that it allows us to perform leave-one-out cross-validation (CV) with actually no additional computational cost. As an analogue to the supervised CV-scheme for finding suitable kernel bandwidths [15], the reconstruction error (8) may be replaced by the cross-validated version

$$R_{\text{cv}}(\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{f}_{-i}(\mathbf{x}_i; \mathbf{X})\|^2 = \frac{1}{N} \|\mathbf{Y} - \mathbf{Y}\tilde{\mathbf{B}}(\mathbf{X})\|_F^2,$$

where $\mathbf{f}_{-i}$ denotes the above linear combination of the data in (2) with the i-th data point excluded. This leave-one-out UKR CV-error can simply be realized by a slightly modified matrix of basis function activations $\tilde{\mathbf{B}}(\mathbf{X})$ where all kernel functions $K(\mathbf{0})$ have been replaced by zeros. To see the regularizing impact of the leave-one-out scheme, the CV-error can be written as a modification of the original objective function (8) with penalty functions S_i scaling the terms of the original error sum

$$R_{\text{cv}}(\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} S_i(\mathbf{X}) \|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i; \mathbf{X})\|^2 \tag{10}$$

6

with penalty factors

$$S_i(\mathbf{X}) = \left\{ 1 + \frac{K(\mathbf{0})}{\sum_{j \neq i} K(\mathbf{x}_i - \mathbf{x}_j)} \right\}^2 \propto \left\{ \frac{p(\mathbf{x}_i)}{p_{-i}(\mathbf{x}_i)} \right\}^2.$$

From this formulation it is clear that the penalty is realized by an increasing weight $S_i(\cdot)$ according to an increasing ratio between the latent density and its leave-one-out version p_{-i}. In that way, the kernel functions are forced to overlap in order to prevent the latent points from occupying regions of vanishing density. In the following we will outline an overall optimization scheme for achieving good local minima of that objective function.

2.3 Parameter Optimization

For minimization of the UKR CV-error (10) it is important to find a sufficiently deep local minimum of the highly nonconvex objective function. As with other approaches to principal surfaces, shallow local minima can be associated with arbitrarily bad manifolds which completely miss the latent structure of the data. Because general methods for global optimization, like simulated annealing, become impractical for large parameter spaces, a good initial guess for the parameters is crucial for the success of any method for learning of principal surfaces.

If the optimal surface is not too far from a linear manifold then principal component analysis (PCA) can provide a sufficiently good initialization of the parameters. However, with UKR it is straightforward to exploit any algorithm for multidimensional scaling (MDS) to yield a suitable initialization of the latent points, which are the only free parameters of the model. In particular recent spectral embedding methods [18, 19, 20] which attempt to recover the coordinates of the data with respect to some implicit nonlinear manifold model should be well-suited for an initial guess of the UKR-parameters. Given a set of latent coordinates for the data, all that is needed to specify a suitable initial UKR-function is an adequate scaling of these coordinates. The scaling corresponds to the selection of the kernel bandwidth in the case of normalized latent coordinates with a fixed scale (e.g. unit variance). Optimal scale factors are crucial for adjusting the complexity of the initial UKR model and can be achieved by minimization of the UKR CV-error.

In practice we might have to choose an optimal initialization from a set of candidates. If we are using a nonlinear spectral method like LLE [18] to find an initialization, for example, the set of candidates will result from the fact that spectral embedding methods depend on a natural neighborhood parameter that will have to be adjusted prior to using UKR. In addition, it can be valuable to include a PCA solution in the candidate set, since for example in case of very noisy data, the nonlinear spectral method might fail.

In order to select the optimal initialization, we can compare the CV-errors for several candidates and choose the one with the lowest error. If a particular nonlinear manifold model provides the best candidate solution we assume that this solution is close enough to the optimal manifold and therefore, starting with the corresponding scaled coordinates, we directly minimize the CV-error. If, however, a coarse PCA-based solution is selected among all candidate solutions, we apply a homotopy method [21] for gradually increasing the model complexity during optimization in order to prevent bad local minima. This can

be achieved by stepwise relaxation of the density constraint (7) via decreasing η until finally the unconstrained CV-error is minimized.

In the following we summarize the overall optimization scheme for learning of the UKR model:

1. *Initial candidates:* Provide a set of L normalized candidate solutions

$$\mathcal{C} = \{\bar{\mathbf{X}}_j \in \mathbb{R}^{q \times N} | \bar{\mathbf{X}}_j \bar{\mathbf{X}}_j^T = \mathbf{I}_q, \ j = 1, \ldots, L\}.$$

2. *Scale optimization:* Find optimal scale factors $\mathbf{s}_j = [s_1^{(j)}, \ldots, s_q^{(j)}]^T$ according to

$$\mathbf{s}_j = \arg\min_{\mathbf{s}} R_{\mathrm{cv}}(\mathrm{diag}(\mathbf{s})\bar{\mathbf{X}}_j)$$

and set $\mathbf{X}_j = \mathrm{diag}(\mathbf{s}_j)\bar{\mathbf{X}}_j$.

3. *Candidate selection:* As an initial guess select best matrix $\hat{\mathbf{X}} = \mathbf{X}_k$ according to

$$k = \arg\min_{j} R_{\mathrm{cv}}(\mathbf{X}_j).$$

4. *Homotopy method:* If $\hat{\mathbf{X}}$ corresponds to the scaled PCA solution then for a sequence of n decreasing density thresholds $\eta = \eta_1, \ldots, \eta_n$ solve

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_\eta} R_{\mathrm{cv}}(\mathbf{X}).$$

5. *CV-error minimization:* Starting either with step 3) or step 4) solution, compute

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} R_{\mathrm{cv}}(\mathbf{X}).$$

6. *Density threshold:* Select final density threshold according to

$$\eta = \min_{i} p(\mathbf{x}_i; \mathbf{X})/K(\mathbf{0}).$$

For gradient-based optimization in steps 4 and 5, the required matrix gradient together with some considerations concerning computational complexity of function and gradient evaluation can be found in Appendix 7.2.

With regard to the last step of the above scheme we like to stress that the choice of a suitable density threshold η should usually be driven by application specific constraints. In applications where no additional data have to be projected after training there is actually no need to specify that parameter at all. In other applications a close inspection of the latent density after training may for example suggest to adjust η according to the 90 percent contour of the density in order to reduce the influence of outliers on the boundaries of the final UKR-manifold. Therefore the above choice should only provide a default value which just ensures that all reconstructions of the training data are part of the manifold.

3 Feature Space UKR

As another convenient feature of the UKR approach the corresponding function model (2) also suggests to generalize unsupervised learning of nonlinear functions, beyond conventional finite-dimensional data spaces, to general Hilbert spaces, implied by the associated inner product of a given Mercer or *positive definite* kernel. In order to apply the kernel trick [12] to UKR learning we first have to write our algorithm in a form solely based on inner products of the data. The reconstruction error (8) therefore can be written as

$$R(\mathbf{X}) = \frac{1}{N}\|\mathbf{Y} - \mathbf{Y}\mathbf{B}(\mathbf{X})\|_F^2 = \frac{1}{N}\|\mathbf{Y}\bar{\mathbf{B}}\|_F^2 = \frac{1}{N}\mathrm{tr}(\bar{\mathbf{B}}^T\mathbf{Y}^T\mathbf{Y}\bar{\mathbf{B}}) = \frac{1}{N}\mathrm{tr}(\bar{\mathbf{B}}^T\mathbf{G}\bar{\mathbf{B}}), \qquad (11)$$

where $\bar{\mathbf{B}} \equiv \mathbf{I} - \mathbf{B}(\mathbf{X})$ and $\mathbf{G} \equiv \mathbf{Y}^T\mathbf{Y}$ is the Gram matrix of the data with entries $g_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$. Replacing the inner product $\langle \cdot, \cdot \rangle$ with a kernel function $k(\cdot, \cdot)$ yields an implicit mapping $\mathbf{\Phi}(\cdot)$ from data space into some feature space $\mathcal{F}$ with the property $\langle \mathbf{\Phi}(\mathbf{y}_i), \mathbf{\Phi}(\mathbf{y}_j) \rangle = k(\mathbf{y}_i, \mathbf{y}_j)$. Note that the kernels applicable here belong to a different class of functions than the kernels used for the UKR basis functions, which have been derived from kernel density estimators[3]. UKR as a translation invariant algorithm works not only with the commonly used positive definite or Mercer kernels, but also with the larger class of *conditionally positive definite* kernels [12].

Computation of the feature space error only requires to compute the Gram matrix of the transformed data matrix, i.e. the matrix of all inner products between feature vectors, and therefore, only the corresponding kernel matrix $\mathbf{K}(\mathbf{Y})$ with elements $k_{ij} = k(\mathbf{y}_i, \mathbf{y}_j)$ is required. Consequently, also the partial derivatives may solely be computed in terms of inner products, i.e. by using the kernel matrix:

$$\frac{\partial R(\mathbf{X})}{\partial x_{ij}} = \frac{1}{N}\frac{\partial}{\partial x_{ij}}\mathrm{tr}(\bar{\mathbf{B}}^T\mathbf{K}(\mathbf{Y})\bar{\mathbf{B}}) = -\frac{2}{N}\mathrm{tr}\left(\bar{\mathbf{B}}^T\mathbf{K}(\mathbf{Y})\frac{\partial\mathbf{B}}{\partial x_{ij}}\right), \qquad (12)$$

where the matrix $\partial\mathbf{B}/\partial x_{ij}$ contains all partial derivatives $\partial b_r(\mathbf{x}_s; \mathbf{X})/\partial x_{ij}$ of the basis functions. By using the above formalism, UKR-functions may be fitted in feature spaces in the same way as in usual data spaces. For complexity control of the feature space variant the previous cross-validation scheme based on (10) applies in the same way as in the corresponding data space formulation. For very high-dimensional data spaces $\mathbb{R}^d$ it may also make sense to consider the feature space formulation (using the standard dot product as a kernel) in order to reduce computational cost. For that purpose one can precalculate the Gram matrix of the data in order to avoid direct evaluation of the UKR-function $\mathbf{Y}\mathbf{b}(\cdot; \mathbf{X})$, which scales linearly with d.

A minor drawback of the feature space variant remains: Because the UKR-function in a kernel feature space takes the form

$$\mathbf{f}(\mathbf{x}; \mathbf{X}) = \sum_j \mathbf{\Phi}(\mathbf{y}_j) b_j(\mathbf{x}; \mathbf{X}), \qquad (13)$$

in general the points of the UKR-manifold in feature space cannot be given explicitly. Although several methods that can map feature space vectors back to data space have recently

---

[3]Therefore, a different notation is used for both kinds of kernels: $k(\mathbf{v}, \mathbf{v}')$ for the not necessarily translation invariant Mercer kernels, $K(\mathbf{v} - \mathbf{v}')$ for the density kernels.

been proposed (see [22] and [23], for example), in the general case where an exact pre-image does not exist these methods can only yield an approximation.

Fortunately, in applications where only dot products between (13) and other feature space elements are needed, an explicit computation of the mapping is not necessary. Here one may compute a linear combination of dot products, instead of a dot product with a linear combination:

$$
\begin{aligned}
\langle \Phi(\mathbf{y}), \mathbf{f}(\mathbf{x}; \mathbf{X}) \rangle &= \left\langle \Phi(\mathbf{y}), \sum_j \Phi(\mathbf{y}_j) b_j(\mathbf{x}; \mathbf{X}) \right\rangle \\
&= \sum_j \langle \Phi(\mathbf{y}), \Phi(\mathbf{y}_j) \rangle \, b_j(\mathbf{x}; \mathbf{X}) \\
&= \sum_j k(\mathbf{y}, \mathbf{y}_j) b_j(\mathbf{x}; \mathbf{X}).
\end{aligned}
\tag{14}
$$

This can for instance be utilized to visualize the classification boundary of a kernel-based support vector machine. In that case the additional kernel parameters may be selected in a supervised manner. But also in a completely unsupervised setting feature space UKR may be useful, and in the following we propose a special kernel for realization of an alternative metrics.

If the Euclidean norm does not provide an adequate distance metrics for the data at hand, it can be beneficial to measure L_1-distances instead. UKR is explicitly based on the Euclidean norm, but by means of the kernel trick one can nonetheless effectively work with distances based on the L_1-norm. Defining a kernel function

$$
k(\mathbf{y}, \mathbf{y}') = \frac{1}{2} \left(\|\mathbf{y}\|_1 + \|\mathbf{y}'\|_1 - \|\mathbf{y} - \mathbf{y}'\|_1 \right)
\tag{15}
$$

yields a positive definite kernel matrix $\mathbf{K}(\mathbf{Y}) = (k(\mathbf{y}_i, \mathbf{y}_j))$ and thus implies existence of a mapping $\Phi(\mathbf{y})$ into some feature space $\mathcal{F}$ (see Appendix 7.3 for a derivation). The Euclidean distance between two feature space images can be expressed by

$$
\begin{aligned}
\|\Phi(\mathbf{y}) - \Phi(\mathbf{y}')\|_2^2 &= \langle \Phi(\mathbf{y}) - \Phi(\mathbf{y}'), \Phi(\mathbf{y}) - \Phi(\mathbf{y}') \rangle \\
&= k(\mathbf{y}, \mathbf{y}) - 2k(\mathbf{y}, \mathbf{y}') + k(\mathbf{y}', \mathbf{y}') \\
&= \|\mathbf{y} - \mathbf{y}'\|_1 .
\end{aligned}
\tag{16}
$$

In that way the derivation of the above L_1-kernel corresponds to the definition of kernels on the basis of loss functions [24].

4 Experiments

We applied UKR on synthetic and real-world datasets in order to illustrate the proposed learning scheme of section 2.3 on practical examples. As indicated in 2.3, the proposed UKR optimization scheme can easily incorporate spectral embedding methods for initialization of the UKR parameters. For the exemplary results presented here, we utilized the Locally Linear Embedding algorithm [18]. In addition, we used an efficient shortcut for finding optimal

scalings that are necessary to turn the normalized eigenvector solutions into good UKR model initializations: We found that for this purpose good initial scalings can be obtained from kernel bandwidth selection methods for density estimation. In our experiments we utilized likelihood cross-validation (LCV, see e.g. [25]) to select suitable bandwidths, i.e. to adjust the initial scale of the latent coordinates. Subsequent optimization of the scale of the latent variables with the objective to minimize the CV-error (10) was performed by a direct search algorithm [26] which may easily be replaced by more sophisticated methods for unconstrained nonlinear optimization.

4.1 "Noisy Spiral" data

To demonstrate the complete UKR learning scheme on a non-trivial toy example we fitted an UKR curve to a sample ($N = 300$) of the "noisy spiral" distribution depicted in figure 1. The data was created by adding Gaussian noise of standard deviation $\sigma = 0.05$ in both dimensions to a sample of a spiral with two whorls, its radius ranging from 0.2 to 1.2.

For initialization (cf. step 1 in section 2.3) we applied the LLE algorithm based on K nearest neighbors. In this experiment, we varied K from 4 to 14 neighbors, giving rise to 11 different candidate sets of latent coordinates. As a twelfth candidate, we used a PCA solution. For each candidate, the optimal scaling was determined as described above, according to step 2 of our optimization scheme.

Fig. 1 shows the resulting initializations and their corresponding CV-errors (10). For each of the twelve cases we have visualized the initial UKR-manifold by evaluating the UKR function (2) on 1000 regularly spaced points between the smallest and the largest latent coordinate.

As described in step 3 of section 2.3, we selected the candidate model with the lowest CV-error, which in this case was the scaled LLE solution of neighborhood size $K = 10$. For the fine tuning (step 5) of the UKR model, we further minimized the CV-error (10), this time varying not only the scale, but the whole set of latent parameters. For this task, we used the RPROP algorithm [27].

Fig. 2 shows how the UKR model is affected by the final gradient-based optimization. Note that the CV-error as well as the visualized manifold practically remains almost unchanged between 500 and 1000 RPROP steps.

From the UKR curves in Fig. 2 the reader might argue that the method already tends to over-fitting in this example. This is actually not the case as can be seen from estimation of the expected reconstruction error for the different stages of the optimization. For that purpose we projected 3000 test data points sampled from the same "noisy spiral" distribution onto the UKR manifold in the different stages. For these projections we initialized the minimization problem

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{f}(\mathbf{x}; \mathbf{X})\| \quad \text{s.t.} \quad p(\mathbf{x}) \geq \eta K(\mathbf{0}) \tag{17}$$

with the "nearest reconstruction" among the training data. That is, we chose that $\mathbf{x}_i$ among the 300 latent locations as an initial value which yields the nearest point on the manifold with respect to the given test point. The optimization itself was carried out by the RPROP algorithm in combination with a barrier function to realize the constraint. The density threshold η used to restrict the latent domain was selected by the heuristics described by
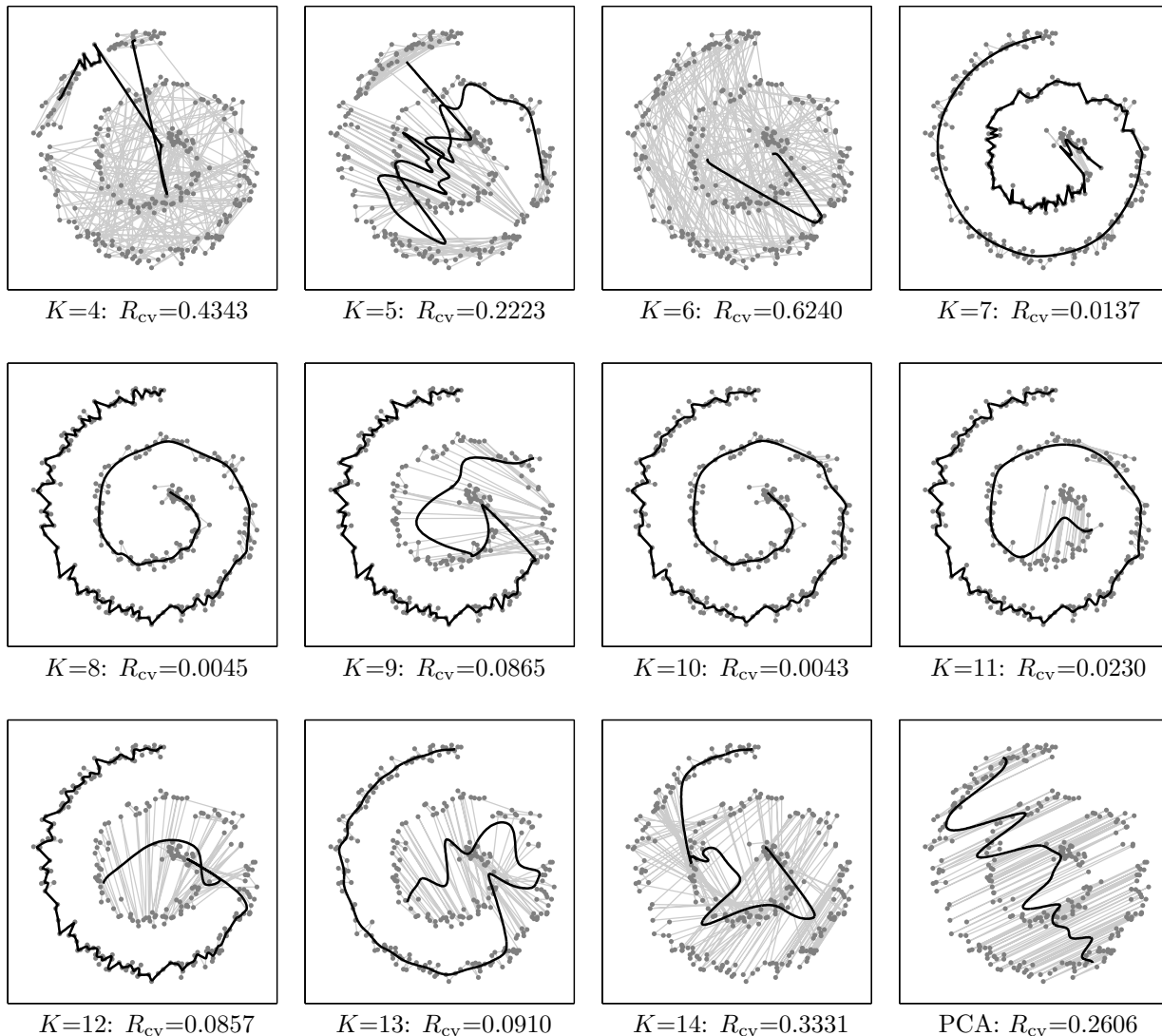
Figure 1: Fitting UKR curves to a "noisy spiral" sample; Below each plot, the size of the LLE neighborhood (K) is given, as well as the CV-error after initialization. The plot in the lower right corner shows the manifold retrieved by PCA initialization. The black curve depicts the UKR manifold whereas the sample data is shown as dark gray dots. The initial solution (as given by LLE or PCA) is plotted in light gray. Here we just connected the data points in the order LLE/PCA places them in latent space, which corresponds to an infinite scale of the latent variable. Note that the visually best solutions ($K = 8, 10$) correspond to the smallest CV-errors.

step 6 in section 2.3. Table 1 shows the resulting errors for the different stages and clearly shows that both training (CV) *and* test error are decreasing for the more complex curves.
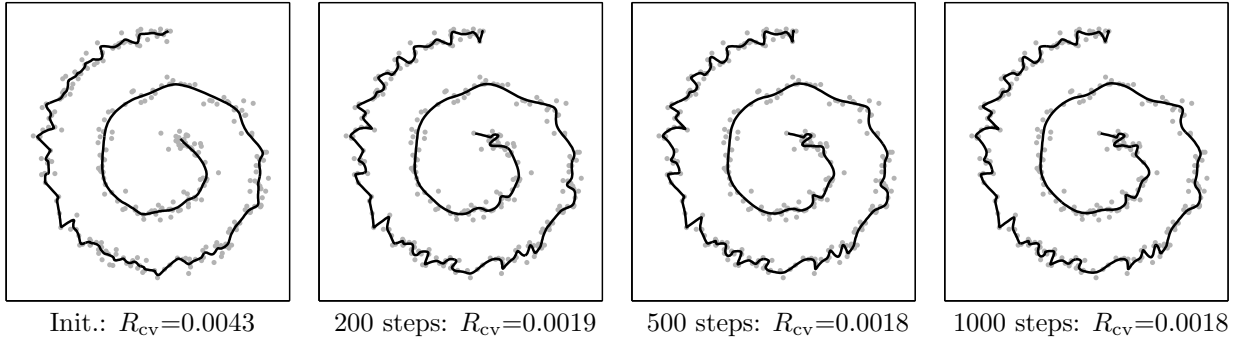
Init.: R_{cv}=0.0043 200 steps: R_{cv}=0.0019 500 steps: R_{cv}=0.0018 1000 steps: R_{cv}=0.0018

Figure 2: Gradient-based optimization of the UKR curve on "noisy spiral" sample; The shape of the UKR manifold and the corresponding CV-error is depicted after initialization based on the best scaled LLE solution (K=10) as well as after 200, 500 and 1000 RPROP steps.

No. of steps	initialization	200	500	1000
CV-error	0.00429	0.00189	0.00182	0.00178
Proj. error	0.00287	0.00244	0.00237	0.00232

Table 1: Mean error for projecting 3000 test points onto the "noisy spiral" UKR manifold

4.2   Handwritten Digits

We applied UKR to the problem of learning suitable manifold representations for handwritten digit data. In this experiment we used the USPS digits dataset (see e.g. [12]), which contains a total of 7291 grayscale images of size 16x16 divided into 10 classes. For visualization purposes, we fitted a 2-D manifold to the USPS subset representing the digit "2", which left us at 731 data vectors with 256 dimensions.

To demonstrate UKR's flexibility with respect to the choice of the latent kernel functions and to show how a finite support kernel can improve computational efficiency, we performed this experiment with both the Gaussian kernel and the Quartic kernel. The latter corresponds (up to some constant scaling) to the square of the multivariate Epanechnikov kernel [15], which due to squaring becomes differentiable:

$$K_Q(\mathbf{x}_r - \mathbf{x}_s) \propto [1 - \|\mathbf{x}_r - \mathbf{x}_s\|^2]_+^2. \tag{18}$$

The resulting sparse structure of $\mathbf{B}(\mathbf{X})$ can lead to a significant performance gain with respect to evaluation of the error function and its gradient and is therefore particularly useful in combination with large datasets (see Appendix 7.2).

For fitting the model, we followed the parameter optimization scheme described in section 2.3. That is, we first calculated the PCA solution and several LLE solutions for a broad range of neighborhood sizes ($K = 2, 3, \ldots, 21$). Next, as in our "noisy spiral" experiment, we optimized the scale of these initialization candidates by first performing LCV-based bandwidth selection and then applying a direct search method for fine tuning.

As before, in order to select the best candidate among our set of scaled LLE and PCA solutions, we calculated the CV-error (10) for each candidate. Fig. 3 shows these errors for both the Quartic and the Gaussian kernels. Note that both error curves are close and that

13

the corresponding UKR models share the LLE solution with $K = 12$ nearest neighbors as the best initialization candidate (of course, each uses its own scale).
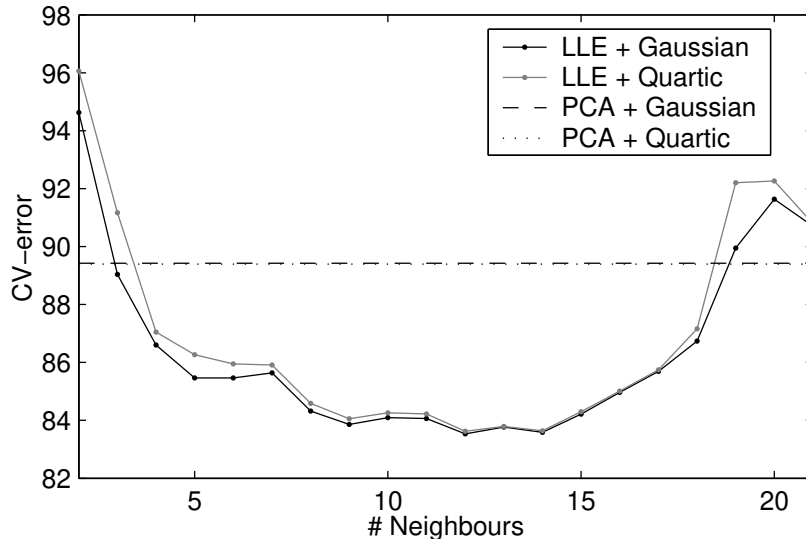

Figure 3: CV-errors for the different LLE and PCA solutions

As a last training step, we performed 500 RPROP steps minimizing the CV-error (10). At this stage, the error dropped from 83.53 to 50.90 for the Gaussian-based UKR model and from 83.62 to 51.52 for the model using the Quartic kernel in latent space.

Fig. 4(a,b) shows the resulting latent coordinates and the corresponding density. To create Fig. 4(c,d), we sampled the latent space on a regular grid and evaluated the regression function (2) as well as the latent density on every node. We displayed the grayscale images corresponding to the UKR function value and additionally scaled their intensity by the latent density at the specific grid coordinates.

Note that it is possible to assign meaning to the axes in the sampled manifold plots: From left to right, the digits loose their lower left bow and look more and more like a "Z". From top to bottom, the digits get thinner and flatter.

While fitting UKR models with both the Gaussian and the Quartic kernel led to similar results with respect to CV-errors and visualization, the difference in terms of CPU time is remarkable. We measured the time needed to perform 500 RPROP steps on a Linux PC with a Pentium IV processor running at 1.8 GHz and 1 GB RAM. Since the number of dimensions (256) is nearly half as big as the number of data (731), we performed this test using both normal data space UKR and feature space UKR with the trivial dot product kernel (cf. Appendix 7.2). The resulting CPU times (averaged across two very close test runs) can be found in Table 2.

Method \ Kernel	Gaussian	Quartic
Data Space UKR	753 s	30 s
Feature Space UKR	1085 s	25 s

Table 2: CPU time for 500 RPROP steps on the digit "2" dataset

(a) Latent coord. and density (Gaussian kernel)   (b) Latent coord. and density (Quartic kernel)

(c) Sampled manifold (Gaussian kernel)   (d) Sampled manifold (Quartic kernel)
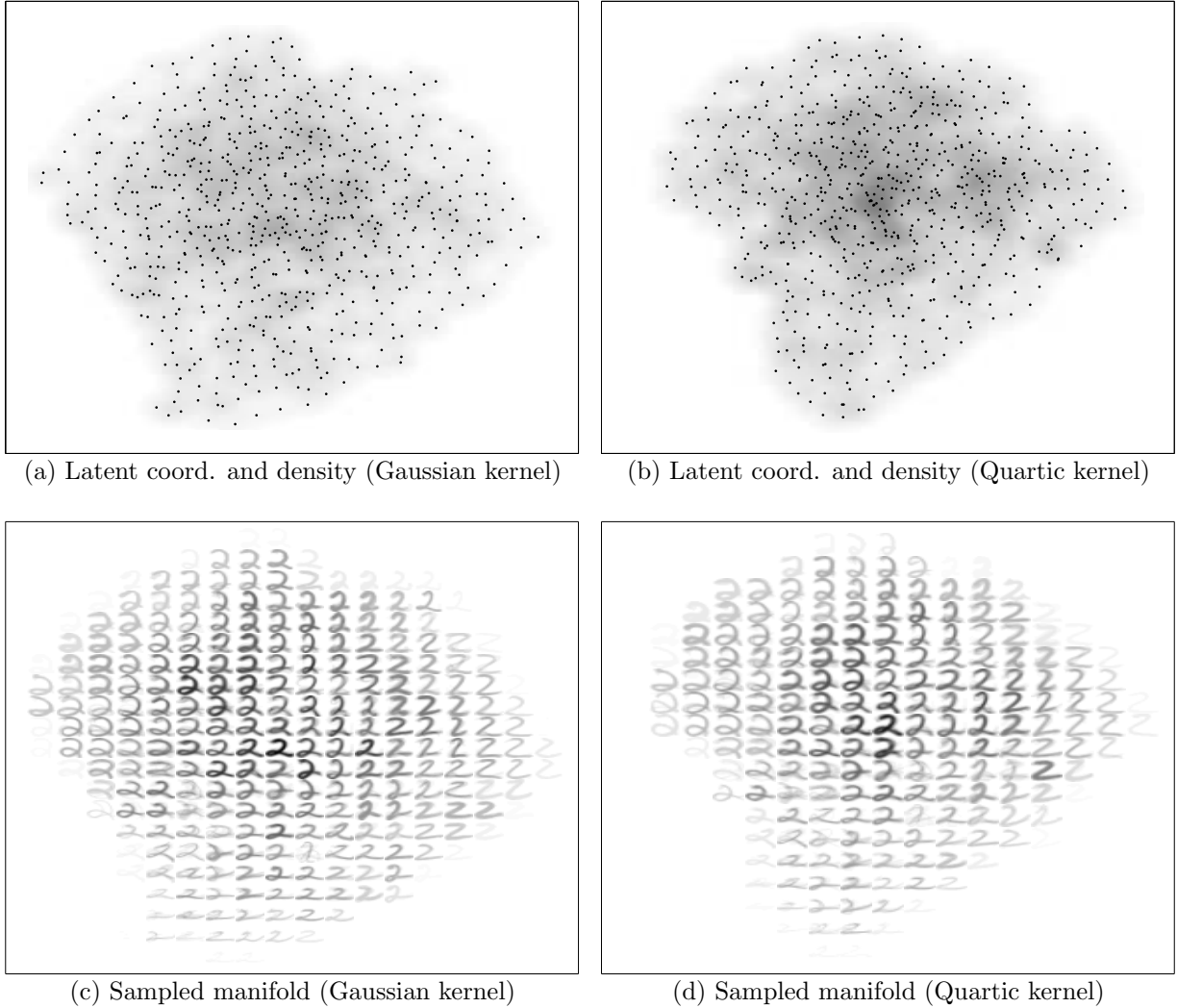
Figure 4: Latent coordinates, densities and sampled manifolds of both the Gaussian and Quartic kernel based UKR model

4.3   "Oil Flow" Data

For our third experiment we used the "oil flow" data set [7], which has also been used in [5]. The data set consists of 12-dimensional (labelled) observations, each taking on one of three possible labels that indicate different geometric configurations inside an oil pipeline. It is subdivided into a training set, a validation set and a test set containing 1000 examples each.

Since the data set is fairly clustered (see Fig. 5(a)), LLE performs poorly. For LLE to return reasonable results, one has to make sure that the neighborhood graph associated with respect to a neighborhood value K is connected. For the "oil flow" dataset this is the case for $K \geq 46$. Within neighborhoods of that size the real local properties of the data are mostly hidden, so one would hardly expect to retrieve a good mapping. As a result, all LLE solutions that we tested (we used $K = 46, 51, 56 \ldots 101$) led to substantially larger CV-errors (10) than a PCA solution.

We therefore chose the PCA solution as an initialization. According to step 4 of the

optimization scheme (section 2.3), we used a homotopy based approach for optimization in order to avoid poor local minima. Starting from the PCA solution scaled down to a total variance of 0.01, we carried out 7 homotopy steps corresponding to density constraints with $\eta = 0.5, 0.25, 0.1, 0.05, 0.025, 0.01, 0.005$. For each homotopy step, we performed 100 RPROP steps in combination with a barrier function to realize the constraints. After the homotopy steps, we performed further 300 RPROP steps minimizing the unconstrained CV-error (10). Note that the model selection problem is again tackled solely by minimization of the UKR CV-error. Although the η-values may affect which local minimum is found, they have no (at least no conceptual) influence on the model complexity.

In this experiment, we again used both the Gaussian kernel and the Quartic kernel in latent space, but here we present only the results of our Gaussian-based UKR model, since it led to slightly better results in terms of the final CV-error. Note that the gain in efficiency we get by using the Quartic kernel is not that big for the homotopy-based training anyway. This is due to the fact that in early homotopy steps the density constraint prevents a high sparseness of the matrix of basis functions $\mathbf{B}(\mathbf{X})$.

To demonstrate UKR's ability to work with a different metrics, we fitted another manifold to the training set: In addition to the above data space UKR we now performed feature space UKR using the L_1-kernel (15) proposed in section 3. In this case, we replaced PCA by Kernel PCA (KPCA) for initialization.

In a follow-up experiment, we exploited the manifold structure by mapping the "oil flow" test set onto the trained manifolds. Here, we followed the strategy already described in our "noisy spiral" experiment. That is, we constrained the latent domain according to our density threshold heuristics and again performed the test set projections by the RPROP algorithm in combination with a barrier function.

As an objective measure for the preservation of structure that the different methods give rise to, we then considered classification performance in the latent space, now using the labels of the data. For this end we used a kernel density classifier (KDC), as suggested by UKR's built in latent space density model. For each test case projection we measured the class specific densities and assigned the test case to the class with the highest density. Table 3 shows the resulting error rates in comparison with the rates obtained by applying the kernel density classifier on the PCA and the KPCA solution. For these solutions we determined suitable kernel bandwidths using LCV. For comparison, we also included the rates obtained by using nearest neighbor (NN) classification in the 2D latent space.

Method \ Kernel	L_2	L_1
UKR + KDC	0.9%	0.6%
UKR + NN	1.2%	0.9%
PCA/KPCA + KDC	12.0%	16.6%
PCA/KPCA + NN	15.2%	17.7%

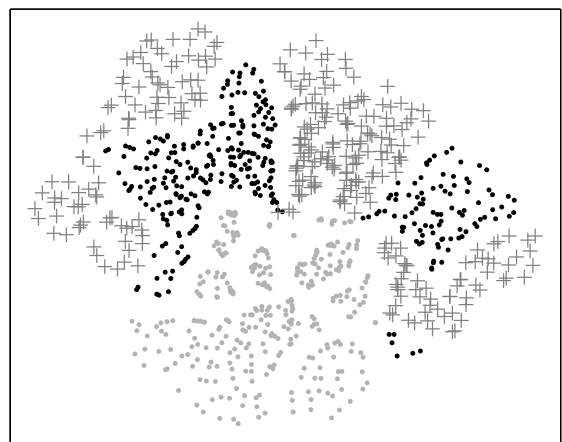Table 3: Comparison of classification performance (error rate) on the "oil flow" test data.
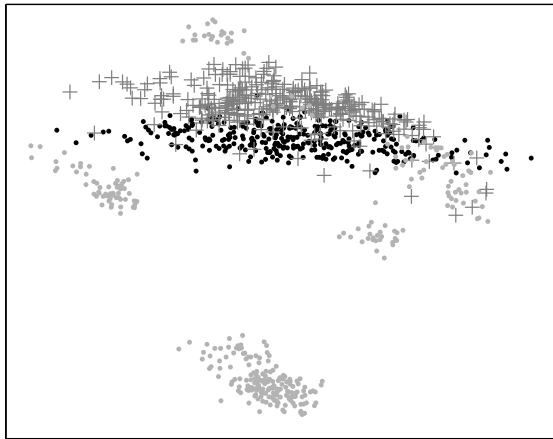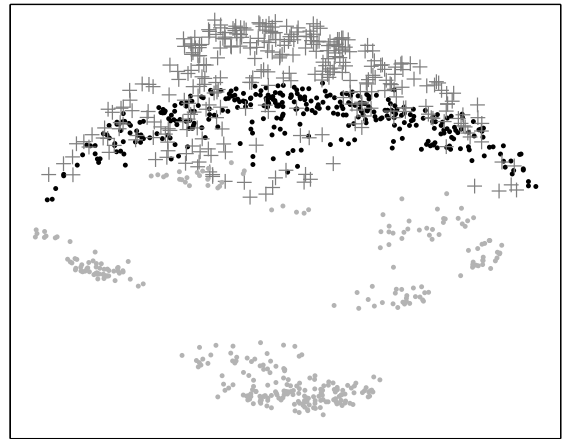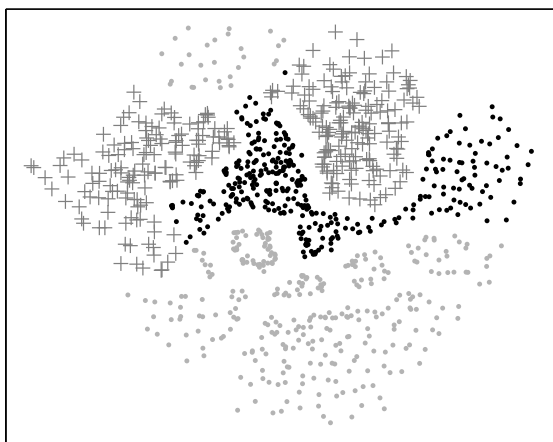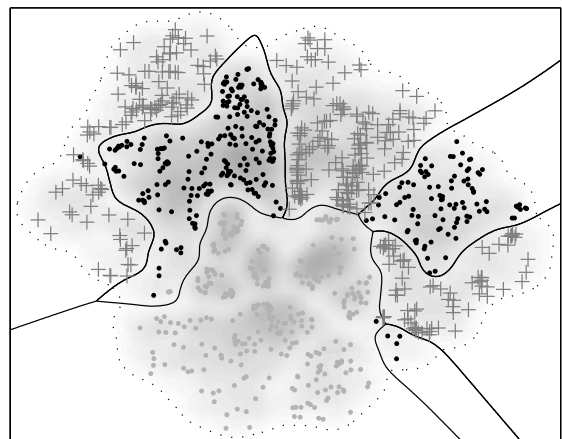
Figure 5: Latent coordinates of "oil flow" dataset retrieved by (K)PCA and (feature space) UKR. (e) and (f) show the resulting latent densities, the class borders based on kernel density classifiers as well as the test data projections.

Figures 5(a-d) show the PCA resp. Kernel PCA as well as the UKR solutions for the latent variables. The data points are depicted by different symbols corresponding to their class membership. Note that UKR separates the classes much better than (K)PCA and that the different metrics have an effect on which classes are contiguous. Figures 5(e,f) show the total density, the class borders and the density threshold contour $p(\mathbf{x}) = \eta K(\mathbf{0})$ as well as the test data projections depicted by their corresponding class symbols.

5 Discussion

Considering recent *spectral embedding methods* for nonlinear dimensionality reduction [18, 19, 20] one may raise the question whether algorithms for learning of principal surfaces are actually necessary in order to derive a suitable low-dimensional representation for high-dimensional data. As compared with classical methods for nonlinear dimensionality reduction, which rely on local minimization of highly nonconvex objective functions (see e.g. [28]), spectral methods provide a favourable scheme for global optimization, based on computation of some N-dimensional eigenvectors. Unfortunately, for nonlinear dimensionality reduction these methods require the specification of some essential complexity parameter, which usually adjusts the effective size of a data space neighborhood. Although the resulting representations are rather sensitive with respect to this complexity parameter, no common rule is known how to adjust this parameter. Consequently, all available implementations lack an automatic neighborhood selection procedure and therefore the user is restricted to the unsatisfactory practice of adjusting this parameter "by eye".

Another shortcoming of the above spectral methods concerns the projection of new data which is impossible due to the lack of an explicit manifold model which would make it possible to measure the distance of data points with respect to the low-dimensional surface. For the same reason it is not possible to sample the manifold, i.e. to compute the data space images of latent points.

Although Kernel PCA [29] has not been proposed for learning of descriptive low-dimensional representations [5], in comparison with the above spectral methods for nonlinear dimensionality reduction, a linear manifold in feature space provides an explicit manifold representation in this case. Therefore the projection indices, i.e. the latent coordinates of new data points can be computed with ease, but again, it is not generally possible to sample the manifold due to the lack of an exact mapping from latent space to data space. This shortcoming is also faced with feature space UKR and has been discussed in section 3. Although the feature space distance with respect to the linear manifold can be measured, it is not clear whether it can be used for model selection with regard to the kernel parameters. To our knowledge, in practice no unsupervised model selection has been performed for Kernel PCA so far. The close relations between the above spectral methods and Kernel PCA have recently been indicated in [30].

Summarizing, we argue that spectral methods for nonlinear dimensionality reduction do not provide a general alternative to learning of principal surfaces. Due to the lack of an explicit data space representation of the manifold, as compared with principal surfaces, the application scope of these methods is restricted. In addition, these methods do not provide solutions to the inherent model selection problem which requires to adjust the complexity of

the underlying mapping to latent space. However, regarding the inherent parameter initialization problem of principal surfaces, as shown in section 4, spectral embedding methods in combination with UKR can provide a valuable alternative to classical initializations based on linear PCA.

6   Conclusion

Unsupervised Kernel Regression (UKR) has been proposed as an alternative approach to learning of *principal surfaces* which remedies some of the inherent difficulties of previous approaches within that field. As compared with previous approaches to principal surfaces, UKR offers a practical way to cope with the model selection problem, since it allows for leave-one-out cross-validation without any additional computational cost. Thereby, the UKR parametrization obviates the necessity of an a priori specification of a set of basis functions which has to cover the latent space in a suitable way. Therefore, also higher-dimensional latent spaces can easily be realized, provided there are enough data. Although previous approaches to principal surfaces always assumed very simple topologies of the manifold, the variability of UKR consequently meets the requirements of a good approximation of a general data space distribution.

Regarding optimization issues, it is straightforward to initialize UKR parameters by means of nonlinear spectral methods. The incorporation of an initial set of latent coordinates is easily possible, because the latent points themselves are the only parameters of the UKR model. In this way, UKR can even be used to select the best initial set of latent coordinates among several candidate sets which may arise from different choices for the hyperparameter of the utilized spectral method.

Finally, UKR offers a new perspective on principal surfaces to cope with non-Euclidean data spaces. By means of the kernel trick principal surfaces may now be fitted in general feature spaces according to a given kernel for computing the corresponding inner product. In that way UKR allows us to utilize string-, tree- or even specialized engineering kernels for nonlinear projection of text documents, genomic sequences and other data which are difficult to represent within traditional vector spaces.

7   Appendix

7.1   Spatial extension of UKR-manifold

To generalize our argument from section 2 to the case $q > 1$, we have to show that the manifold's area or volume is bounded. For this end, recall that an infinitesimal volume element $dx_1 dx_2 \ldots dx_q$ at a point $\mathbf{x}$ in latent space transforms into a hyperparallelepiped at $\mathbf{f}(\mathbf{x}; \mathbf{X})$ in data space, spanned by the vectors $\mathbf{J}_1 dx_1, \mathbf{J}_2 dx_2 \ldots \mathbf{J}_q dx_q$. Here, $\mathbf{J}_i$ denotes the i-th column of the Jacobian of $\mathbf{f}(\mathbf{x}; \mathbf{X})$. Since the volume of a hyperparallelepiped is less or equal to the volume of a hyperrectangle of corresponding side lengths, and since the norm of each $\mathbf{J}_i$ is less or equal to the largest singular value (and thus, the matrix norm) of the complete Jacobian $\mathbf{J}_\mathbf{f}$, the volume of our infinitesimal hyperparallelepiped is bounded by $\|\mathbf{J}_\mathbf{f}\|_2^q dx_1 dx_2 \ldots dx_q$.

We can use this inequality to estimate the complete manifold's volume V by integration over the latent domain:

$$V \leq \int_{\mathcal{X}} \|\mathbf{J_f}(\mathbf{x}; \mathbf{X})\|_2^q dx_1 dx_2 \ldots dx_q \leq \|\mathbf{Y}\|_2^q \int_{\mathcal{X}} \|\mathbf{J_b}(\mathbf{x}; \mathbf{X})\|_2^q dx_1 dx_2 \ldots dx_q. \tag{19}$$

The components of $\mathbf{J_b}(\mathbf{x}; \mathbf{X})$, the Jacobian of $\mathbf{b}(\mathbf{x}; \mathbf{X})$, are given by

$$(\mathbf{J_b})_{ij} = \frac{\nabla_j K(\mathbf{x} - \mathbf{x}_i)}{N p(\mathbf{x})} - \frac{K(\mathbf{x} - \mathbf{x}_i) \nabla_j p(\mathbf{x})}{N p^2(\mathbf{x})} \tag{20}$$

and are finite for continuously differentiable kernels and non-vanishing density. Thus, the matrix norm $\|\mathbf{J_b}(\cdot)\|_2 \leq \|\mathbf{J_b}(\cdot)\|_F$ is bounded. For compact support $\mathcal{X}$, this implies that the manifold's volume (or area for $q = 2$) is bounded, too.

7.2   UKR Gradient

With the matrix $\mathbf{B}$ of basis function activations according to the elements

$$b_{rs} = \frac{K(\mathbf{x}_r - \mathbf{x}_s)}{\sum_{t=1}^{N} K(\mathbf{x}_t - \mathbf{x}_s)} \tag{21}$$

the partial derivatives of the UKR reconstruction error (8) w.r.t. the latent variable parameters have the general form

$$\frac{\partial R(\mathbf{X})}{\partial x_{ij}} = \frac{1}{N} \sum_{r,s} \frac{\partial \mathrm{tr}([\mathbf{B} - \mathbf{I}]^T \mathbf{Y}^T \mathbf{Y} [\mathbf{B} - \mathbf{I}])}{\partial b_{rs}} \cdot \frac{\partial b_{rs}}{\partial x_{ij}} = \frac{2}{N} \sum_{r,s} m_{rs} \frac{\partial b_{rs}}{\partial x_{ij}} \tag{22}$$

with elements m_{rs} of matrix $\mathbf{M} \equiv \mathbf{Y}^T \mathbf{Y} [\mathbf{B} - \mathbf{I}]$ where the Gram matrix $\mathbf{Y}^T \mathbf{Y}$ may be replaced by a general kernel matrix $\mathbf{K}(\mathbf{Y})$ (see 3).

In order to give a more specific form of the partial derivatives (22) one has to make a choice concerning the kernel functions in latent space (which have been derived from a specific kernel density estimator). An important class of kernels, which will be considered in the following, arises from functions of the squared Euclidean norm:

$$K(\mathbf{x}_r - \mathbf{x}_s) = f(\|\mathbf{x}_r - \mathbf{x}_s\|_2^2) \tag{23}$$

with partial derivatives given by

$$\frac{\partial K(\mathbf{x}_r - \mathbf{x}_s)}{\partial x_{ij}} = 2 f'(\|\mathbf{x}_r - \mathbf{x}_s\|_2^2)(x_{ir} - x_{is})(\delta_{rj} - \delta_{sj}). \tag{24}$$

Defining matrix $\mathbf{P}$ with elements

$$p_{rs} = \frac{-2 f'(\|\mathbf{x}_r - \mathbf{x}_s\|_2^2)}{\sum_{t=1}^{N} K(\mathbf{x}_t - \mathbf{x}_s)} \tag{25}$$

the partial derivatives of the basis function responses can be written

$$\frac{\partial b_{rs}}{\partial x_{ij}} = p_{rs}[\delta_{rj}x_{is} - \delta_{rj}x_{ir} - \delta_{sj}x_{is} + \delta_{sj}x_{ir}]$$

$$-b_{rs}\left[\delta_{sj}\sum_t p_{ts}(x_{it} - x_{is}) - p_{js}(x_{ij} - x_{is})\right]. \tag{26}$$

Insertion of the partial derivatives into (22) and rearranging the terms yields

$$\frac{\partial R(\mathbf{X})}{\partial x_{ij}} = \frac{2}{N}\sum_r (x_{ir} - x_{ij})(q_{jr} + q_{rj}), \tag{27}$$

where we used the elements

$$q_{jr} = p_{jr}m_{jr} - p_{jr}\sum_s m_{sr}b_{sr} \tag{28}$$

of an $N \times N$-matrix $\mathbf{Q}$. Using the Schur product $\mathbf{A} = \mathbf{B} * \mathbf{C}$ with $a_{ij} = b_{ij}c_{ij}$ and $\mathbf{1}$ as a vector of N ones, (27) and (28) can be written in matrix form as

$$\mathbf{Q} = \mathbf{P} * [\mathbf{M} - \mathbf{1}\mathbf{1}^T(\mathbf{M} * \mathbf{B})] \tag{29}$$

$$\nabla_{\mathbf{X}}R = \frac{2}{N}\mathbf{X}\left\{\mathbf{Q} + \mathbf{Q}^T - \text{diag}(\mathbf{1}^T[\mathbf{Q} + \mathbf{Q}^T])\right\}. \tag{30}$$

In this paper we consider Gaussian and Quartic kernels given by

$$K_G(\mathbf{x}_r - \mathbf{x}_s) \propto \exp(-\frac{1}{2}\|\mathbf{x}_r - \mathbf{x}_s\|_2^2) \tag{31}$$

$$K_Q(\mathbf{x}_r - \mathbf{x}_s) \propto [1 - \|\mathbf{x}_r - \mathbf{x}_s\|_2^2]_+^2, \tag{32}$$

which imply specific realizations of the $\mathbf{P}$-matrix:

$$p_{rs}^{\text{gauss}} = \frac{K_G(\mathbf{x}_r - \mathbf{x}_s)}{\sum_t K_G(\mathbf{x}_t - \mathbf{x}_s)} \tag{33}$$

$$p_{rs}^{\text{quart}} = 4 \cdot \frac{\sqrt{K_Q(\mathbf{x}_r - \mathbf{x}_s)}}{\sum_t K_Q(\mathbf{x}_t - \mathbf{x}_s)}. \tag{34}$$

Note, that for Gaussian Kernels $\mathbf{P}$ equals the matrix of basis function responses $\mathbf{B}$.

As already mentioned in section 2.2, the CV-error (10) can be calculated by a slight modification of the matrix of basis functions. This also applies to computing the gradient, where all occurences of both $K(\mathbf{0})$ and $f'(\mathbf{0})$ have to be replaced by zeros.

Most of the computational effort to perform a single gradient evaluation goes into calculating the matrix $\mathbf{M} = \mathbf{Y}^T\mathbf{Y}[\mathbf{B} - \mathbf{I}]$ or, in case of feature space UKR, $\mathbf{M} = \mathbf{K}(\mathbf{Y})[\mathbf{B} - \mathbf{I}]$. When using a latent kernel of infinite support (e.g. the Gaussian kernel), this operation has complexity $O(dN^2)$ for using the $d \times N$ data matrix or $O(N^3)$ in case of feature space UKR. For $2d > N$, it is therefore cheaper to perform feature space UKR with the precalculated kernel (feature) matrix $\mathbf{K}(\mathbf{Y}) = \mathbf{Y}^T\mathbf{Y}$.

If a latent kernel of finite support is utilized (e.g. the Quartic kernel), the resulting matrices $\mathbf{B}$ and $\mathbf{P}$ are sparse. This can be directly exploited to speed up the calculation of $\mathbf{M}$ by using sparse BLAS routines. Because the matrix $\mathbf{M}$ only appears within Schur products together with sparse matrices (29), as a second speed-up possibility $\mathbf{M}$ does not even have to be computed completely. For a $d \times N$ data matrix, computing the necessary elements of $\mathbf{M}$ is only of complexity $O(dN_{nz})$ where N_{nz} is the number of non-zero elements in $\mathbf{B}$.

Since feature space UKR does not involve an intermediate multiplication with the data matrix $\mathbf{Y}$, it profits even more from $\mathbf{B}$'s sparseness. Here, the computational complexity is $O(\sum_i (n_{nz}^i)^2)$ where n_{nz}^i is the number of non-zero elements in the i-th column of $\mathbf{B}$. For this reason, fitting of Quartic-based UKR models can be accelerated by using trivial dot product feature space UKR already for values of d substantially smaller than $\frac{N}{2}$.

7.3 Positive definiteness of L_1-kernel

In section 3 we introduced a kernel function that maps input space L_1-distances to L_2-distances in feature space.

To show that the kernel we used is indeed positive definite (pd), and thus a mapping $\Phi(\cdot)$ exists, we rely on theorems stated in [12]. First, it is known that $k_2(\mathbf{y}, \mathbf{y}') = -\|\mathbf{y} - \mathbf{y}'\|_2^\beta$ is conditionally positive definite (cpd) for all values $0 \leq \beta \leq 2$. Now, considering the special case of 1-D data and $\beta = 1$, one gets $k_1(y, y') = -|y - y'|$ as a cpd kernel. A sum of cpd kernels is again cpd, so $k_1(\mathbf{y}, \mathbf{y}') = -\|\mathbf{y} - \mathbf{y}'\|_1 = \sum_i -|y_i - y_i'|$ is cpd. Finally, one can construct a pd kernel $k(\cdot, \cdot)$ from a cpd kernel $\tilde{k}(\cdot, \cdot)$ in the following way :

$$ k(\mathbf{y}, \mathbf{y}') = \frac{1}{2} \left(\tilde{k}(\mathbf{y}, \mathbf{y}') - \tilde{k}(\mathbf{y}, \mathbf{y}_0) - \tilde{k}(\mathbf{y}_0, \mathbf{y}') + \tilde{k}(\mathbf{y}_0, \mathbf{y}_0) \right). $$

Setting $\mathbf{y}_0 = \mathbf{0}$ and $\tilde{k} = k_1$ immediately yields that $k(\mathbf{y}, \mathbf{y}') = \frac{1}{2}(\|\mathbf{y}\|_1 + \|\mathbf{y}'\|_1 - \|\mathbf{y} - \mathbf{y}'\|_1)$ is positive definite. As mentioned before UKR also works with cpd kernels, so the last step was not really necessary.

References

[1] P. Meinicke, "Unsupervised learning in a generalized regression framework," Ph.D. dissertation, Universität Bielefeld, 2000.

[2] T. Hastie, "Principal curves and surfaces," Ph.D. dissertation, Stanford University, 1984.

[3] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger, "Learning and design of principal curves," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 3, pp. 281–297, 2000.

[4] M. LeBlanc and R. Tibshirani, "Adaptive principal surfaces," *Journal of the American Statistical Association*, vol. 89, pp. 53–64, 1994.

[5] A. J. Smola, S. Mika, B. Schölkopf, and R. C. Williamson, "Regularized principal manifolds," *Journal of Machine Learning Research*, vol. 1, pp. 179–209, 2001.

[6] K. Chang and J. Ghosh, "A unified model for probabilistic principal surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 22–41, 2001.

[7] C. M. Bishop, M. Svensen, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, pp. 215–234, 1998.

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

[9] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-organizing Maps.* Addison Wesley, 1992.

[10] J. Walter and H. Ritter, "Rapid learning with parametrized self-organizing maps," *Neurocomputing*, vol. 12, pp. 131–153, 1996.

[11] C. M. Bishop, M. Svensén, and C. K. I. Williams, "Developments of the generative topographic mapping," *Neurocomputing*, vol. 21, pp. 203–224, 1998.

[12] B. Schölkopf and A. J. Smola, *Learning with Kernels.* MIT Press, 2002.

[13] E. A. Nadaraya, "On estimating regression," *Theory of Probability and Its Application*, vol. 10, pp. 186–190, 1964.

[14] G. Watson, "Smooth regression analysis," *Sankhya Series A*, vol. 26, pp. 359–372, 1964.

[15] D. W. Scott, *Multivariate Density Estimation.* Wiley, 1992.

[16] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning.* Springer-Verlag, 2001.

[17] S. Sandilya and S. R. Kulkarni, "Principal curves with bounded turn," *IEEE Transactions on Information Theory*, vol. 48, pp. 2789–2793, 2002.

[18] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[20] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15 (6), pp. 1373–1396, June 2003.

[21] H. Reiner, *Handbook of Global Optimization*, ser. Nonconvex optimization and its applications. Dordrecht: Kluwer Academic Publishers, 1995.

[22] G. H. Bakir, J. Weston, and B. Schölkopf, "Learning to find pre-images," in *Advances in Neural Information Processing Systems*, 2003.

[23] J. T. Kwok and I. W. Tsang, "Finding the pre images in kernel principal component analysis," in *6th Annual Workshop On Kernel Machines.* Whistler, 2002.

[24] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik, "Kernel dependency estimation," in *Advances in Neural Information Processing Systems 15*, 2003.

[25] B. Silverman, *Density Estimation for Statistics and Data Analysis*, ser. Monographs on Statistics and Applied Probability. London-New York: Chapman and Hall, 1986.

[26] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder-mead simplex algorithm in low dimensions," *SIAM Journal on Optimization*, vol. 9, pp. 112–147, 1998.

[27] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. of the IEEE Intl. Conf. on Neural Networks*, 1993, pp. 586–591.

[28] J. W. Sammon, Jr., "A non-linear mapping for data structure analysis," *IEEE Transactions on Computers*, vol. 18, pp. 401–409, 1969.

[29] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[30] Y. Bengio, J.-F. P. O. Delalleau, P. Vincent, and M. Ouimet, "Learning eigenfunctions links spectral embedding and kernel PCA," *Neural Computation*, vol. 16, pp. 2197–2219, 2004.