ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Variants of Unsupervised Kernel Regression: General Cost Functions

Stefan Klanke and Helge Ritter

University of Bielefeld, Faculty of Technology, Neuroinformatics Group
P.O. Box 10 01 31, 33501 Bielefeld, Germany

Abstract. We present an extension to a recent method for learning of nonlinear manifolds, which allows to incorporate general cost functions. We focus on the ϵ-insensitive loss and visually demonstrate our method on both toy and real data.

1 Introduction

Unsupervised Kernel Regression (UKR) is a recent approach for the learning of principal manifolds. It has been introduced as an unsupervised counterpart of the Nadaraya-Watson kernel regression estimator in [1]. In this work, we extend UKR by introducing general cost functions, which for example allows to tune the method to specific noise models. The paper is organized as follows: In the next section we recall the UKR algorithm, then we present our extensions with a focus on the ϵ-insensitive loss function and finally we show some example experiments.

2 The UKR Algorithm

In classical (supervised) kernel regression, the Nadaraya-Watson estimator [2, 3]

$$\mathbf{f}(\mathbf{x}) = \sum_i \mathbf{y}_i \frac{K(\mathbf{x} - \mathbf{x}_i)}{\sum_j K(\mathbf{x} - \mathbf{x}_j)} \tag{1}$$

is used to describe a smooth mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$ that generalizes the relation between available input and output data samples $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$. Here, $K(\cdot)$ is a density kernel function, e.g. the Gaussian kernel $K(\mathbf{v}) \propto \exp\left[-\frac{1}{2h^2}\|\mathbf{v}\|^2\right]$, where h is a bandwidth parameter which controls the smoothness of the mapping.

In unsupervised learning, one seeks both a faithful lower dimensional representation (latent variables) $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ of an observed data set $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$ and a corresponding functional relationship.

UKR addresses this problem by using (1) as the mapping from latent space to data space, whereby the latent variables take the role of the input data and are treated as *parameters* of the regression function. By introducing a vector $\mathbf{b}(\cdot) \in \mathbb{R}^N$ of basis functions (holding the fraction of eq. 1), the latter can conveniently be written as $\mathbf{f}(\mathbf{x}; \mathbf{X}) = \mathbf{Y}\mathbf{b}(\mathbf{x}; \mathbf{X})$. While the bandwidth parameter h is crucial in classical kernel regression, here we can set h=1, because the scaling of $\mathbf{X}$ itself is free. Thus, UKR requires no additional parameters besides the

choice of a density kernel[1]. This distinguishes UKR from many other algorithms (e.g. [4, 5]) that, albeit using a similar form of regression function, need an a priori specification of many parameters (e.g. the number of basis functions).

Training an UKR manifold, that is, finding optimal latent variables $\mathbf{X}$, involves gradient-based minimization of the reconstruction error (or empirical risk)

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

where the $N \times N$-matrix of basis functions $\mathbf{B}(\mathbf{X})$ is given by

$$(\mathbf{B}(\mathbf{X}))_{ij} = b_i(\mathbf{x}_j) = \frac{K(\mathbf{x}_i - \mathbf{x}_j)}{\sum_k K(\mathbf{x}_k - \mathbf{x}_j)}. \tag{3}$$

To avoid getting stuck in poor local minima, one can incorporate nonlinear spectral embedding methods (e.g. [6, 7]) to find good initializations.

It is easy to see that without any form of regularization, (2) can be trivially minimized to $R(\mathbf{X}) = 0$ by moving the $\mathbf{x}_i$ infinitely apart from each other. In this case, since $K(\cdot)$ is a density function, $\forall_{i\neq j}\|\mathbf{x}_i - \mathbf{x}_j\| \to \infty$ implies that $K(\mathbf{x}_i - \mathbf{x}_j) \to \delta_{ij}K(\mathbf{0})$ and thus $\mathbf{B}(\mathbf{X})$ becomes the $N \times N$ identity matrix.

2.1 Regularization approaches

2.1.1 Extension of latent space

A straight-forward way to prevent the aforementioned trivial interpolation solution and to control the complexity of an UKR model is to restrict the latent variables to lie within a certain allowed (finite) domain $\mathcal{X}$, e.g. a sphere of radius R. Training of the UKR model then means solving the optimization problem

$$\text{minimize} \quad R(\mathbf{X}) = \frac{1}{N}\|\mathbf{Y} - \mathbf{YB}(\mathbf{X})\|_F^2 \quad \text{subject to} \quad \forall_i\|\mathbf{x}_i\| \leq R. \tag{4}$$

A closely related, but softer and numerically easier method is to add a penalty term to the reconstruction error (2) and minimize $R_e(\mathbf{X},\lambda) = R(\mathbf{X}) + \lambda\sum_i \|\mathbf{x}_i\|^2$. One may choose other forms of penalty terms, for example the general L_p-norm.

With the above formalism, the model complexity can be directly controlled by the pre-factor λ or the parameterization of $\mathcal{X}$. However, normally one has no information about how to choose these parameters. Bigger values of λ lead to stronger overlapping of the density kernels and thus to smoother manifolds, but it is not clear how to select λ to achieve a *certain* degree of smoothness.

2.1.2 Density in latent space

The denominator in (1) is proportional to the Rosenblatt-Parzen density estimator $p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^N K(\mathbf{x} - \mathbf{x}_i)$. Stronger overlap of the kernel functions coincides with higher densities in latent space, which gives rise to another method for complexity control. As in the last section, the density $p(\mathbf{x})$ can be used both in a constraint minimization of $R(\mathbf{X})$ subject to $\forall_i p(\mathbf{x}_i) \geq \eta$ or in form of a penalty function with some pre-factor λ. Compared to a regularization based on the

[1] which is known to be of relatively small importance in classical kernel regression

extension of latent space, the density based regularization tends to work more locally and allows a clustered structure of the latent variables (non-contiguous manifolds). Again, suitable values for λ and η can be difficult to specify.

2.1.3 Leave-one-out cross-validation

Perhaps the strongest feature of UKR is the ability to include leave-one-out cross-validation (LOO-CV) without additional computational cost. Instead of minimizing the reconstruction error of a UKR model including the complete dataset, in LOO-CV each data vector $\mathbf{y}_i$ has to be reconstructed without using $\mathbf{y}_i$ itself:

$$R_{cv}(\mathbf{X}) = \frac{1}{N}\sum_i \|\mathbf{y}_i - \mathbf{f}_{-i}(\mathbf{x}_i;\mathbf{X})\|^2 = \frac{1}{N}\|\mathbf{Y} - \mathbf{Y}\mathbf{B}_{cv}(\mathbf{X})\|_F^2 \tag{5}$$

$$\mathbf{f}_{-i}(\mathbf{x}) = \sum_{l\neq i} \mathbf{y}_l \frac{K(\mathbf{x}-\mathbf{x}_l)}{\sum_{j\neq i} K(\mathbf{x}-\mathbf{x}_j)} \tag{6}$$

For the computation of the matrix of basis functions $\mathbf{B}_{cv}$, this just means zeroing the diagonal elements before normalizing the column sums to 1. A similar strategy works also for calculating the gradient of (5).

As long as the dataset is not totally degenerated (e.g. each $\mathbf{y}_i$ exists at least twice), LOO-CV can be used as a built-in *automatic* complexity control. However, under certain circumstances LOO-CV can severely undersmooth the manifold, particularly in the case of densely sampled noisy data. See Fig. 1 (left plot) for a UKR curve fitted to a sample of a noisy spiral distribution as a result of minimizing the CV-error (5).

3 General Cost Functions

Up to now, the UKR reconstruction error always contained the squared Euclidean distance between training data and its reconstruction as an error measure. In Support Vector Regression it is common practice to replace this by a more general cost function [8], either because it complies with a certain noise model (e.g. L_1 loss for Laplacian noise) or because of application specific needs (e.g. ϵ-insensitive loss to match tolerance levels). For UKR, this means replacing the squared Euclidean norm in (2) by a general cost function $L(\cdot)$, which yields

$$R_L(\mathbf{X}) = \frac{1}{N}\sum_i L(\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i;\mathbf{X})) = \frac{1}{N}\sum_i L(\mathbf{r}_i), \tag{7}$$

with the residuals $\mathbf{r}_i$ given by $\mathbf{r}_i = \mathbf{y}_i - \mathbf{Y}\mathbf{b}(\mathbf{x}_i;\mathbf{X})$. Since UKR training involves gradient-based minimization, the function $L(\cdot)$ has to be at least once differentiable ($L \in C^1$). Then, letting $G_{ji} = (\nabla L(\mathbf{r}_i))_j$, the gradient of the general UKR cost function can be expressed by

$$\frac{\partial R(\mathbf{X})}{\partial X_{ij}} = -\frac{1}{N}\sum_{m,n}(\mathbf{Y}^T\mathbf{G})_{mn}\frac{\partial b_m(\mathbf{x}_n;\mathbf{X})}{\partial X_{ij}}. \tag{8}$$

The special case of $L(\cdot)$ being the squared L_2-norm yields $\mathbf{G} = \mathbf{Y} - \mathbf{Y}\mathbf{B}(\mathbf{X})$, whereby (8) coincides with the normal UKR gradient (see [1]). The L_1-norm is not differentiable and thus can not be used directly. However, Huber's robust loss function, which has the same asymptotic characteristics, can be deployed.

3.1 ϵ-insensitive loss function

In Support Vector Regression a popular choice is the ϵ-insensitive loss function

$$L_\epsilon(r) = \begin{cases} 0 & |r| < \epsilon \\ |r| - \epsilon & |r| \geq \epsilon \end{cases} \tag{9}$$

which is not differentiable at $|r| = \epsilon$. By squaring it, we get a differentiable variant, which asymptotically behaves like the squared Euclidean error, but does not penalize the UKR manifold as long as the reconstructions lie within an epsilon box around the corresponding original data vectors.

Using this loss function without any form of regularization does not make much sense, since a wiggly and a smooth manifold that both pass through the above mentioned epsilon boxes do not differ in terms of the UKR loss. Furthermore, simply adding a penalty term from section 2.1.1 or 2.1.2 still requires the specification of a suitable pre-factor λ. A practical solution is to switch the roles of penalty term and objective function and to choose a very large λ. This corresponds to solving the problem

$$\text{minimize } S(\mathbf{X}) \quad \text{subject to} \quad R_{L_\epsilon}(\mathbf{X}) = 0, \tag{10}$$

where for example $S(\mathbf{X}) = \sum_i \|\mathbf{x}_i\|^2$ (cf. section 2.1.1). In other words, we search the smoothest manifold (that of least possible latent extension) which passes through every data vectors epsilon box. Of course, one still has to specify suitable values for ϵ, but this is geometrically much more intuitive and may even be pre-defined by application specific needs.

Note that different data vectors may be assigned different ϵ values. In practice, it works best to first fit a rather un-smooth manifold regularized by LOO-CV and then use this result as an initialization for minimizing (10). This has the advantage that one can check the residual errors made already by the un-smooth manifold, which in turn gives an upper bound for the per-data-vector ϵ-values. This is a convenient tool for reducing the influence of outliers and guarantees a valid solution. See section 4 for example experiments.

3.2 Comparison to feature space UKR

As another way of incorporating general and even non differentiable loss functions, one might consider using *feature space* UKR. The basic idea is to construct a Mercer kernel that implicitly maps the data to some feature space so that the standard Euclidean norm in that space coincides with the demanded loss function in data space. As an example consider the L_1-kernel[2] $k(\mathbf{y}, \mathbf{y}') = \frac{1}{2}(|\mathbf{y}|+|\mathbf{y}'|-|\mathbf{y}-\mathbf{y}'|)$ through which the Euclidean distance between two feature space images is given by $\|\Phi(\mathbf{y}) - \Phi(\mathbf{y}')\|^2 = |\mathbf{y} - \mathbf{y}'|$.

However, there is one main problem with feature space UKR: The regression function $\mathbf{f}(\mathbf{x}; \mathbf{X})$ now linearly combines feature space images and cannot be calculated explicitly for general Mercer kernels. Furthermore, there is no guaranteed existence of a pre-image of such linear combinations.

[2] see [1] for a derivation and an application

4 Experiments

Noisy Spiral. As a first example, we fitted a UKR model to a 2D "noisy spiral" toy dataset, which contains 300 samples with noise distributed uniformly in the interval $[-0.1; 0.1]$. For initialization, we calculated multiple LLE [6] solutions of neighborhood sizes $K = 4 \dots 12$, which we compared with respect to their CV-error after a coarse optimization of their overall scale. While this procedure may seem rather computationally expensive, it greatly enhances the robustness, because LLE and other nonlinear spectral embedding methods can depend critically on the choice of K. In this experiment, the best set of latent variables belonged to $K = 7$, which we further fine-tuned by gradient-based minimization using the RPROP scheme [9]. Please see [1] for a more detailed description of UKR training. After fitting the CV-regularized manifold, we measured the errors made in the reconstruction of each data vector, which served as an upper bound for our per-data-vector ϵ-values. The lower bound (responsible for smoothing the curve) was set to $\epsilon = 0.02$, $\epsilon = 0.04$, $\epsilon = 0.06$ and finally $\epsilon = 0.08$. Fig. 1 shows the UKR manifolds resulting from applying the constrained optimization (10), which we realized by using $R_{L_\epsilon}(\mathbf{X})$ as a penalty term in conjunction with the RPROP algorithm. Note the increasing smoothness of the curve for larger values of ϵ, which is paid with an increased bias towards the inner of the spiral.

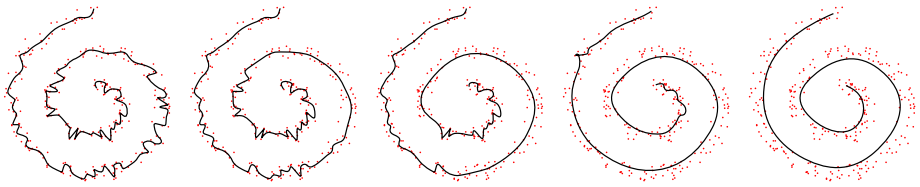

Fig. 1: UKR model of a noisy spiral using LOO-CV and the ϵ-insensitive loss function, respectively. From left to right: CV regularized, $\epsilon = 0.02$, $\epsilon = 0.04$, $\epsilon = 0.06$, $\epsilon = 0.08$. The black line shows $\mathbf{f}(\mathbf{x}; \mathbf{X})$ sampled in latent space, the dots depict the original data vectors.

USPS Digit "2". As a second example, we fitted a 2D UKR manifold to the subset of the USPS handwritten digits dataset corresponding to the digit "2", which consists of 731 data vectors in 256 dimensions (16x16 pixel gray-scale images). Again, for initialization we picked an LLE solution ($K = 12$) from a set of candidates and then fitted a CV-regularized manifold by minimizing (5) with the RPROP scheme (for details please see [1]). In this dataset, most pixel values are ± 1. Since for smoothing the manifold we wanted to ignore small errors in the 16x16 image as a whole and not on a pixel-by-pixel basis, we used the ϵ-insensitive loss in the form $L_\epsilon(\|\mathbf{y} - \mathbf{f}(\mathbf{x})\|)$. The upper bounds for the per-data-vector ϵ-values were calculated like before. The lower bound was set to $\epsilon = 10$, which roughly corresponds to 25 wrong pixels. Fig. 2 visualizes the resulting manifolds by sampling $\mathbf{f}(\mathbf{x}; \mathbf{X})$ on a grid in latent space and depicting the function value as an 16x16 image. Note the smaller extension of latent space and the blurrier images from the ϵ-optimized manifold (right plot).

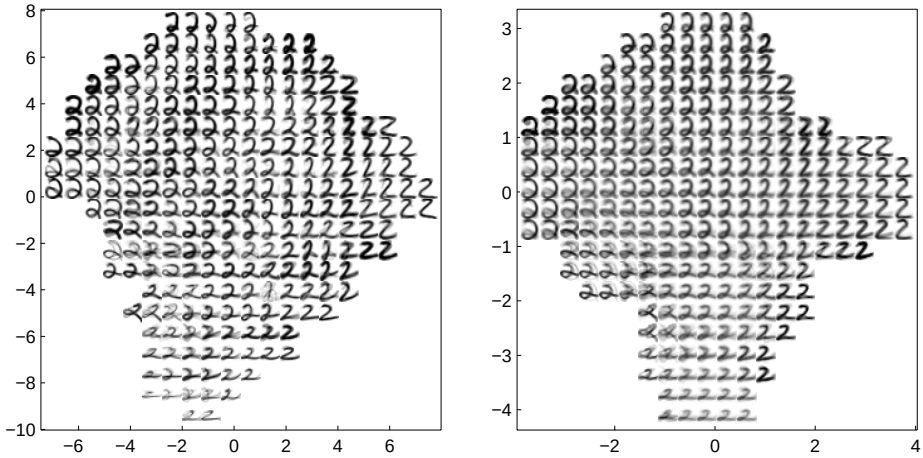

Fig. 2: UKR model of the USPS digit "2", shown by evaluating $\mathbf{f}(\mathbf{x};\mathbf{X})$ on a 20x20 grid enclosing the latent variables. Grid positions of low density $p(\mathbf{x})$ are left blank. Left: CV regularized. Right: Squared ϵ-insensitive loss as penalty function, $\epsilon = 10$.

5 Conclusion

We presented an extension to the manifold learning method UKR, which now allows to incorporate general cost functions. We focused on the ϵ-insensitive loss and demonstrated a practical optimization approach on both toy and real data.

References

[1] P. Meinicke, S. Klanke, R. Memisevic, and H. Ritter. Principal surfaces from unsupervised kernel regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1379–1391, September 2005.

[2] E.A. Nadaraya. On estimating regression. *Theory of Probability and Its Application*, 10:186–190, 1964.

[3] G.S. Watson. Smooth regression analysis. *Sankhya Series A*, 26:359–372, 1964.

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

[5] A.J. Smola, R.C. Williamson, S. Mika, and B. Schölkopf. Regularized principal manifolds. *Lecture Notes in Computer Science*, 1572:214–229, 1999.

[6] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

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

[8] A. Smola, B. Schölkopf, and K. Müller. General cost functions for support vector regression. In T. Downs, M. Frean, and M. Gallagher, editors, *Proc. of the 9th Australian Conf. on Neural Networks*, pages 79–83, Brisbane, Australia, 1998.

[9] 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*, pages 586–591, 1993.