Optimal Kernel Shapes for Local Linear Regression

DIRK ORMONEIT AND TREVOR HASTIE ormoneit@stat.stanford.edu

Department of Computer Science, Technische Universität München
Department of Statistics, Stanford University, Stanford, CA 94305-4065

Abstract. Local linear regression performs very well in many low-dimensional forecasting problems. In high-dimensional spaces, its performance typically decays due to the well-known "curse-of-dimensionality". Specifically, the volume of a weighting kernel that contains a fixed number of samples increases exponentially with the number of dimensions. The bias of a local linear estimate may thus become unacceptable for many real-world data sets. A possible way to control the bias is by varying the "shape" of the weighting kernel. In this work we suggest a new, data-driven method to estimating the optimal kernel shape. Experiments using two artificially generated data sets and data from the UC Irvine repository show the benefits of kernel shaping.

Keywords: Local Linear Regression, Smoothing, Lazy Learning, Entropy, Bandwidth Selection, Nearest Neighbors, Projection Pursuit Regression, Sliced Inverse Regression

1. Introduction

Local linear regression has attracted considerable attention in both statistical and machine learning literature as a flexible tool for nonparametric regression analysis [3, 5, 1]. Like most statistical smoothing approaches, local modeling suffers from the so-called "curse-of-dimensionality", the well-known fact that the proportion of the training data that lie in a fixed-radius neighborhood of a point decreases to zero at an exponential rate with increasing dimension of the input space. Due to this problem, the bandwidth of a weighting kernel must be chosen very big so as to contain a reasonable sample fraction. As a result, the estimates produced are typically highly biased. One possible way to reduce the bias of local linear estimates is to vary the "shape" of the weighting kernel. In this work, we suggest a method for estimating the optimal kernel shape using the training data. For this purpose, we parameterize the kernel in terms of a suitable "shape matrix", L, and minimize the mean squared forecasting error with respect to L. For such an approach to be meaningful, the "size" of the weighting kernel must be constrained during the minimization to avoid overfitting. We propose a new, entropy-based measure of the kernel size as a constraint. In analogy to the nearest neighbor approach to bandwidth selection [5], the suggested measure is adaptive with regard to the local data density. In addition, it leads to an efficient gradient descent algorithm for the computation of the optimal kernel shape. Experiments using artificially generated data sets and data from the UC Irvine repository show that kernel shaping can improve the performance of local linear estimates substantially.

The remainder of this work is organized as follows. In Section 2 we briefly review local linear models and introduce our notation. In Section 3 we formulate an objective function for kernel shaping, and in Section 4 we discuss entropic neighborhoods. The estimation procedure is described in Section 5, and Section 6 summarizes our experimental results. Section 7 presents conclusions and delineates some of our efforts to apply kernel shaping locally. Also, we discuss the relationship between kernel shaping and alternative methods such as projection pursuit regression and sliced inverse regression in Section 7.

2. Local Linear Models

Consider a nonlinear regression problem where a continuous response $y \in \mathbb{R}$ is to be predicted based on a d-dimensional predictor $x \in \mathbb{R}^d$. Let $D \equiv \{(x_t, y_t), t = 1, \ldots, T\}$ denote a set of training data. To estimate the conditional expectation $f(x_0) \equiv E[y|x_0]$, we consider the local linear expansion $f(x) \approx \alpha_0 + (x - x_0)'\beta_0$ in the neighborhood of x_0. In detail, we minimize the weighted least squares criterion

$$\mathcal{C}(\alpha, \beta; x_0) \equiv \sum_{t=1}^{T} (y_t - \alpha - (x_t - x_0)'\beta)^2 k(x_t, x_0) \tag{1}$$

to determine estimates of the parameters α_0 and β_0. A standard way to define the *weighting kernel* $k(x_t, x_0)$ is by applying a univariate, non-negative "mother kernel" $\phi(z)$ to the distance measure $||x_t - x_0||_\Omega \equiv \sqrt{(x_t - x_0)'\Omega(x_t - x_0)}$:

$$k(x_t, x_0) \equiv \frac{\phi\left(||x_t - x_0||_\Omega\right)}{\sum_{s=1}^{T} \phi\left(||x_s - x_0||_\Omega\right)}. \tag{2}$$

Intuitively, $k(x_t, x_0)$ assigns more weight to residuals in the neighborhood of x_0 than to residuals distant from x_0. Here Ω is a positive definite $d \times d$ matrix determining the relative importance assigned to different directions of the input space. For example, if $\phi(z)$ is a standard normal density, $k(x_t, x_0)$ is a normalized multivariate Gaussian with mean x_0 and covariance matrix Ω^{-1}. For simplicity, $k(x_t, x_0)$ is defined so as to satisfy $\sum_{t=1}^{T} k(x_t, x_0) = 1$ in equation (2). Though this rescaling is irrelevant with regard to the minimization of (1), it will be convenient for our discussion of entropic neighborhoods in Section 4.

Using the shorthand notation $\hat{\gamma}(x_0, \Omega) \equiv (\hat{\alpha}_0, \hat{\beta}_0')'$, the solution of the minimization problem (1) may conveniently be written as

$$\hat{\gamma}(x_0, \Omega) = (X^{0\prime} W^0 X^0)^{-1} X^{0\prime} W^0 Y, \tag{3}$$

where X^0 is the $T \times (d+1)$ design matrix with rows $(1, x_t' - x_0')'$, Y is the vector of response values, and W^0 is a $T \times T$ diagonal matrix with entries $W^0_{t,t} = k(x_t, x_0)$.

The resulting local linear fit at x_0 using the inverse covariance matrix Ω is simply $\hat{f}(x_0; \Omega) \equiv \hat{\alpha}_0$. Obviously, $\hat{f}(x_0; \Omega)$ depends on Ω through the definition of the weighting kernel (2). In the discussion below, our focus is on choices of Ω that lead to favorable estimates of the unknown function value $f(x_0)$.

3. Kernel Shaping

The local linear estimates resulting from different choices of Ω vary considerably in practice. A common strategy is to choose Ω proportional to the inverse sample covariance matrix. The problem of finding the correct scaling factor is then equivalent to the problem of bandwidth selection in univariate smoothing [5]. For example, the bandwidth is frequently chosen as a function of the distance between x_0 and its kth nearest neighbor in practical applications. In this paper, we take a different viewpoint and argue that optimizing the "shape" of the weighting kernel is at least as important as optimizing the bandwidth. More specifically, for a fixed "volume" of the weighting kernel, the bias of the estimate can be reduced drastically by shrinking the kernel in directions of large nonlinear variation of $f(x)$, and stretching it in directions of small nonlinear variation. This statement can be made formal by noting that the asymptotic bias of a local linear estimate at x_0 is proportional to $tr(\Omega^{-1}\mathcal{H}(x_0))$, where $\mathcal{H}(x_0)$ is the Hessian of $f(x)$ [8]. The idea of kernel shaping is illustrated using the example shown in Figure 1. The plotted function is sigmoidal along an index vector κ and constant in directions orthogonal to κ. Therefore, a "shaped" weighting kernel is shrunk in the direction κ and stretched in the orthogonal direction, thus minimizing the exposure of the kernel to the nonlinear variation.

To formally distinguish the metric and the bandwidth of the weighting kernel, we rewrite Ω as follows:

$$\Omega \equiv \lambda \cdot (LL' + I). \tag{4}$$

Here λ corresponds to the inverse of the bandwidth, and L may be interpreted as a metric- or shape-matrix. Below we suggest an algorithm which is designed to min-

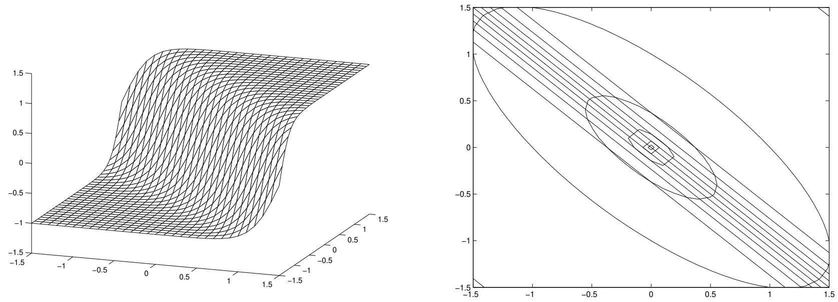

Figure 1. *Left:* Example of a single index model of the form $y = g(x'\kappa)$ with $\kappa = (1,1)$ and $g(z) = \tanh(3z)$. *Right:* The contours of this function are straight lines orthogonal to κ.

imize the bias with respect to the kernel metric. Clearly, for such an approach to be meaningful, we need to restrict the "volume" of the weighting kernel; otherwise, the bias of the estimate could be minimized trivially by choosing a zero bandwidth. There are several ways to define the kernel volume. One possibility would be to consider the determinant $|\Omega|$. Then a natural way to formulate the volume constraint would be to choose λ in equation (4) so as to satisfy $|\Omega| = c$ for some constant c. A serious disadvantage of this idea is that, by contrast to the nearest neighbor approach, $|\Omega|$ is independent of the design. As a more appropriate alternative, we define λ in terms of a measure of the number of neighboring observations. In detail, we fix the volume of $k(x_t, x_0)$ in terms of the "entropy" of the weighting kernel. Then, we choose λ so as to satisfy the resulting entropy constraint. Given this definition of the bandwidth, we determine the metric of $k(x_t, x_0)$ by minimizing the mean squared prediction error:

$$\mathcal{E}(L; D) \equiv \sum_{t=1}^{T} (y_t - \hat{f}(x_t; \Omega))^2 \tag{5}$$

with respect to L. In this way, we obtain an approximation of the optimal kernel shape because the expectation of $\mathcal{E}(L; D)$ differs from the bias only by a variance term which is independent of L. Details of the entropic neighborhood criterion and of the numerical minimization procedure are described next.

4. Entropic Neighborhoods

We mentioned above that, for a given shape matrix L, we choose the bandwidth parameter λ in (4) so as to fulfill a volume constraint on the weighting kernel. For this purpose, we interpret the kernel weights $k(x_t, x_0)$ as probabilities. In particular, as $k(x_t, x_0) > 0$ and $\sum_t k(x_t, x_0) = 1$ by definition (2), we can formulate the local *entropy* of $k(x_t, x_0)$:

$$H(\Omega) \equiv -\sum_{t=1}^{T} k(x_t, x_0) \log k(x_t, x_0). \tag{6}$$

The entropy of a probability distribution is typically thought of as a measure of uncertainty. In the context of the weighting kernel $k(x_t, x_0)$, $H(\Omega)$ may be used as a smooth measure of the "size" of the neighborhood that is used for averaging. To see this, note that in the extreme case where equal weights are placed on all observations in D, the entropy is maximized. At the other extreme, if the single nearest neighbor of x_0 is assigned the entire weight of one, the entropy attains its minimum value zero. Thus, fixing the entropy at a constant value c is essentially equivalent to fixing the number k in the nearest neighbor approach. Besides justifying (6), the correspondence between k and c can also be used to derive an intuitive parametrization of the entropy constraint. In detail, we specify c in terms of a hypothetical weighting kernel that places equal weight on the k nearest neighbors of x_0 and zero weight on the remaining observations. Note that the entropy of this hypothetical kernel is $\log k$. Thus, it is natural to characterize the size of an entropic neighborhood in terms of k, and then to determine λ by numerically solving the nonlinear equation system

$$H(\Omega) = \log k. \tag{7}$$

More precisely, we report the number of neighbors in terms of the *equivalent sample fraction* $\rho \equiv k/T$ to further intuition. This idea is illustrated in Figure 2 using a one- and a two-dimensional example. The equivalent sample fractions are $\rho = 30\%$ and $\rho = 50\%$, respectively. Note that in both cases the weighting kernel is wider in regions with few observations, and narrower in regions with many observations.

As a consequence, the number of observations within contours of equal weighting remains approximately constant across the predictor space.

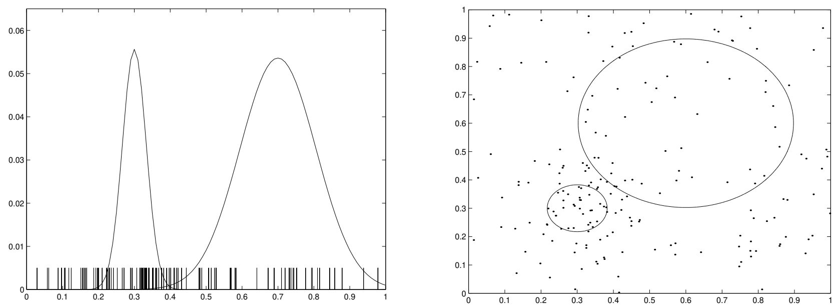

Figure 2. *Left:* Univariate weighting kernel $k(\cdot, x_0)$ evaluated at $x_0 = 0.3$ and $x_0 = 0.7$ based on a sample data set of 100 observations (indicated by the bars at the bottom). The samples were generated according to a mixture of a uniform distribution $U(0,1)$ that was selected with probability 0.4, and of a normal distribution $N(0.3, 0.1^2)$. We chose an equivalent neighborhood size of 30 ($\rho = 30/100 = 30\%$). *Right:* Multivariate weighting kernel $k(\cdot, x_0)$ based on a sample data set of 200 observations. The samples were generated according to a mixture of a uniform distribution on $[0,1]^2$ that was selected with probability 0.4, and of a normal distribution with mean (0.3,0.3) and covariance matrix $(0.1^2, 0; 0, 0.1^2)$. The two ellipsoids correspond to 95% contours of a weighting kernel evaluated at $(0.3, 0.3)'$ and $(0.6, 0.6)'$.

To summarize, we define the value of λ by fixing the equivalent sample fraction parameter ρ, and subsequently minimize the prediction error on the training set with respect to the shape matrix L. As a minimization procedure, we use a variant of gradient descent that accounts for the entropy constraint. Details of the optimization procedure are described next.

5. Estimation

For the practical implementation of kernel shaping, we need an efficient procedure for the minimization of (5) with respect to the shape matrix L. In particular, as $\hat{f}(x_t, \Omega)$ depends on L in a rather complicated fashion, no closed-form expression for the optimal kernel shape is available in general. Using numerical methods instead, we have to account for the entropy constraint (7) during minimization which prohibits the use of standard toolboxes. Specifically, we suggest to use a variant of the gradient descent algorithm where the gradient is evaluated using projections. This approach performed well in our experiments, identifying the optimal solution in relatively few iterations and largely independent of the initialization values.

At the heart of our algorithm is the evaluation of the gradient of the prediction error (5):

$$\frac{\partial \mathcal{E}}{\partial L} = -2\sum_{t=1}^{T}\left[(y_t - \hat{f}(x_t;\Omega^t))\frac{\partial}{\partial L}\hat{f}(x_t;\Omega^t)\right]. \tag{8}$$

We use the notation Ω^t to emphasize that, given our exposition in Section 4, Ω depends on the location of forecast x_t through the value of λ in equation (4). To apply equation (8) in practice, we first need to evaluate the derivative matrix $\frac{\partial}{\partial L}\hat{f}(x_t;\Omega^t)$; this evaluation is carried out in two steps, first accounting for the dependence of the local linear estimate $\hat{f}(x_t;\Omega^t)$ on Ω^t, and subsequently adjusting for the reparametrization (4). With regard to the first step, note that $\hat{f}(x_t;\Omega^t)$ depends on Ω^t through changes in $\hat{\alpha}_t$, the first component of $\hat{\gamma}(x_t,\Omega^t)$, as the weighting matrix W^t changes. Implicit differentiation of equation (3) gives that

$$\frac{\partial}{\partial \Omega^t}\hat{f}(x_t;\Omega^t) = e_1'(X^{t\prime}W^tX^t)^{-1}X^{t\prime}V^t, \tag{9}$$

where $e_1 \equiv (1,0,\ldots,0)'$ and V^t is a $T\times d\times d$ array defined in Appendix A.1.

In the second step, we apply the chain rule to $\frac{\partial}{\partial\Omega^t}\hat{f}(x_t;\Omega^t)$; together with the condition $\frac{\partial H}{\partial L} = 0$ (H is a constant) we obtain that

$$\frac{\partial}{\partial L}\hat{f}(x_t;\Omega^t) = 2\lambda\left(\frac{\partial}{\partial\Omega^t}\hat{f}(x_t;\Omega^t) - \xi\frac{\partial H}{\partial \Omega^t}\right)L. \tag{10}$$

This result is formally derived in Appendix A.1. Interestingly enough, the term in parenthesis can be interpreted as a projection of the unconstrained gradient $\frac{\partial}{\partial\Omega^t}\hat{f}(x_t;\Omega^t)$ onto the manifold that is characterized by equation (7).

Given the error gradient $\nabla\mathcal{E} \equiv \frac{\partial\mathcal{E}}{\partial L}$, we carry out a straightforward gradient descent search for the optimal L:

$$L^{(s+1)} := L^{(s)} - \alpha\nabla\mathcal{E}. \tag{11}$$

Here α is a step-size parameter that has to be determined experimentally. Note that in Section 4 we defined the multiplier λ contingent on the shape matrix L. Therefore, λ must be updated after each gradient descent step. We spent considerable effort on the development of an efficient numerical algorithm for this purpose, which is described in detail in Appendix A.2. In most test runs, the gradient descent algorithm converged after a few hundred steps, corresponding to about 15 minutes of computation time on a heavily used HP workstation for a data set of 500 observations. In particular, more sophisticated optimization routines such as a robust conjugate gradient method combined with a backtracking line search led to no efficiency improvements in our experiments.

From a statistical perspective, a crucial issue with regard to the estimation error is the number of free parameters in L. As a means of controlling this number, we allow for the possibility that L may be of reduced rank $l \leq d$. Note also that, as a result of definition (4), Ω is positive-definite regardless of the value of L. A summary of our experimental results is described next.

6. Experiments

In this section we will compare kernel shaping to standard local linear regression using a fixed spherical kernel in three examples. First, we evaluate the performance using a five-dimensional toy problem which allows us to estimate confidence intervals for the prediction accuracy using Monte Carlo simulation. Second, we repeat the same experiment using a ten-dimensional predictor space. Third, we investigate a data set from the machine learning data base at UC Irvine [2].

6.1. Mexican Hat Function I

In our first example, we employ Monte Carlo simulation to evaluate the performance of kernel shaping in a five-dimensional regression problem. For this purpose, 20 sets of 500 data points each are generated independently according to the model

$$y = \cos(5\sqrt{x_1^2 + x_2^2}) \cdot \exp(-(x_1^2 + x_2^2)). \tag{12}$$

Here the predictor variables $x_1, \ldots, x_5$ are drawn randomly according to a five-dimensional standard normal distribution. Note that, even though the regression is carried out in a five-dimensional predictor space, y is really only a function of the variables x_1 and x_2. In particular, as dimensions two through five do not contribute any information with regard to the value of y, kernel shaping should effectively discard these variables. Note also that there is no noise in this example.

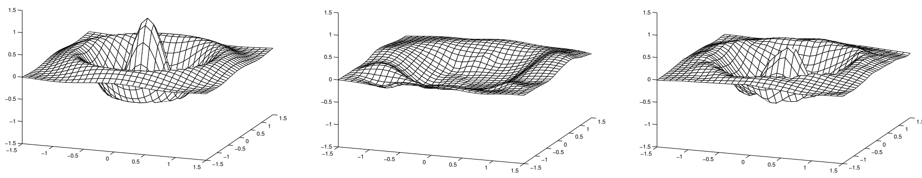

Figure 3. *Left:* "True" Mexican hat function. *Middle:* Local linear estimate using a spherical kernel ($\rho = 2\%$). *Right:* Local linear estimate using kernel shaping ($\rho = 2\%$). Both estimates are based on a training set consisting of 500 data points.

Figure 3 shows a plot of the true function, the spherical estimate, and the estimate using kernel shaping as functions of x_1 and x_2. The true function has the familiar "Mexican hat" shape, which is recovered by the estimates to different degrees. To quantify the influence of kernel shaping on the predictive performance in this example, we evaluate the local linear estimates for values of the equivalent neighborhood fraction parameter ρ in the range from 1% to 15%. For each value of ρ, 20 models are estimated using the 20 artificially generated training sets, and subsequently their performance is evaluated on the training set as well as on the test set of 31×31 grid points shown in Figure 3. The shape matrix L has maximal rank $l = 5$ in this experiment. Our results for local linear regression using both a fixed spherical kernel and kernel shaping are summarized in Table 1. Performance is measured in terms of the mean R^2-value of the 20 models, and standard deviations are reported in parenthesis.

The results in Table 1 indicate that the optimal performance on the test set is obtained using the parameter values $\rho = 2\%$ both for kernel shaping ($R^2 = 0.909$)

Table 1. Performances in the first toy example. The results for kernel shaping were obtained using 200 gradient descent steps.

Kernel		**Training**	**Test**
spherical kernel	$\rho = 1\%$	0.961 (0.005)	0.215 (0.126)
spherical kernel	$\rho = 2\%$	0.871 (0.014)	0.293 (0.082)
spherical kernel	$\rho = 5\%$	0.680 (0.029)	0.265 (0.043)
spherical kernel	$\rho = 10\%$	0.507 (0.038)	0.213 (0.030)
spherical kernel	$\rho = 20\%$	0.341 (0.039)	0.164 (0.021)
kernel shaping	$\rho = 1\%$, $\alpha = 0.2$	0.995 (0.001)	0.882 (0.024)
kernel shaping	$\rho = 2\%$, $\alpha = 0.2$	0.984 (0.002)	0.909 (0.017)
kernel shaping	$\rho = 5\%$, $\alpha = 0.2$	0.923 (0.009)	0.836 (0.023)
kernel shaping	$\rho = 15\%$, $\alpha = 0.2$	0.628 (0.035)	0.517 (0.035)

and and for the spherical kernel ($R^2 = 0.293$). Given the large difference between the R^2 values, we conclude that kernel shaping clearly outperforms the spherical kernel on this data set.

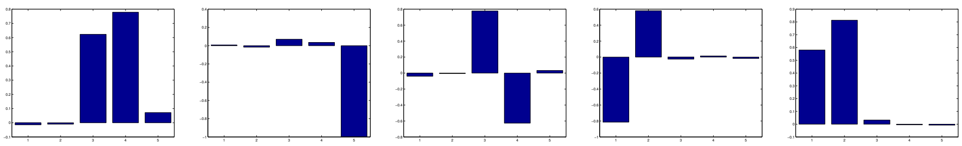

Figure 4. The eigenvectors of the estimate of Ω obtained using the first of twenty training sets. The graphs are ordered from left to right by increasing eigenvalues (decreasing extension of the kernel in that direction): 0.76, 0.76, 0.76, 33.24, 34.88.

Finally, Figure 4 shows the eigenvectors of the optimized Ω on the first training set. The eigenvectors are ordered according to the size of the corresponding eigenvalues. Note that the two rightmost eigenvectors, which correspond to the directions of minimum kernel extension, span exactly the x_1-x_2-space where the true function lives. The kernel is stretched in the remaining directions, effectively discarding nonlinear contributions from x_3, x_4, and x_5.

6.2. Mexican Hat Function II

Next, we repeat the previous experiment in an even higher-dimensional environment using ten predictor variables instead of five. In detail, the artificial data are again generated according to model (12); however, x_1 and x_2 are augmented by eight normally distributed "nuisance" variables $x_3, \ldots, x_{10}$ in this case. To make this difficult prediction problem more feasible, we double the size of the training set from 500 to 1000. Simultaneously, the number of training data sets used for the Monte Carlo simulation is reduced from 20 to 10 to limit the computational burden. Note that the maximum number of free parameters in the shape matrix L quadruples in comparison to the previous section. To control the estimation efficiency, we choose a shape matrix of rank $l = 5$ in this experiment. Note that we make no use of the fact that really a rank of two would be sufficient to represent the Mexican Hat function.

The results of our experiments are summarized in Table 1 and in Figure 5, which are organized as in Section 6.1. Note that the performance of the spherical estimate suffers drastically from the increase of dimensionality (optimal $R^2 = 0.117$ for $\rho = 10\%$ in comparison to $R^2 = 0.293$ in Section 6.1). Using kernel shaping, the optimal performance also decreases ($R^2 = 0.635$ for $\rho = 10\%$ in comparison to $R^2 = 0.909$). Again, kernel shaping clearly outperforms local linear estimation using the spherical kernel. With regard to the shape matrix Ω, the eigenvectors in Figure 5 qualitatively resemble our findings of the previous section. That is, the weighting kernel correctly identifies the two most important random variables x_1 and x_2 *(bottom right)* and discards nonlinear contributions from $x_3, \ldots, x_{10}$. However, the estimates are more noisy due to the increased number of free parameters.

6.3. Abalone Database

The task in our third example is to predict the age of abalone based on several measurements. More specifically, the response variable is obtained by counting the number of rings in the shell in a time-consuming procedure. Preferably, the

Table 2. Performances in the second Mexican Hat example. The results for kernel shaping were obtained using 200 gradient descent steps.

Kernel		**Training**	**Test**
spherical kernel	$\rho = 1\%$	0.983 (0.001)	-0.008 (0.257)
spherical kernel	$\rho = 2\%$	0.958 (0.003)	0.060 (0.175)
spherical kernel	$\rho = 5\%$	0.886 (0.008)	0.106 (0.090)
spherical kernel	$\rho = 10\%$	0.780 (0.011)	0.117 (0.050)
spherical kernel	$\rho = 20\%$	0.605 (0.014)	0.111 (0.025)
spherical kernel	$\rho = 50\%$	0.269 (0.011)	0.078 (0.009)
spherical kernel	$\rho = 95\%$	0.043 (0.005)	0.007 (0.006)
kernel shaping	$l = 5,\ \rho = 10\%,\ \alpha = 0.2$	0.826 (0.014)	0.502 (0.125)
kernel shaping	$l = 5,\ \rho = 20\%,\ \alpha = 0.2$	0.629 (0.013)	0.183 (0.032)
kernel shaping	$l = 5,\ \rho = 10\%,\ \alpha = 0.5$	0.838 (0.013)	0.608 (0.098)
kernel shaping	$l = 5,\ \rho = 10\%,\ \alpha = 1$	0.840 (0.013)	0.635 (0.072)
kernel shaping	$l = 5,\ \rho = 20\%,\ \alpha = 1$	0.629 (0.013)	0.188 (0.035)
kernel shaping	$l = 5,\ \rho = 10\%,\ \alpha = 2$	0.841 (0.011)	0.660 (0.027)
kernel shaping	$l = 5,\ \rho = 10\%,\ \alpha = 5$	0.828 (0.034)	0.566 (0.217)
kernel shaping	$l = 5,\ \rho = 20\%,\ \alpha = 5$	0.594 (0.013)	0.145 (0.029)
kernel shaping	$l = 5,\ \rho = 10\%,\ \alpha = 10$	0.798 (0.061)	0.509 (0.265)
kernel shaping	$l = 5,\ \rho = 20\%,\ \alpha = 10$	0.535 (0.020)	0.120 (0.032)

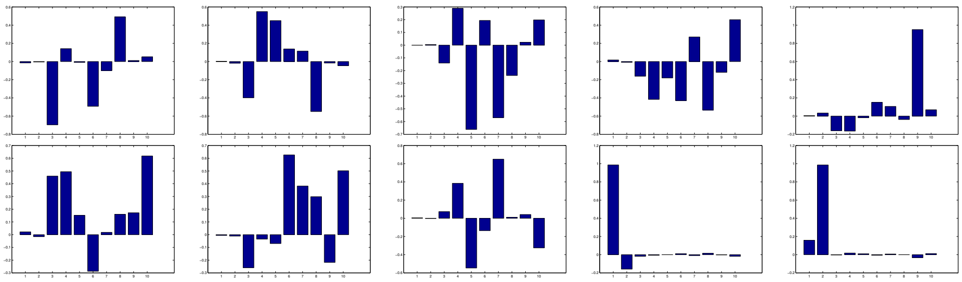

Figure 5. The eigenvectors of the estimate of Ω obtained using the first of ten training sets. The graphs are ordered from top left to bottom right by increasing eigenvalues (decreasing extension of the kernel in that direction): 0.1817, 0.1817, 0.1817, 0.1817, 0.1817, 0.2431, 0.2834, 0.3038, 12.1833, 14.5929.

age of the abalone could be predicted from alternative measurements that may be obtained more easily. In the data set, eight candidate measurements including sex, dimensions, and various weights are reported along with the number of rings of the abalone. The predictors are normalized to zero mean and unit variance prior to estimation. Overall, the data set consists of 4177 observations. To prevent possible artifacts resulting from the order of the data records, we randomly draw 2784 observations as a training set and use the remaining 1393 observations as a test set. Our results are summarized in Table 3 using various settings for the rank l, the equivalent fraction parameter ρ, and the gradient descent step size α. The optimal choice for ρ is 20% both for kernel shaping ($R^2 = 0.582$) and for the spherical kernel ($R^2 = 0.572$). Note that the performance improvement due to kernel shaping is negligible in this experiment.

Table 3. Results using the Abalone database after 200 gradient descent steps.

Kernel		**Training R^2**	**Test R^2**
spherical kernel	$\rho = 5\%$	0.752	0.543
spherical kernel	$\rho = 10\%$	0.686	0.564
spherical kernel	$\rho = 20\%$	0.639	<u>0.572</u>
spherical kernel	$\rho = 50\%$	0.595	0.565
spherical kernel	$\rho = 70\%$	0.581	0.552
spherical kernel	$\rho = 90\%$	0.568	0.533
kernel shaping	$l = 5$, $\rho = 20\%$, $\alpha = 0.5$	0.705	0.575
kernel shaping	$l = 5$, $\rho = 20\%$, $\alpha = 0.2$	0.698	0.577
kernel shaping	$l = 2$, $\rho = 10\%$, $\alpha = 0.2$	0.729	0.574
kernel shaping	$l = 2$, $\rho = 20\%$, $\alpha = 0.2$	0.663	<u>0.582</u>
kernel shaping	$l = 2$, $\rho = 50\%$, $\alpha = 0.2$	0.603	0.571
kernel shaping	$l = 2$, $\rho = 20\%$, $\alpha = 0.5$	0.669	<u>0.582</u>

7. Extensions and Conclusions

We introduced a data-driven method to improve the performance of local linear estimates in high dimensions by optimizing the shape of the weighting kernel. In our experiments we found that kernel shaping clearly outperformed local linear regression using a spherical kernel in a five- and a ten-dimensional toy example, and led to a small performance improvement in a third, real-world example. To explain the results of the third experiment, we note that kernel shaping aims at exploiting global structure in the data. Thus, the absence of a larger performance improvement may simply suggest that there is no globally optimal kernel shape in this example. That is, even though an optimal kernel shape may exist *locally* at each point of prediction, the optimal shapes vary sufficiently across the predictor space so that they cannot all be approximated by any particular *global* shape.

7.1. "Local" Kernel Shaping

A promising idea to circumvent this difficulty is to apply kernel shaping locally at x_0. To do this, we augment the error criterion (5) with a "secondary" weighting kernel $\bar{k}(x_t, x_0)$:

$$\mathcal{E}^{local}(L; x_0, D) \equiv \sum_{t=1}^{T} \bar{k}(x_t, x_0)(y_t - \hat{f}(x_t; \Omega))^2. \tag{13}$$

Intuitively, $\bar{k}(x_t, x_0)$ constrains the local domain of the training data used to estimate the shape matrix L. Minimizing (13) at different locations of the predictor space may thus lead to different estimates of L. To assess this approach, we applied (13) to the Mexican Hat example of Section 6.1, omitting the "nuisance variables" x_3, x_4, and x_5 for simplicity. The shape estimates at three sample points are shown in Figure 6. Note that the computed kernel shapes are intuitively obvious; that is, they correctly emphasize the tangential direction of the rotation-symmetric surface along which the function is locally constant. For more complicated problems, however, we found that the shape estimates become extremely noisy, because the secondary kernel $\bar{k}(x_t, x_0)$ effectively reduces the number of available training data. As an additional disadvantage, the numerical minimization of (13) has to be carried out at each location of forecast x_0 rather than once and for all as in "global" kernel shaping. Due to these difficulties, our efforts along these lines did not lead to a favorable procedure.

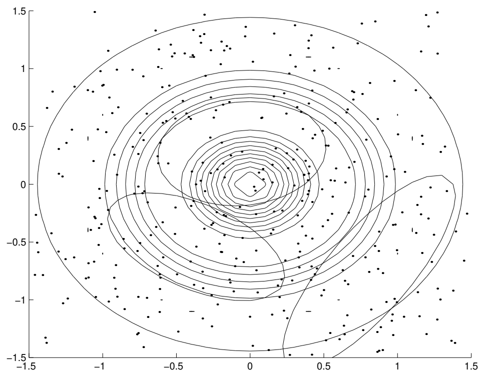

Figure 6. Locally estimated kernel shapes at the observations $x_1 = (1, -0.0562, 0.2814)'$, $x_2 = (1, 0.8063, -0.7584)'$, and $x_3 = (1, -0.3664, -0.5401)'$. We used 500 observations to estimate the kernel shapes using an equivalent sample fraction of $\rho = 8.33\%$. The three ellipsoids correspond to 95% contours of the local weighting kernels.

7.2. Projection Pursuit Regression

There exists an interesting connection between kernel shaping and alternative regression methods based on linear projections of x onto a subspace of the original predictor space. This subspace is typically defined in terms of a set of projection vectors $\beta^1, \ldots, \beta^M$. For example, the projection pursuit regression method by Friedman and Stuetzle [6] is concerned with models of the form

$$\hat{f}(x) \equiv \sum_{i=1}^{M} \phi_i(x'\beta^i), \tag{14}$$

where the $\phi_i(\cdot)$ are univariate kernel smoothers. More generally, Li [7] considers the model

$$\hat{f}(x) \equiv \chi(x'\beta^1, \ldots, x'\beta^M) \tag{15}$$

using "sliced inverse regression". In both cases, the authors suggest numerical procedures to identify projection vectors $\beta^1, \ldots, \beta^M$ that maximize the fit of the training data. This problem is closely related to kernel shaping. Specifically, we may think of (14) and (15) as multivariate kernel smoothers where the shape matrix is of the form $L = (\beta^1, \ldots, \beta^M)$, and where the following parametrization is used for Ω:

$$\Omega \equiv \lambda \cdot LL'. \tag{16}$$

That is, we effectively project x onto $\beta^1, \ldots, \beta^M$ and then evaluate the weighting function in the projected space. This gives a model of the form (15) if local averaging is used as a smoothing method. For a local linear estimate, we effectively obtain a varying coefficient model of the form

$$\hat{f}(x) \equiv x'\gamma(x'L), \tag{17}$$

where the coefficient vector γ depends on the weighting in the projected space.

The correspondence between kernel smoothers and projection methods suggests to combine these methods in practice. In particular, kernel shaping could be used to compute the optimal projection vectors in (14) and (15). (For this purpose, we need to modify the computation of the error gradient $\nabla\mathcal{E}$ in Section 5 to account for (16).) This approach appears particularly appealing if the dimensionality M of the projection space is unknown, because for kernel shaping the correct value of l need not be specified in advance. This was demonstrated in our toy examples in Sections 6.1 and 6.2, where the correct dimensionality and orientation of the subspace could be inferred from the eigenvalues and eigenvectors of $\hat{\Omega}$. Alternatively, we may first carry out a "pilot estimation" using a full-rank L to determine M, and subsequently estimate the values of the projection vectors using projection pursuit regression or sliced inverse regression.

Acknowledgments

The work of Dirk Ormoneit was supported by a grant of the Deutsche Forschungsgemeinschaft (DFG) as part of its post-doctoral program. Trevor Hastie was partially supported by grant DMS-9803645 from the National Science Foundation and grant ROI-CA-72028-01 from the National Institutes of Health. Carrie Grimes pointed us to misleading formulations in earlier drafts of this work.

Appendix

A.1. Derivative Matrices

The elements of the tree-way array V in equation (9) are

$$V^t_{r,i,j} \equiv (y_r - e_1'\hat{\gamma}(x_t,\Omega^t))\frac{\partial}{\partial \Omega^t_{i,j}}k(x_r,x_t). \tag{A.1}$$

Here the last term corresponds the elements of the matrix

$$\frac{\partial}{\partial \Omega^t}k(x_r,x_t) = \frac{1}{2}k(x_r,x_t)\left[-(x_r-x_t)(x_r-x_t)' + X^{t\prime}W^tX^t\right]. \tag{A.2}$$

With regard to the transformed gradient (10), an application of the chain rule to (9) gives that

$$\frac{\partial}{\partial L}\hat{f}(x_t;\Omega) = 2\lambda\frac{\partial}{\partial \Omega^t}\hat{f}(x_t;\Omega^t)L + vec(\frac{\partial}{\partial \Omega}\hat{f}(x_t;\Omega))'vec(LL'+I)\frac{\partial \lambda}{\partial L}. \tag{A.3}$$

By applying the same transformation to the entropy level H instead of $\hat{f}(x_t;\Omega)$, and noting that $\frac{\partial H}{\partial L}=0$ by the definition of λ, we obtain

$$\frac{\partial \lambda}{\partial L} = -2\lambda\frac{\frac{\partial H}{\partial \Omega}L}{vec(\frac{\partial H}{\partial \Omega})'vec(LL'+I)}. \tag{A.4}$$

Substituting back into (A.3) gives (10); here the scalar ξ is defined according to

$$\xi = \frac{vec(\frac{\partial}{\partial \Omega}\hat{f}(x_t;\Omega))'vec(LL'+I)}{vec(\frac{\partial H}{\partial \Omega})'vec(LL'+I)}. \tag{A.5}$$

A.2. The Scaling Algorithm

We noted in Section 5 that one disadvantage of the entropic neighborhood method is that the scaling factor λ in (4) has to be computed numerically after each gradient descent step for the optimization of L. More specifically, recall that λ is defined as the solution of

$$g(\lambda) \equiv H(\lambda(LL'+I)) - \log(\rho T) = 0. \tag{A.6}$$

For an in-depth discussion of numerical solution methods for nonlinear equation systems such as (A.6), see [4]. With regard to our application, it turns out that the solution of (A.6) is a very time-critical operation of our algorithm. Therefore, rather then using a standard bisection search, we suggest to use Newton's method for this purpose:

$$\lambda^{(s+1)} := \lambda^{(s)} - \frac{g(\lambda^{(s)})}{g'(\lambda^{(s)})}. \tag{A.7}$$

The derivative of $g(\lambda)$ is

$$g'(\lambda) = -\sum_{t=1}^{T} [k(x_t, x_0) \log k(x_t, x_0) + H] \left[-\frac{1}{2} \bar{x}_t' (LL' + I) \bar{x}_t \right] \tag{A.8}$$

where $\bar{x}_t \equiv x_t - x_0$. It is important to note that the convergence of (A.7) depends crucially on the absolute value of $g'(\lambda^{(s)})$. In particular, $g(\lambda)$ can be non-concave and non-convex for small bandwidths, and $g'(\lambda)$ can take on values very close to zero. Thus, (A.7) may fail to converge in some situations. The standard way to modify (A.7) to give a globally convergent algorithm is by carrying our a backtracking line-search [4]. That is, rather than automatically accepting the new $\lambda^{(s+1)}$, we first compare the new and the old function value. In detail, $\lambda^{(s+1)}$ is accepted only if

$$g(\lambda^{(s+1)}) g(\lambda^{(s)}) \geq 0 \text{ OR } \left(g(\lambda^{(s+1)}) g(\lambda^{(s)}) < 0 \text{ AND } |g(\lambda^{(s)})| < 0.9 |g(\lambda^{(s)})| \right).$$

If $\lambda^{(s+1)}$ is rejected, we use $\lambda^{(s+1)} := \frac{\lambda^{(s)} + \lambda^{(s+1)}}{2}$ as the next trial value. This backtracking step is repeated until $\lambda^{(s+1)}$ is sufficiently close to $\lambda^{(s)}$ to be accepted. Note that the backtracking restriction essentially prevents situations where λ runs into a cycle. In our experiments, we found that (A.7) usually converged in very few iterations without any backtracking steps.

References

1. C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
2. C. Blake, E. Koegh, and C. J. Merz. UCI Repository of machine learning databases. Technical report, Department of Information and Computer Science, University of California, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.
3. W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.
4. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics. SIAM, 1996.
5. J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications.* Chapman & Hall, 1996.
6. J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Assiciation*, 76(376):817–823, 1981.
7. Ker-Chau Li. Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–342, 1991. With discussion.
8. D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *Annals of Statistics*, 22(3):1346–1370, 1994.