Locally Weighted Learning

Christopher G. Atkeson[*], Andrew W. Moore[†], and Stefan Schaal[‡]
College of Computing, Georgia Institute of Technology[*‡]
801 Atlantic Drive, Atlanta, GA 30332-0280
cga@cc.gatech.edu, sschaal@cc.gatech.edu
http://www.cc.gatech.edu/fac/Chris.Atkeson
http://www.cc.gatech.edu/fac/Stefan.Schaal
Carnegie Mellon University[†]
5000 Forbes Ave, Pittsburgh, PA 15213
awm@cs.cmu.edu
http://www.cs.cmu.edu/~awm/hp.html
ATR Human Information Processing Research Laboratories[‡]
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

October 12, 1996 at 10:21

Abstract

This paper surveys locally weighted learning, a form of lazy learning and memory-based learning, and focuses on locally weighted linear regression. The survey discusses distance functions, smoothing parameters, weighting functions, local model structures, regularization of the estimates and bias, assessing predictions, handling noisy data and outliers, improving the quality of predictions by tuning fit parameters, interference between old and new data, implementing locally weighted learning efficiently, and applications of locally weighted learning. A companion paper surveys how locally weighted learning can be used in robot learning and control.

Keywords: locally weighted regression, LOESS, LWR, lazy learning, memory-based learning, least commitment learning, distance functions, smoothing parameters, weighting functions, global tuning, local tuning, interference.

1 Introduction

Lazy learning methods defer processing of training data until a query needs to be answered. This usually involves storing the training data in memory, and finding relevant data in the database to answer a particular query. This type of learning is also referred to as *memory-based* learning. Relevance is often measured using a distance function, with nearby points having high relevance. One form of lazy learning finds a set of nearest

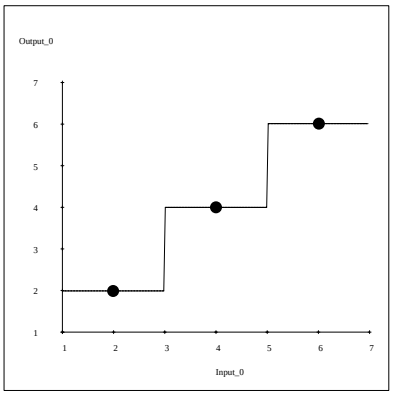
neighbors and selects or votes on the predictions made by each of the stored points. This paper surveys another form of lazy learning, *locally weighted learning,* that uses locally weighted training to average, interpolate between, extrapolate from, or otherwise combine training data (Vapnik, 1992; Bottou and Vapnik, 1992; Vapnik and Bottou, 1993).

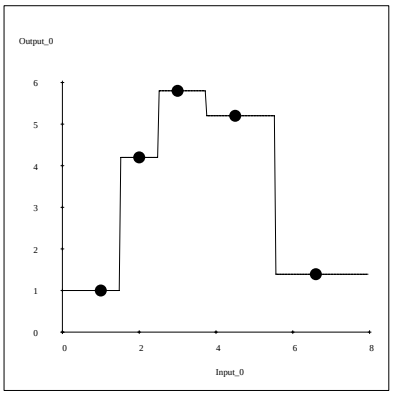
In most learning methods a single global model is used to fit all of the training data. Since the query to be answered is known during processing of training data, training query-specific local models is possible in lazy learning. Local models attempt to fit the training data only in a region around the location of the query (the query point). Examples of types of local models include nearest neighbor, weighted average, and locally weighted regression (Figure 1). Each of these local models combine points near a query point to estimate the appropriate output. *Nearest neighbor* local models simply choose the closest point and use its output value. *Weighted average* local models average the outputs of nearby points, inversely weighted by their distance to the query point. *Locally weighted regression* fits a surface to nearby points using a distance weighted regression.

Weighted averages and locally weighted regression will be discussed in the following sections, and our survey focuses on locally weighted linear regression. The core of the survey discusses distance functions, smoothing parameters, weighting functions, and local model structures. Among the lessons learned from research on locally weighted learning are that practical implementations require dealing with locally inadequate amounts of training data, regularization of the estimates by deliberate introduction of bias, methods for predicting prediction quality, filtering of noise and identifying outliers, automatic tuning of the learning algorithm's parameters to specific tasks or data sets, and efficient implementation techniques. Our motivation for exploring locally weighted learning techniques came from their suitability for real time online robot learning because of their fast incremental learning and their avoidance of negative interference between old and new training data. We provide an example of interference to clarify this point. We briefly survey published applications of locally weighted learning. A companion paper (Atkeson et al., 1996) surveys how locally weighted learning can be used in robot learning and control. This review is augmented by a Web page (Atkeson, 1996).

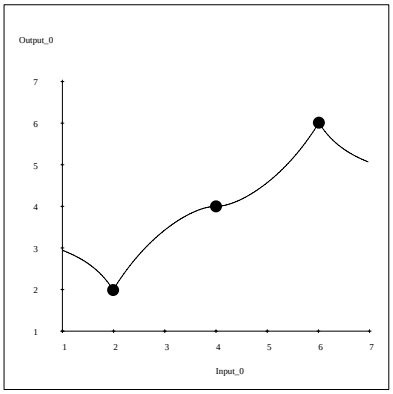
This review emphasizes a statistical view of learning, in which function approximation plays the central role. In order to be concrete, the review focuses on a narrow problem formulation, in which training data consists of input vectors of specific attribute values and the corresponding output values. Both the input and output values are assumed to be continuous. Alternative approaches for this problem formulation include other statistical nonparametric regression techniques, multi-layer sigmoidal neural networks, radial basis functions, regression trees, projection pursuit regression, and global regression techniques. The discussion section (Section 16) argues that locally weighted learning can be applied in a much broader context. Global learning methods can often be improved by localizing them using locally weighted training criteria (Vapnik, 1992; Bottou and Vapnik, 1992; Vapnik and Bottou, 1993). Although this survey emphasizes regression applications (real valued outputs), the discussion section outlines how these techniques have been applied in classification (discrete outputs). We conclude with a short discussion of future research directions.

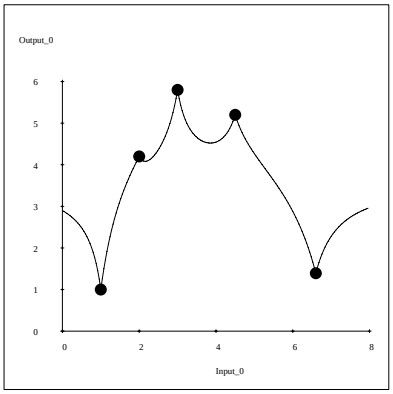
Nearest neighbor



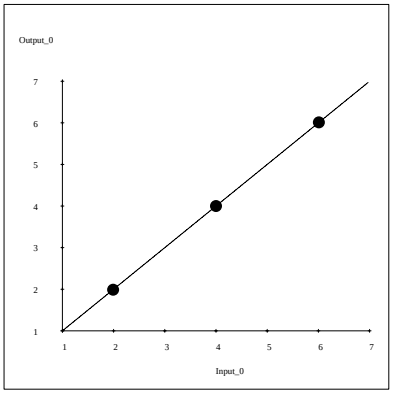


Weighted average





Locally weighted regression



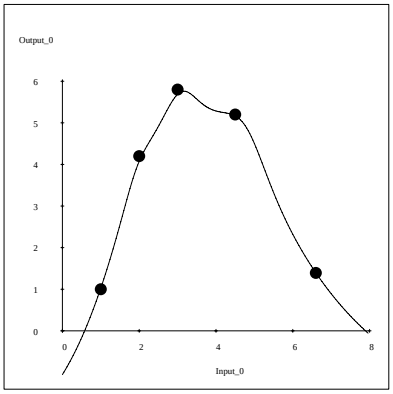


Figure 1: Fits using different types of local models for three and five data points.

Notation

In this paper scalars are represented by italic lower case letters (y). Column vectors are represented as boldface lower case letters ($\mathbf{x}$) and row vectors are represented as the column vectors transposed ($\mathbf{x}^{\mathrm{T}}$). Matrices are represented by bold face upper case letters ($\mathbf{X}$).

2 Distance Weighted Averaging

To illustrate how locally weighted learning using a distance function is applied, we will first consider a simple example, *distance weighted averaging*. This will turn out to be a form of locally weighted regression in which the local model is a constant. A prediction $\hat{y}$ can be based on an average of n training values $\{y_1, y_2, ..., y_n\}$:

$$\hat{y} = \frac{\sum y_i}{n} \tag{1}$$

This estimate minimizes a criterion:

$$C = \sum_i (\hat{y} - y_i)^2 \tag{2}$$

In the case where the training values $\{y_1, y_2, ..., y_n\}$ are taken under different conditions $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, it makes sense to emphasize data that is similar to the query $\mathbf{q}$ and deemphasize dissimilar data, rather than treat all the training data equally. We can do this in two equivalent ways: weighting the data directly or weighting the error criterion used to choose $\hat{y}$.

2.1 Weighting the Data Directly

Weighting the data can be viewed as replicating relevant instances and discarding irrelevant instances. In our case an instance is represented as a data point $(\mathbf{x}, y)$. Relevance is measured by calculating a *distance* $d(\mathbf{x}_i, \mathbf{q})$ between the query point $\mathbf{q}$ and each data point input vector $\mathbf{x}_i$. A typical distance function is the Euclidean distance ($\mathbf{x}_i$ is the ith input vector, while $\mathbf{x}_j$ is the jth component of the vector $\mathbf{x}$):

$$d_{\mathrm{E}}(\mathbf{x}, \mathbf{q}) = \sqrt{\sum_j (\mathbf{x}_j - \mathbf{q}_j)^2} = \sqrt{(\mathbf{x} - \mathbf{q})^{\mathrm{T}}(\mathbf{x} - \mathbf{q})} \tag{3}$$

A *weighting function* or *kernel function* $K()$ is used to calculate a weight for that data point from the distance. A typical weighting function is a Gaussian (Figure 8):

$$K(d) = e^{-d^2} \tag{4}$$

The weight is then used in a weighted average:

$$\hat{y}(\mathbf{q}) = \frac{\sum y_i K(d(\mathbf{x}_i, \mathbf{q}))}{\sum K(d(\mathbf{x}_i, \mathbf{q}))} \tag{5}$$

Note that the estimate $\hat{y}$ depends on the location of the query point $\mathbf{q}$.

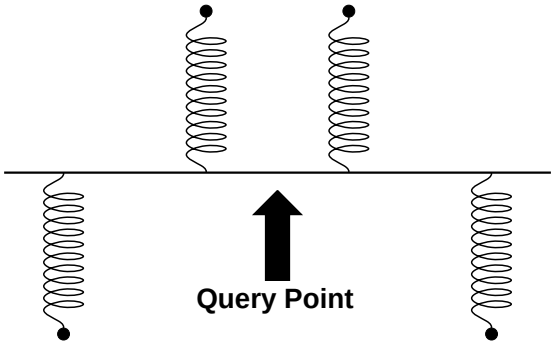


Figure 2: Unweighted averaging using springs.

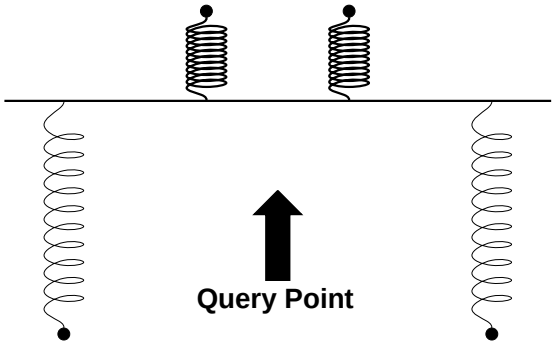


Figure 3: Locally weighted averaging using springs.

2.2 Weighting the Error Criterion

We are trying to find the best estimate for the outputs y_i, using a local model that is a constant, $\hat{y}$. Distance weighting the error criterion corresponds to requiring the local model to fit nearby points well, with less concern for distant points:

$$C(\mathbf{q}) = \sum_{i=1}^{n} \left[(\hat{y} - y_i)^2 K(d(\mathbf{x}_i, \mathbf{q})) \right] \tag{6}$$

The best estimate $\hat{y}(\mathbf{q})$ will minimize the cost $C(\mathbf{q})$. For that value of $\hat{y}$, $\frac{\partial C}{\partial \hat{y}} = 0$. This is achieved by the $\hat{y}$ given in Equation 5, and so in this case weighting the error criterion and weighting the data are equivalent. Note that both the criterion $C(\mathbf{q})$ and the estimate $y(\mathbf{q})$ depend on the location of the query point $\mathbf{q}$.

This process has a physical interpretation. Figures 2 and 3 show the data points (black dots), which are fixed in space, pulling on a horizontal line (the constant model) with springs. The strength of the springs are equal in the unweighted case, and the position of the horizontal line minimizes the sum of the stored energy in the springs (Equation 2). We will ignore a factor of $\frac{1}{2}$ in all our energy calculations to simplify notation. In the weighted case, the springs are not equal, and the spring constant of each spring is given by $K(d(\mathbf{x}_i, \mathbf{q}))$. The stored energy in the springs in this case is C of Equation 6, which is minimized by the physical process. Note that the locally weighted average emphasizes points close to the query point, and produces an answer (the height of the horizontal line) that is closer to the height of points near the query point than the unweighted case.

2.3 The Distance Weighted Averaging Literature

In statistics the approach of fitting constants using a locally weighted training criterion is known as *kernel regression* and has a vast literature (Härdle, 1990; Wand and Jones, 1994). Nadaraya (1964) and Watson (1964) proposed using a weighted average of a set of nearest neighbors for regression. The approach was also independently reinvented in computer graphics (Shepard, 1968). Specht (1991) describes a memory-based neural network approach based on a probabilistic model that motivates using weighted averaging

as the local model for regression. Connell and Utgoff (1987), Kibler et al. (1989) and Aha (1990) have applied weighted averaging to artificial intelligence problems.

3 Locally Weighted Regression

In *locally weighted regression* (LWR) local models are fit to nearby data. As described later in this section, this can be derived by either weighting the training criterion for the local model (in the general case) or by directly weighting the data (in the case that the local model is linear in the unknown parameters). LWR is derived from standard regression procedures for global models. We will start our exploration of LWR by reviewing regression procedures for global models.

3.1 Nonlinear Local Models

3.1.1 Nonlinear Global Models

A general global model can be trained to minimize the following unweighted training criterion:

$$C = \sum_i L(f(\mathbf{x}_i, \beta), y_i) \tag{7}$$

where the y_i are the output values corresponding to the input vectors $\mathbf{x}_i$, β is the parameter vector for the nonlinear model $\hat{y}_i = f(\mathbf{x}_i, \beta)$, and $L(\hat{y}_i, y_i)$ is a general loss function for predicting $\hat{y}_i$ when the training data is y_i. For example, if the model were a neural net, then β would be a vector of the synaptic weights. Often the least squares criterion is used for the loss function ($L(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$), leading to the training criterion:

$$C = \sum_i (f(\mathbf{x}_i, \beta) - y_i)^2 \tag{8}$$

Sometimes no values of the parameters of a global model can provide a good approximation of the true function. There are two approaches to this problem. First, we could use a larger, more complex global model and hope that it can approximate the data sufficiently. The second approach, which we discuss here, is to fit the simple model to local patches instead of the whole region of interest.

3.1.2 A Training Criterion For Nonlinear Local Models

The data set can be tailored to the query point by emphasizing nearby points in the regression. We can do this by weighting the training criterion:

$$C(\mathbf{q}) = \sum_i [L(f(\mathbf{x}_i, \beta), y_i) K(d(\mathbf{x}_i, \mathbf{q}))] \tag{9}$$

where $K()$ is the weighting or kernel function and $d(\mathbf{x}_i, \mathbf{q})$ is the distance between the data point $\mathbf{x}_i$ and the query $\mathbf{q}$. Using this training criterion, $f(\mathbf{x}, \beta(\mathbf{q}))$ now becomes a *local* model, and can have a different set of parameters $\beta(\mathbf{q})$ for each query point $\mathbf{q}$.

3.2 Linear Local Models

Given that we are using local models, it seems advantageous to keep them simple, and to keep the training criterion simple as well. This leads us to explore local models that are linear in the unknown parameters, and to use the least squares training criterion. We derive least squares training algorithms for linear local models from regression procedures for linear global models.

3.2.1 Linear Global Models

A global model that is linear in the parameters β can be expressed as (Myers, 1990):

$$\mathbf{x}_i^{\mathrm{T}}\beta = y_i \tag{10}$$

In what follows we will assume that the constant 1 has been appended to all the input vectors $\mathbf{x}_i$ to include a constant term in the regression. The training examples can be collected in a matrix equation:

$$\mathbf{X}\beta = \mathbf{y} \tag{11}$$

where $\mathbf{X}$ is a matrix whose ith row is $\mathbf{x}_i^{\mathrm{T}}$ and $\mathbf{y}$ is a vector whose ith element is y_i. Thus, the dimensionality of $\mathbf{X}$ is $n \times d$ where n is the number of data points and d is the dimensionality of $\mathbf{x}$. Estimating the parameters β using an unweighted regression minimizes the criterion

$$C = \sum_i (\mathbf{x}_i^{\mathrm{T}}\beta - y_i)^2 \tag{12}$$

by solving the normal equations

$$\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)\beta = \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{13}$$

for β:

$$\beta = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{14}$$

Inverting the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is not the numerically best way to solve the normal equations from the point of view of efficiency or accuracy, and usually other matrix techniques are used to solve Equation 13 (Press et al., 1988).

3.2.2 Weighting the Criterion: A Physical Interpretation

In fitting a line or plane to a set of points, unweighted regression gives distant points equal influence with nearby points on the ultimate answer to the query, for equally spaced data. The linear local model can be specialized to the query by emphasizing nearby points. As with the distance weighted average example we can either weight the error criterion that is minimized, or weight the data directly. The two approaches are equivalent for planar local models. Weighting the criterion is done in the following way

$$C(\mathbf{q}) = \sum_i \left[(\mathbf{x}_i^{\mathrm{T}}\beta - y_i)^2 K(d(\mathbf{x}_i, \mathbf{q}))\right] \tag{15}$$

We again have a physical interpretation for $C(\mathbf{q})$ of Equation 15. Much as thin plate splines minimize a bending energy of a plate and the energy of the constraints pulling

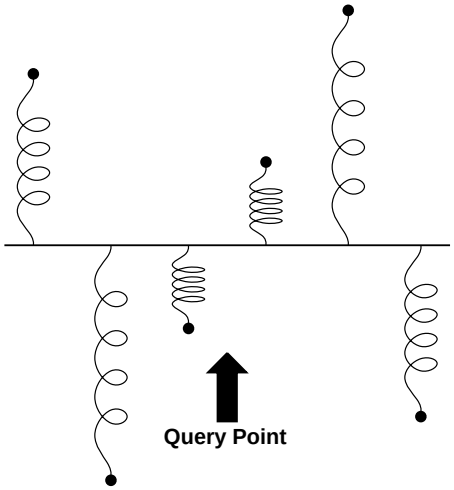


Figure 4: Unweighted springs.

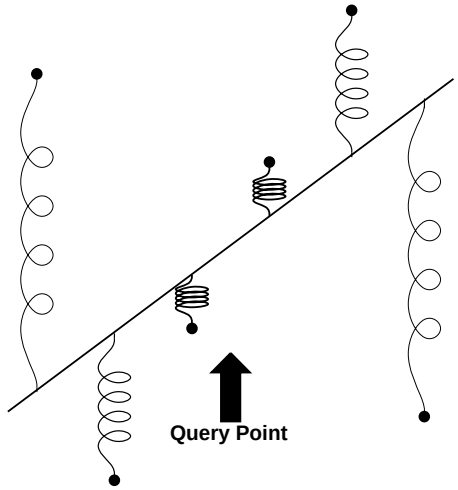


Figure 5: Weighted springs.

on the plate, locally weighted regression can also be interpreted as a physical process. In LWR with a planar local model, the line in Figures 2 and 3 can now rotate as well as translate. The springs are forced to remain oriented vertically, rather than move to the smallest distance between the data points and the line. Figure 4 shows the fit (the line) produced by equally strong springs to a set of data points (the black dots), minimizing the criterion of Equation 12. Figure 5 shows what happens to the fit as the springs nearer to the query point are strengthened and the springs further away are weakened. The strengths of the springs are given by $K(d(\mathbf{x}_i, \mathbf{q}))$, and the fit minimizes the criterion of Equation 15.

3.2.3 Direct Data Weighting

Our version of directly weighting the data involves the following steps. For computational and analytical simplicity the origin of the input data is first shifted by subtracting the query point from each data point (making the query point $\mathbf{q} = (0, ..., 0, 1)^{\mathrm{T}}$, where the 1 is appended for the constant term in the regression). A distance is calculated from each of the stored data points to the query point $\mathbf{q}$. The weight for each stored data point is the square root of the kernel function used in Equation 15, to simplify notation later:

$$w_i = \sqrt{K(d(\mathbf{x}_i, \mathbf{q}))} \tag{16}$$

Each row i of $\mathbf{X}$ and $\mathbf{y}$ is multiplied by the corresponding weight w_i creating new variables $\mathbf{Z}$ and $\mathbf{v}$. This can be done using matrix notation by creating a diagonal matrix $\mathbf{W}$ with diagonal elements $\mathbf{W}_{ii} = w_i$ and zeros elsewhere and multiplying $\mathbf{W}$ times the original variables.

$$\mathbf{z}_i = w_i \mathbf{x}_i \tag{17}$$

$$\mathbf{Z} = \mathbf{W}\mathbf{X} \tag{18}$$

and

$$v_i = w_i y_i \tag{19}$$

$$\mathbf{v} = \mathbf{W}\mathbf{y} \tag{20}$$

Equation 13 is solved for β using the new variables:

$$\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right)\beta = \mathbf{Z}^{\mathrm{T}}\mathbf{v} \tag{21}$$

Formally, this gives us an estimator of the form

$$\hat{y}(\mathbf{q}) = \mathbf{q}^{\mathrm{T}}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{v} \tag{22}$$

3.3 The Relationship of Kernel Regression and Locally Weighted Regression

For data distributed on a regular grid away from any boundary locally weighted regression and kernel regression are equivalent (Lejeune, 1985; Müller, 1987). However, for irregular data distributions there is a significant difference, and LWR has many advantages over kernel regression (Hastie and Loader, 1993; Jones et al., 1994). LWR with a planar local model is often preferred over kernel smoothing because it exactly reproduces a line (with any data distribution). The failure to reproduce a line, or any function used to generate the training data, indicates the bias of a function approximation method. LWR methods with a planar local model will fail to reproduce a quadratic function, reflecting the bias due to the planar local model. LWR methods with a quadratic local model will fail to reproduce a cubic function, and so on.

3.4 The Locally Weighted Regression Literature

Cleveland and Loader (1994a,c), Fan (1995) and Fan and Gijbels (1996) review the history of locally weighted regression and discuss current research trends. Barnhill (1977) and Sabin (1980) survey the use of distance weighted nearest neighbor interpolators to fit surfaces to arbitrarily spaced points, and Eubank (1988) surveys their use in nonparametric regression. Lancaster and Šalkauskas (1986) refer to nearest neighbor approaches as "moving least squares" and survey their use in fitting surfaces to data. Härdle (1990) surveys kernel and LWR approaches to nonparametric regression. Farmer and Sidorowich (1987, 1988a,b) survey the use of nearest neighbor and local model approaches in modeling chaotic dynamic systems.

Local models (often polynomials) have been used for over a century to smooth regularly sampled time series and interpolate and extrapolate from data arranged on rectangular grids. Crain and Bhattacharyya (1967), Falconer (1971) and McLain (1974) suggested using a weighted regression on irregularly spaced data to fit a local polynomial model at each point a function evaluation was desired. All of the available data points were used. Each data point was weighted by a function of its distance to the desired point in the regression. Many authors have suggested fitting a polynomial surface only to nearby points also using distance weighted regression (McIntyre et al., 1968; Pelto et al., 1968; Legg and Brent, 1969; Palmer, 1969; Walters, 1969; Lodwick and Whittle, 1970; Stone, 1975, 1977; Benedetti, 1977; Tukey, 1977; Franke and Nielson, 1980; Friedman, 1984) Cleveland (1979) proposed using robust regression procedures to eliminate outlying or

erroneous points in the regression process. Programs implementing a refined version of this approach (LOCFIT and LOESS) are available directly and also as part of the S+ package (Cleveland et al., 1992; Cleveland and Loader, 1994a,b,c). Katkovnik (1979) also developed a robust locally weighted smoothing procedure. Cleveland et al. (1988) analyzes the statistical properties of the LOESS algorithm and Cleveland and Devlin (1988) and Cleveland et al. (1993) show examples of its use. Stone (1977), Devroye (1981), Lancaster (1979), Lancaster and Šalkauskas (1981), Cheng (1984), Li (1984), Tsybakov (1986) and Farwig (1987) provide analyses of LWR approaches. Stone (1980, 1982) shows that LWR has an optimal rate of convergence in a minimax sense. Fan (1992) shows that local linear regression smoothers are the best smoothers, in that they are the asymptotic minimax linear smoother and have a high asymptotic efficiency (which can be 100% with a suitable choice of kernel and bandwidth) among all possible linear smoothers, including those produced by kernel, orthogonal series, and spline methods, when the unknown regression function is in the class of functions having bounded second derivatives. Fan (1993) extends this result to show that LWR has a high minimax efficiency among all possible estimators, including nonlinear smoothers such as median regression. Fan (1992), Fan and Gijbels (1992), Hastie and Loader (1993) and Jones et al. (1994) show that LWR handles a wide range of data distributions and avoids boundary and cluster effects. Ruppert and Wand (1994) derive asymptotic bias and variance formulas for multivariate LWR, while Cleveland and Loader (1994c) argue that asymptotic results have limited practical relevance. Fan and Gijbels (1992) explore the use of a variable bandwidth locally weighted regression. Vapnik and Bottou (1993) give error bounds for local learning algorithms.

Locally weighted regression was introduced into the domain of machine learning and robot learning by Atkeson (Atkeson and Reinkensmeyer, 1988, 1989; Atkeson, 1990, 1992), who also explored techniques for detecting irrelevant features, and Zografski (Zografski, 1989, 1991, 1992; Zografski and Durrani, 1995). Atkeson and Schaal (1995) explore locally weighted learning from the point of view of neural networks. Dietterich et al. (1994) report on a recent workshop on memory-based learning, including locally weighted learning.

4 Distance Functions

Locally weighted learning is critically dependent on the distance function. There are many different approaches to defining a distance function, and this section briefly surveys them. Distance functions in locally weighted learning do not need to satisfy the formal mathematical requirements for a distance metric. The relative importance of the input dimensions in generating the distance measurement depends on how the inputs are scaled (i.e., how much they are stretched or squashed). We use the term *scaling* for this purpose having reserved the term *weight* for the contribution of individual points (not dimensions) in a regression. We refer to the scaling factors as m_j in this paper. There are many ways to define and use distance functions (Scott, 1992):

- **Global distance functions.** The same distance function is used at all parts of the input space.

- **Query-based local distance functions:** The form of $d()$ or the distance function parameters are set on each query by an optimization process that typically minimizes cross validation error or a related criterion. This approach is referred to as a *uniform metric* by Stanfill (1987) and is discussed in Stanfill and Waltz (1986), Hastie and Tibshirani (1994) and Friedman (1994).

- **Point-based local distance functions:** Each stored data point has associated with it a distance function and the values of corresponding parameters. The training criterion uses a different $d_i()$ for each point $\mathbf{x}_i$:

$$C(\mathbf{q}) = \sum_i \left[(f(\mathbf{x}_i, \beta) - y_i)^2 K(d_i(\mathbf{x}_i, \mathbf{q})) \right] \tag{23}$$

  The $d_i()$ can be selected either by a direct computation or by minimizing cross validation error. Frequently, the $d_i()$ are chosen in advance of the queries and are stored with the data points. This approach is referred to as a *variable metric* by Stanfill (1987). For classifiers, one version of a point-based local distance function is to have a different distance function for each class (Waltz, 1987; Aha and McNulty, 1989; Aha, 1989, 1990). Aha and Goldstone (1990, 1992) explore the use of point-based distance functions by human subjects.

Distance functions can be asymmetric and nonlinear, so that a distance along a particular dimension can depend on whether the query point's value for the dimension is larger or smaller than the stored point's value for that dimension (Medin and Shoben, 1988). The distance along a dimension can also depend on the values being compared (Nosofsky et al., 1989).

4.1 Feature Scaling

Altering the distance function can serve two purposes. If the feature scaling factors m_j are all nonzero, the input space is warped or distorted, which might lead to more accurate predictions. If some of the scaling factors are set to zero, those dimensions are ignored by the distance function, and the local model becomes global in those directions. Zeroing feature scaling factors can be used as a tool to combat the curse of dimensionality by reducing the locality of the function approximation process in this way.

Note that a feature scaling factor of zero does not mean the local model ignores that feature in locally weighted learning. Instead, all points aligned along that direction get the same weight, and the feature can affect the output of the local model. For example, fitting a global model using all features is equivalent to setting all feature scaling factors to zero and fitting the same model as a local model. Local model feature selection is a separate process from distance function feature scaling. Ignoring features using ridge regression, dimensionality reduction of the entire modeling process, and algorithms for feature scaling and selection are discussed in later sections.

Stanfill and Waltz (1986) describe a variant of feature selection ("predictor restriction") in which the scaling factor for a feature becomes so large that any difference from the query in that dimension causes the training point to be ignored. They also describe using an initial prediction of the output in an augmented distance function to select training data with similar or equal outputs ("goal restriction") (Jabbour et al., 1987).

4.2 Distance Functions For Continuous Inputs

The functions discussed in this section are especially appropriate for ordered (vs. categorical, symbolic, or nominal) input values, which are either continuous or an ordered set of discrete values.

- **Unweighted Euclidean distance:**

$$d_{\mathrm{E}}(\mathbf{x},\mathbf{q}) = \sqrt{\sum_j (\mathbf{x}_j - \mathbf{q}_j)^2} = \sqrt{(\mathbf{x}-\mathbf{q})^{\mathrm{T}}(\mathbf{x}-\mathbf{q})} \tag{24}$$

- **Diagonally weighted Euclidean distance:**

$$d_m(\mathbf{x},\mathbf{q}) = \sqrt{\sum_j (m_j(\mathbf{x}_j - \mathbf{q}_j))^2} = \sqrt{(\mathbf{x}-\mathbf{q})^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{M}(\mathbf{x}-\mathbf{q})} = d_{\mathrm{E}}(\mathbf{Mx},\mathbf{Mq}) \tag{25}$$

  where m_j is the feature scaling factor for the jth dimension and $\mathbf{M}$ is a diagonal matrix with $\mathbf{M}_{jj} = m_j$.

- **Fully weighted Euclidean distance:**

$$d_{\mathbf{M}}(\mathbf{x},\mathbf{q}) = \sqrt{(\mathbf{x}-\mathbf{q})^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\mathbf{M}(\mathbf{x}-\mathbf{q})} = d_{\mathrm{E}}(\mathbf{Mx},\mathbf{Mq}) \tag{26}$$

  where $\mathbf{M}$ is no longer diagonal but can be arbitrary. This is also known as the Mahalanobis distance (Tou and Gonzalez, 1974; Weisberg, 1985).

- **Unweighted L_p norm (Minkowski metric):**

$$d_p(\mathbf{x},\mathbf{q}) = \left(\sum_i |\mathbf{x}_i - \mathbf{q}_i|^p\right)^{\frac{1}{p}} \tag{27}$$

- **Diagonally weighted and fully weighted L_p norm:** The weighted L_p norm is $d_p(\mathbf{Mx},\mathbf{Mq})$.

A diagonal distance function matrix $\mathbf{M}$ (1 coefficient for each dimension) can make a radially symmetric scaling function into an axis parallel ellipse (Figure 6 shows ellipses with the axes of symmetry aligned with the coordinate axes). Figure 7 shows an example of how a full distance function matrix $\mathbf{M}$ with cross terms can arbitrarily orient the ellipse (Ruppert and Wand, 1994; Wand and Jones, 1993). Cleveland and Grosse (1991), Cleveland et al. (1992) and Cleveland (1993a) point out that for distance functions with zero coefficients ($m_i = 0$, an entire column of $\mathbf{M}$ is zero, or $\mathbf{M}$ is singular), the model is global in the corresponding directions. They refer to this as a *conditionally parametric* model.

Fukunaga (1990), James (1985) and Tou and Gonzalez (1974) describe how to choose a distance function matrix to maximize the ratio of the variance between classes to the variance of all the cases in classification. Mohri and Tanaka (1994) extend this approach to symbolic input values. This approach uses an eigenvalue/eigenvector decomposition and can help distinguish relevant attributes from irrelevant attributes and filter out noisy data. This approach is localized by Hastie and Tibshirani (1994). Distance functions for symbolic inputs have been developed and are discussed in Atkeson (1996).

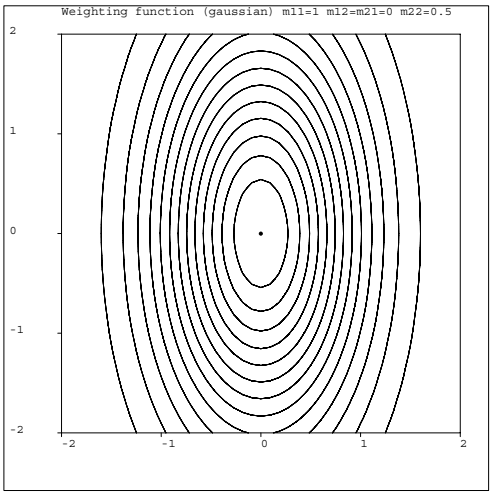


Figure 6: Contours of constant distance from the center with a diagonal $\mathbf{M}$ matrix.

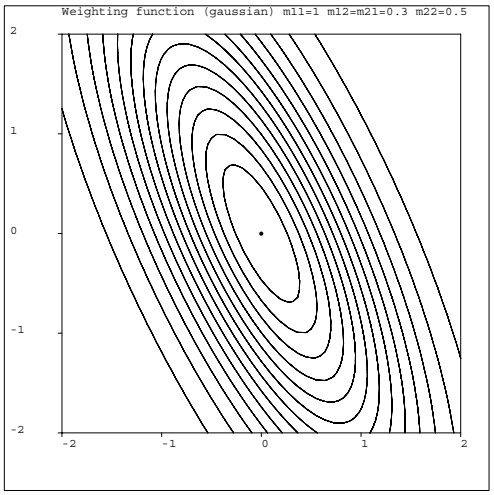


Figure 7: Contours of a constant distance from the center in which the $\mathbf{M}$ matrix has off-diagonal elements.

5 Smoothing Parameters

A smoothing or bandwidth parameter h defines the scale or range over which generalization is performed. There are several ways to use this parameter (Scott, 1992; Cleveland and Loader, 1994c):

- **Fixed bandwidth selection:** h is a constant value (Fan and Marron, 1993), and therefore volumes of data with constant size and shape are used. In this case h can appear implicitly in the distance function as the determinant of $\mathbf{M}$ for fully weighted distance functions ($h = |\mathbf{M}|$) or the magnitude of the vector $\mathbf{m}$ in diagonally weighted distance functions ($h = |\mathbf{m}|$) and/or explicitly in the weighting function:

$$K\left(\frac{d(\mathbf{x}_i, \mathbf{q})}{h}\right) \tag{28}$$

  These parameters, although redundant in the explicit case, provide a convenient way to adjust the radius of the weighting function. The redundancy can be eliminated by requiring the determinant of the scaling factor matrix to be one ($|\mathbf{M}| = 1$), or fixing some element of $\mathbf{M}$.

- **Nearest neighbor bandwidth selection:** h is set to be the distance to the kth nearest data point (Stone, 1977; Cleveland, 1979; Farmer and Sidorowich, 1988a,b; Townshend, 1992; Hastie and Loader, 1993; Fan and Gijbels, 1994; Ge et al., 1994; Næs et al., 1990; Næs and Isaksson, 1992; Wang et al., 1994; Cleveland and Loader, 1994b). The data volume increases and decreases in size according to the density of nearby data. In this case changes in scale of the distance function are canceled by corresponding changes in h, giving a scale invariant weighting pattern to the data. However, h will not cancel changes in distance function coefficients that alter the

shape of the weighting function, and the identity of the kth neighbor can change with distance function shape changes.

- **Global bandwidth selection:** h is set globally by an optimization process that typically minimizes cross validation error over all the data.
- **Query-based local bandwidth selection:** h is set on each query by an optimization process that typically minimizes cross validation error or a related criterion (Vapnik, 1992).
- **Point-based local bandwidth selection:** Each stored data point has associated with it a bandwidth h. The weighted criterion uses a different h_i for each point $\mathbf{x}_i$:

$$C(\mathbf{q}) = \sum_i \left[(f(\mathbf{x}_i, \beta) - y_i)^2 K \left(\frac{d(\mathbf{x}_i, \mathbf{q})}{h_i} \right) \right] \tag{29}$$

  The h_i can be set either by a direct computation or by an optimization process that typically minimizes cross validation error or a related criterion. Typically, the h_i are computed in advance of the queries and are stored with the data points.

Fan and Marron (1994b) argue that a fixed bandwidth is easy to interpret, but of limited use. Cleveland and Loader (1994a) argue in favor of nearest neighbor smoothing over fixed bandwidth smoothing. A fixed bandwidth and a weighting function that goes to zero at a finite distance can have large variance in regions of low data density. This problem is present at edges or between data clusters and gets worse in higher dimensions. In general, fixed bandwidth selection has much larger changes in variance than nearest neighbor bandwidth selection. A fixed bandwidth smoother can also not have any data within its span, leading to undefined estimates (Cleveland and Loader, 1994b). Fan and Marron (1994b) describe three reasons to use variable local bandwidths: to adapt to the data distribution, to adapt for different levels of noise (*heteroscedasticity*), and to adapt to changes in the smoothness or curvature of the function. Fan and Gijbels (1992) argue for point-based in favor of query-based local bandwidth selection, explaining that having a bandwidth associated with each data point will allow rapid or asymmetric changes in the behavior of the data to be accommodated. Section 12 discusses global and local tuning of bandwidths.

6 Weighting Functions

The requirements on a weighting function (also known as a kernel function) are straightforward (Gasser and Müller, 1979; Cleveland and Loader, 1994c; Fedorov et al., 1993). The maximum value of the weighting function should be at zero distance, and the function should decay smoothly as the distance increases. Discontinuities in weighting functions lead to discontinuities in the predictions, since training points cross the discontinuity as the query changes. In general, the smoother the weight function, the smoother the estimated function. Weights that go to infinity when a query equals a stored data point allow exact interpolation of the stored data. Finite weights lead to smoothing of the

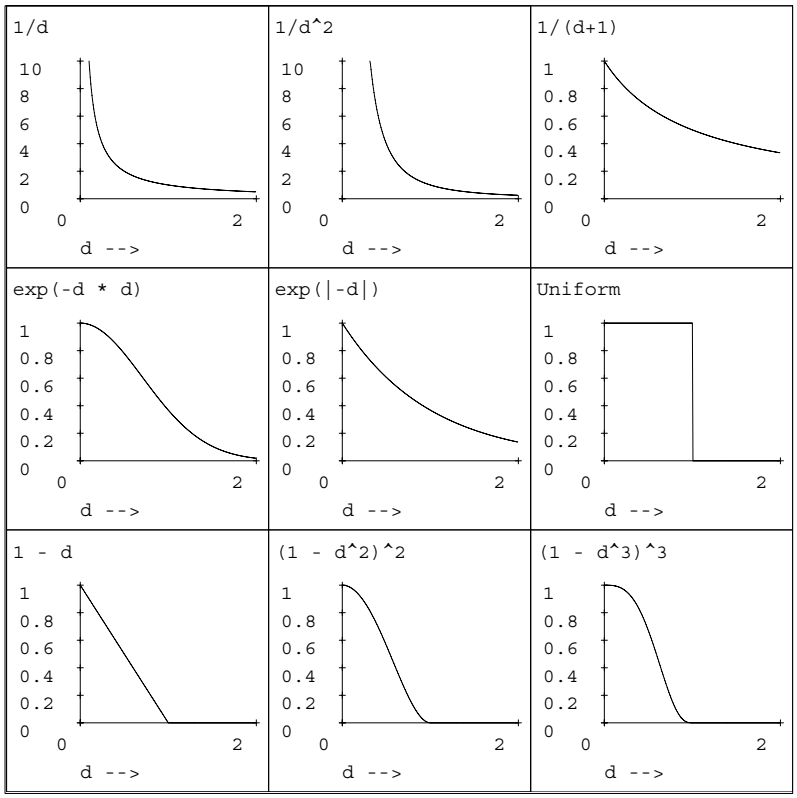


Figure 8: Some of the kernel shapes described in the text.

data. Weight functions that go to zero at a finite distance allow faster implementations, since points further from the query than that distance can be ignored with no error. As mentioned previously, kernels with a fixed finite radius raise the possibility of not having enough or any points within the non-zero area, a possibility that must be handled by the locally weighted learning system. It is not necessary to normalize the kernel function, and the kernel function does not need to be unimodal. The kernel function should always be non-negative, since a negative value would lead to the training process increasing training error in order to decrease the training criterion. The weights (i.e., the square root of the kernel function) can be positive or negative. We have only used non-negative weights. Some of the kernel functions discussed in this section are shown in Figure 8.

A simple weighting function just raises the distance to a negative power (Shepard, 1968; Atkeson, 1992; Ruprecht et al., 1994; Ruprecht and Müller, 1994b). The magnitude of the power determines how local the regression will be (i.e., the rate of dropoff of the weights with distance).

$$K(d) = \frac{1}{d^p} \tag{30}$$

This type of weighting function goes to infinity as the query point approaches a stored data point and forces the locally weighted regression to exactly match that stored point. If the data is noisy, exact interpolation is not desirable, and a weighting scheme with limited magnitude is desired. The *inverse distance* (Wolberg, 1990)

$$K(d) = \frac{1}{1 + d^p} \tag{31}$$

can be used to approximate functions like Equation 30 and the quadratic hyperbola kernel $1/(h^2 + d^2)$ with a well defined value at $d = 0$.

Another smoothing weight function is a *Gaussian kernel* (Deheuvels, 1977; Wand and Schucany, 1990; Schaal and Atkeson, 1994):

$$K(d) = \exp\left(-d^2\right) \tag{32}$$

This kernel also has infinite extent. A related kernel is the *exponential kernel*, which has been used in psychological models (Aha and Goldstone, 1992):

$$K(d) = \exp\left[-|d|\right] \tag{33}$$

These kernels have infinite extent, and can be truncated when they become smaller than a threshold value to ignore data further from a particular radius from the query.

The *quadratic kernel*, also known as the Epanechnikov kernel and the Bartlett-Priestley kernel, is (Epanechnikov, 1969; Lejeune, 1984; Altman, 1992; Hastie and Loader, 1993; Fan and Gijbels, 1995b,a; Fan and Hall, 1994):

$$K(d) = \begin{cases} (1 - d^2) & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

This kernel has finite extent and ignores data further than a radius of 1 from the query when building the local model. Fan and Marron (1993) and Müller (1993) argue that this kernel function is optimal in a mean squared error sense. However, there is a discontinuity in the derivative at $d = 1$, which makes this kernel less attractive in real applications and analytical treatments.

The *tricube kernel* is used by Cleveland (1979), Cleveland and Devlin (1988), Diebold and Nason (1990), LeBaron (1990), Næs et al. (1990), Næs and Isaksson (1992), Wang et al. (1994) and Ge et al. (1994):

$$K(d) = \begin{cases} \left(1 - |d|^3\right)^3 & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

This kernel also has finite extent and a continuous first and second derivative, which means the first and second derivative of the prediction will also be continuous.

For comparison, the *uniform weighting kernel* (or boxcar weighting kernel) is used by Stone (1977), Friedman (1984), Tsybakov (1986) and Müller (1987):

$$K(d) = \begin{cases} 1 & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

and the *triangular kernel* (used in locally weighted median regression) is:

$$K(d) = \begin{cases} 1 - |d| & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

A variant of the triangular kernel is the following (Franke and Nielson, 1980; Ruprecht and Müller, 1993, 1994a; Ruprecht et al., 1994):

$$K(d) = \begin{cases} \frac{1-|d|}{|d|} & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

In general new kernel functions can be created by raising these kernel functions to a power. For example, the *biquadratic kernel* is the square of the quadratic kernel. The power can be non-integral, and also less than one. The triangular, biquadratic, and tricube kernels form a family. Ruprecht and Müller (1994b) generalize the distance function to a point-set metric.

In our view, there is no clear evidence that the choice of weighting function is critical (Scott, 1992; Cleveland and Loader, 1994a,c), however, there are examples where one can show differences (Fedorov et al., 1993). Cleveland and Loader (1994b) criticize the uniform kernel for similar reasons as are used in signal processing and spectrum estimation. Optimal kernels are discussed by Gasser and Müller (1984), Gasser et al. (1985), Scott (1992), Blyth (1993) and Fedorov et al. (1993). Finite extent of the kernel function is useful, but other than that, the literature and our own work have not noted any substantial empirical difference in most cases.

7 Local Model Structures

So far we have discussed only a few kinds of local models, constant and linear local models. There are no limits on what model structure can be used as a local model. Models that are linear in the unknown parameters, such as local polynomials, train faster than more general models. Since a major component of the lookup cost is the training cost, this is an important benefit. Cleveland and Devlin (1988), Atkeson (1992), Farmer and Sidorowich (1988a,b), Cleveland and Loader (1994a), Næs et al. (1990), Næs and Isaksson (1992) and Wang et al. (1994) have applied local quadratic and cubic models, which are analyzed by Ruppert and Wand (1994). Higher order polynomials reduce the bias but increase the variance of the estimates. Locally constant models handle flat regions well, while quadratics and cubics handle areas of high curvature such as peaks and valleys well.

Cleveland and Loader (1994a,c) present an approach to blending polynomial models, where a non-integral model order indicates a weighted blend between two integral model orders. They use cross validation to optimize the local model order on each query.

8 Regularization, Insufficient Data, and Prediction Bias

To uniquely interpolate between and extrapolate from the training data we must express a preference or learning bias. In function approximation that preference is typically expressed as a smoothness criterion to optimize. In the case of locally weighted learning the smoothness constraint is not explicit. However, there are several fit parameters that affect the smoothness of the predicted outputs. The smoothing bandwidth is an important control knob, as is a ridge regression parameter, to be described in the next section. The order of the local model also can serve as a smoothing parameter. The shape of the distance and weighting functions play a secondary role in smoothing the estimates, although in general the number of derivatives with respect to $\mathbf{x}$ of $K(d(\mathbf{x}, \mathbf{q}))$ that exist determine the order of smoothness of the predicted outputs. There is an important link

between smoothness control and overfitting. Seifert and Gasser (1994) explore a variety of approaches to handling insufficient data in local regression.

8.1 Ridge Regression

A potential problem is that the data points can be distributed in such a way as to make the regression matrix $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ in Equation 21 nearly singular. If there are not enough nearby points with non-zero weights in all directions, there are not enough different equations to solve for the unknown parameters β. *Ridge regression* (Draper and Smith, 1981) is used to prevent problems due to a singular data matrix. The following equation, instead of Equation 21, is solved for β:

$$\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \Lambda\right)\beta = \mathbf{Z}^{\mathrm{T}}\mathbf{v} + \Lambda\bar{\beta} \tag{39}$$

where Λ is a diagonal matrix with small positive diagonal elements λ_i^2:

$$\Lambda = \begin{pmatrix} \lambda_1^2 & 0 & \cdots & 0 \\ 0 & \lambda_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^2 \end{pmatrix} \tag{40}$$

and $\bar{\beta}$ is an apriori estimate or expectation of what the local model parameters will be (often $\bar{\beta}$ is taken to be a vector of all zeros). This is equivalent to adding n extra rows to $\mathbf{Z}$, each having a single non-zero element, λ_i, in the ith column. The equation $\mathbf{Z}\beta = \mathbf{v}$ becomes:

$$\begin{bmatrix} & & & \\ & \mathbf{Z} & & \\ & & & \\ \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \beta = \begin{bmatrix} \\ \mathbf{v} \\ \\ \lambda_1\bar{\beta}_1 \\ \lambda_2\bar{\beta}_2 \\ \vdots \\ \lambda_n\bar{\beta}_n \end{bmatrix} \tag{41}$$

Adding additional rows can be viewed as adding "fake" data, which, in the absence of sufficient real data, biases the parameter estimates to $\bar{\beta}$ (Draper and Smith, 1981). Another view of ridge regression parameters is that they are the Bayesian assumptions about the apriori distributions of the estimated parameters (Seber, 1977). As described in Section 12 on tuning, optimizing the ridge regression parameters using cross validation can identify irrelevant dimensions. These techniques also help combat overfitting.

8.2 Dimensionality Reduction

Principal components analysis (PCA) can also be used globally to eliminate directions in which there is no data (Wettschereck, 1994). However, it is rarely the case that there is

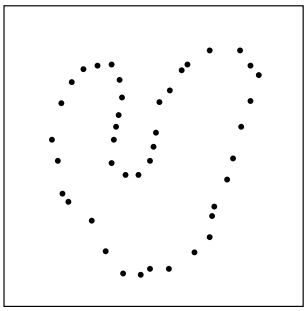


Figure 9: 2-dimensional input points scattered on a 1-dimensional non-linear manifold.

absolutely *no* data in a particular direction. A closely related technique, the singular value decomposition (SVD), is typically used in locally weighted regression to perform dimensionality reduction. Cleveland and Grosse (1991) compute the inverse of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ using the singular value decomposition, and then set small singular values to zero in the calculated inverse. This corresponds to eliminating those directions from the local model. Principal components analysis can also be done locally on the weighted data, either around each stored data point, or in response to a query. Directions can be eliminated in either a hard fashion, explicitly setting the corresponding parameters to zero, or in a soft fashion (such as performing ridge regression in the coordinate system defined by the PCA or SVD).

In Bregler and Omohundro (1994) an interesting locally weighted learning approach is presented for identifying low-dimensional submanifolds on which data is lying. In Figure 9 the space has two dimensions, and yet each dot is locally embedded on a one dimensional curve. Bregler and Omohundro's method uses locally weighted principal component analysis (which performs a singular value decomposition of the $\mathbf{Z}$ matrix from Equation 18) to identify local manifolds. This is a useful analysis tool for identifying local dependencies between variables in a dataset. But it also has important consequences for developing a local distance function: the principal component matrix reveals the directions in input space for which there is no data support.

These approaches only consider the input space (the space spanned by $\mathbf{x}_i$). It is often important to also consider the outputs (the y_i) when performing distance function or smoothing parameter optimization. The outputs can provide more opportunities for dimensionality reduction if they are flat in some direction, or can be predicted by a local model. An alternative perspective is to consider the conditional probability $\mathrm{p}(y|\mathbf{x})$. Perhaps one could do local principal components analysis in the joint density space $\mathrm{p}(\mathbf{x}, y)$ and eliminate the input directions that contribute least to predicting the outputs. A potential problem with dimensionality reduction in general is that the new dimensions, if not aligned with the previous dimensions, are not necessarily meaningful. Our focus is on reducing prediction error, ignoring comprehensibility of the local models.

9 Assessing the Predictions

An important aspect of locally weighted learning is that it is possible to estimate the prediction error, and derive confidence bounds on the predictions. Bottou and Vapnik (1992; Vapnik and Bottou, 1993) analyze confidence intervals for locally weighted classifiers. We

start our analysis of locally weighted regression by pointing out that LWR is an estimator that is linear in the output data $\mathbf{y}$ (using Equations 20, 22, and 39):

$$\hat{y}(\mathbf{q}) = \mathbf{q}^{\mathrm{T}}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \Lambda)^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{W}\mathbf{y} = \mathbf{s}_{\mathbf{q}}^{\mathrm{T}}\mathbf{y} = \sum_{i=1}^{n} \mathbf{s}_i(\mathbf{q}) y_i \tag{42}$$

The vector $\mathbf{s}_{\mathbf{q}}$, also written as $\mathbf{s}(\mathbf{q})$, will be useful for calculating the bias and variance of locally weighted learning.

9.1 Estimating the Variance

To calculate the variance of a prediction we assume the training data came from a sampling process that measures output values with additive random noise:

$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{43}$$

where the ϵ_i are independent, have zero mean, and have variance $\sigma^2(\mathbf{x}_i)$. Under the assumption that $\sigma^2(\mathbf{x}_i) = \sigma^2$ (σ is a constant) and that the linear model correctly models the structure of the data, linear regression generates an unbiased estimate of the regression parameters. Additionally, if the error is normally distributed, $\epsilon_i = N(0, \sigma^2)$, the regression estimate becomes the best linear unbiased estimate in the maximum likelihood sense. However, unless stated explicitly, in this paper we will avoid any distributional assumption on the form of the noise.

Given this model of the additive noise (and dropping the assumption that a linear model correctly models the structure of the data), the expectation and variance of the estimate $\hat{y}$ is ($\mathbf{s}$ is from Equation 42):

$$\mathrm{E}(\hat{y}(\mathbf{q})) = \mathrm{E}(\mathbf{s}_{\mathbf{q}}^{\mathrm{T}}\mathbf{y}) = \mathbf{s}_{\mathbf{q}}^{\mathrm{T}}\mathrm{E}(\mathbf{y}) = \sum_i \mathbf{s}_i(\mathbf{q}) f(\mathbf{x}_i) \tag{44}$$

$$\mathrm{Var}(\hat{y}(\mathbf{q})) = \mathrm{E}\left[\hat{y}(\mathbf{q}) - \mathrm{E}(\hat{y}(\mathbf{q}))\right]^2 = \sum_i \mathbf{s}_i^2(\mathbf{q})\sigma^2(\mathbf{x}_i) \tag{45}$$

One way to derive confidence intervals for the predictions from locally weighted learning is to assume a locally constant variance $\sigma^2(\mathbf{q})$ at the prediction point $\mathbf{q}$ and to use Equation 45. This equation has to be modified to reflect both the additive noise in sampling at the new point ($\sigma^2(\mathbf{q})$) and the prediction error of the estimator ($\sigma^2(\mathbf{q})\mathbf{s}_{\mathbf{q}}^{\mathrm{T}}\mathbf{s}_{\mathbf{q}}$).

$$\mathrm{Var}(y_{\mathrm{new}}(\mathbf{q})) = \sigma^2(\mathbf{q}) + \sigma^2(\mathbf{q})\mathbf{s}_{\mathbf{q}}^{\mathrm{T}}\mathbf{s}_{\mathbf{q}} \tag{46}$$

This expression of the prediction intervals is independent of the output values of the training data y_i, and reflects how well the data is distributed in the input space. However, the variance only reflects the difference between the prediction and the mean prediction, and not the difference between the prediction and the true value, which requires knowledge of the predictor's bias. Only when the local model structure is correct will the bias be zero.

To conveniently derive an estimate of $\sigma^2(\mathbf{x})$ we will define some additional quantities in terms of the weighted variables. A locally weighted linear regression centered at a point

$\mathbf{q}$ produces local model parameters $\beta(\mathbf{q})$. It also produces errors (*residuals*) at all training points. The weighted residual $r_i(\mathbf{q})$ is given by (v_i is defined in Equation 19):

$$r_i(\mathbf{q}) = \mathbf{z}_i^{\mathrm{T}}(\mathbf{q})\beta(\mathbf{q}) - v_i(\mathbf{q}) \tag{47}$$

The training criteria, which is the weighted sum of the squared errors, is given by:

$$C(\mathbf{q}) = \sum_i r_i^2(\mathbf{q}) \tag{48}$$

A reasonable estimator for the local value of the noise variance is

$$\hat{\sigma}^2(\mathbf{q}) = \frac{\sum r_i^2(\mathbf{q})}{n_{\mathrm{LWR}}(\mathbf{q})} = \frac{C(\mathbf{q})}{n_{\mathrm{LWR}}(\mathbf{q})} \tag{49}$$

where n_{LWR} is a modified measure of how many data points there are:

$$n_{\mathrm{LWR}}(\mathbf{q}) = \sum_{i=1}^{n} w_i^2 = \sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{q})}{h}\right) \tag{50}$$

In analogy to unweighted regression (Myers, 1990), we can reduce the bias of the estimate $\hat{\sigma}^2(\mathbf{q})$ by taking into account the number of parameters in the locally weighted regression:

$$\hat{\sigma}^2(\mathbf{q}) = \frac{\sum r_i^2(\mathbf{q})}{n_{\mathrm{LWR}}(\mathbf{q}) - p_{\mathrm{LWR}}(\mathbf{q})} \tag{51}$$

where p_{LWR} is a measure of the local number of free parameters in the local model:

$$p_{\mathrm{LWR}}(\mathbf{q}) = \sum_i w_i^2 \mathbf{z}_i^{\mathrm{T}} (\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1} \mathbf{z}_i \tag{52}$$

We have described a variance estimator that uses only local information. An alternative way to obtain a variance estimate uses global information, i.e., information from more than one LWR fit, and assumes a single global value for the additive noise (Cleveland et al., 1988; Cleveland and Grosse, 1991; Cleveland et al., 1992).

9.2 Estimating the Bias

Assessing the bias requires making assumptions about the underlying form of the true function, and the data distribution. In the case of locally weighted learning this is a weak assumption, since we need to know only the local behavior of the function and the local distribution of the data. Let us assume that the real function f is described locally by a quadratic model:

$$f(\mathbf{x}) = f(\mathbf{q}) + \mathbf{g}^{\mathrm{T}}(\mathbf{x} - \mathbf{q}) + \frac{1}{2}(\mathbf{x} - \mathbf{q})^{\mathrm{T}}\mathbf{H}(\mathbf{x} - \mathbf{q}) \tag{53}$$

where $\mathbf{q}$ is the query point, $\mathbf{g}$ is the true gradient at the query point, and $\mathbf{H}$ is the true Hessian (matrix of second derivatives) at the query point. The expected value of the estimate is given by Equation 44, which can be used to find the bias:

$$\text{bias} = \mathrm{E}(\hat{y}(\mathbf{q})) - y_{\mathrm{true}}(\mathbf{q}) = \sum [\mathbf{s}_i(\mathbf{q}) f(\mathbf{x}_i)] - f(\mathbf{q}) \tag{54}$$

This equation can be solved if we know the true function. For example, for the locally quadratic function, we can plug the quadratic function for $f(\mathbf{x})$ in Equation 53 into Equation 54 to get:

$$\text{bias} = f(\mathbf{q})\sum[\mathbf{s}_i(\mathbf{q})] - f(\mathbf{q}) + \mathbf{g}^{\mathrm{T}}\sum[\mathbf{s}_i(\mathbf{q})(\mathbf{x}-\mathbf{q})] + \frac{1}{2}\sum\left[\mathbf{s}_i(\mathbf{q})(\mathbf{x}-\mathbf{q})^{\mathrm{T}}\mathbf{H}(\mathbf{x}-\mathbf{q})\right] \quad (55)$$

The locally weighted regression process that generates $\mathbf{s_q}$ guarantees that $\sum \mathbf{s}_i(\mathbf{q}) = 1$, and since the linear local model exactly matches any linear trend in the data, $\sum \mathbf{s}_i(\mathbf{q})(\mathbf{x}-\mathbf{q}) = 0$. Therefore, the bias depends only on the quadratic term (Katkovnik, 1979; Cleveland and Loader, 1994a):

$$\text{bias} = \frac{1}{2}\sum_i \mathbf{s}_i(\mathbf{q})(\mathbf{x}_i-\mathbf{q})^{\mathrm{T}}\mathbf{H}(\mathbf{x}_i-\mathbf{q}) \quad (56)$$

assuming the ridge regression parameters λ_i have been set to zero. This formula raises the temptation to estimate and cancel the bias by estimating the second derivative matrix $\mathbf{H}$. It is not clear that this is better than simply using a quadratic local model instead of a linear local model. The quadratic local model would eliminate the local bias due to the quadratic term (and also remove the need for the distance metric to compensate for different curvature in different directions). Of course, if a quadratic local model is used, the bias will then be due to cubic terms in the Taylor series for the true function, whose elimination would require estimation of the cubic terms with a cubic local model, and so on. We have not yet found a principled termination of this cycle.

9.3 Assessment Using Cross Validation

We can assess how well locally weighted learning is doing by testing how well each experience $(\mathbf{x}_i, y_i)$ in the memory is predicted by the rest of the experiences. A simple measure of the ith prediction error is the difference between the predicted output of the input $\mathbf{x}_i$ and the observed value y_i. However, for non-parametric learners that are overfitting the data this measure may be deceptively small. For example, a nearest neighbor learner would always have an error measure of zero because $(\mathbf{x}_i, y_i)$ will be the closest neighbor to itself.

A more sophisticated measure of the ith prediction error is the *leave-one-out cross validation* error, in which the experience is first removed from the memory before prediction. Let $\hat{y}_i^{\mathrm{cv}}$ be the output predicted for input $\mathbf{x}_i$ using the memory with the ith point removed:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_{i-1}, y_{i-1}), (\mathbf{x}_{i+1}, y_{i+1}), \ldots, (\mathbf{x}_n, y_n)\} \quad (57)$$

The ith leave-one-out cross validation error is $e_{\mathrm{i}}^{\mathrm{cv}} = (\hat{y}_i^{\mathrm{cv}} - y_i)$. With the lazy learning formalism, in which most work takes place at prediction time, it is no more expensive to predict a value with one data point removed than with it included. This contrasts with the majority of learning methods that have an explicit training stage—in these cases it is not easy to pick an earlier experience and temporarily pretend we did not see it. Ignoring a training data point typically requires retraining from scratch with a modified training set, which can be fairly expensive with a nonlinear parametric model such as a neural network. In addition, the dependence of nonlinear parametric training on initial

parameter values further complicates the analysis. To handle this effect correctly many training runs with different starting values must be undertaken for each different data set. All of this is avoided with locally weighted learning with local models that are linear in the unknown parameters, although tuning of fit parameters does reintroduce the problem. However, tuning of fit parameters can be a background process that operates on a slower time scale than adding new data and answering queries.

Cross validation can also be performed locally, i.e, from just fitting a locally linear model at one query point $\mathbf{q}$ (Cleveland and Loader, 1994c). We first consider the locally weighted average of the squared cross validation error MSE^{cv} at each training point (Myers, 1990):

$$\text{MSE}^{\text{cv}}(\mathbf{q}) = \frac{\sum (e^{\text{cv}}_{i,\mathbf{x}_i})^2 K(d(\mathbf{x}_i, \mathbf{q}))}{\sum K(d(\mathbf{x}_i, \mathbf{q}))} \tag{58}$$

This estimate requires a locally weighted regression to be performed at each training point with non-zero weight $K(d(\mathbf{x}_i, \mathbf{q}))$. One could imagine storing $e^{\text{cv}}_{i,\mathbf{x}_i}$ with each training point, but this value would have to be updated as new data was learned. We approximate $e^{\text{cv}}_{i,\mathbf{x}_i} \approx e^{\text{cv}}_{i,\mathbf{q}}$. to generate the following:

$$\text{MSE}^{\text{cv}}(\mathbf{q}) = \frac{\sum (e^{\text{cv}}_{i,\mathbf{q}})^2 K(d(\mathbf{x}_i, \mathbf{q}))}{\sum K(d(\mathbf{x}_i, \mathbf{q}))} = \frac{\sum (r^{\text{cv}}_{i,\mathbf{q}})^2}{n_{\text{LWR}}} \tag{59}$$

where $r^{\text{cv}}_{i,\mathbf{q}}$ is the weighted cross validation error with point i removed from a locally weighted regression centered at $\mathbf{q}$. The weighted cross validation residual $r^{\text{cv}}_{i,\mathbf{q}}$ is related to the weighted residual ($r_i = w_i e_i$) by (Myers, 1990):

$$r^{\text{cv}}_i = \frac{r_i}{1 - \mathbf{z}_i^{\text{T}} (\mathbf{Z}^{\text{T}}\mathbf{Z} + \Lambda)^{-1} \mathbf{z}_i} \tag{60}$$

Thus, we obtain the final equation for MSE^{cv} as:

$$\text{MSE}^{\text{cv}}(\mathbf{q}) = \frac{1}{n_{\text{LWR}}} \sum_i \left(\frac{r_i}{1 - \mathbf{z}_i^{\text{T}} (\mathbf{Z}^{\text{T}}\mathbf{Z} + \Lambda)^{-1} \mathbf{z}_i} \right)^2 \tag{61}$$

This equation is a local version of the PRESS statistic (Myers, 1990). It allows us to perform leave-one-out cross validation without recalculating the regression parameters for every excluded point. Often, this is computationally very efficient.

10 Optimal Fit Parameters: An Example

In this section we will try to find optimal fit parameters (distance metric $d()$, weighting function $K()$, and smoothing bandwidth h) for a simple example. We will make the restrictive assumption that the data is uniformly spaced on a rectangular grid. We first approach this question by exploring kernel shapes in one dimension. We allow the weights w_i to be unknown, and numerically optimize them to minimize the mean squared error. We assume the underlying function is quadratic with second derivative $\mathbf{H}$ (Equation 53) and that there is additive independent identically distributed zero mean noise (Equation 43) with constance variance σ^2. The sampled data is regularly spaced with a

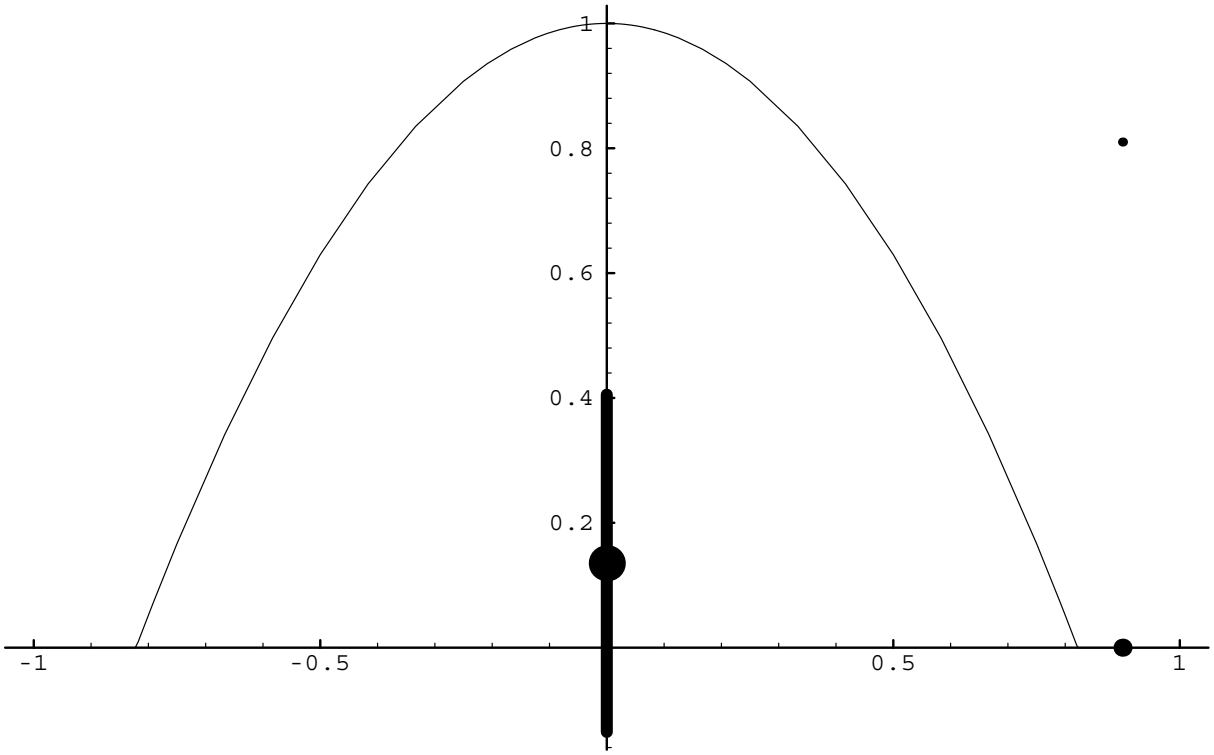


Figure 10: The kernel shape that minimizes mean squared error in one dimension. The large single dot is the predicted value, whose deviation from zero, the correct value, reveals the bias. The vertical bars show the standard deviation of the prediction (i.e., the square root of the variance), which is greatly reduced from the standard deviation of ± 1 of the original data. The set of large dots have been optimized to minimize the mean squared error of the prediction, and reveal the optimal kernel shape for this criterion. The line through these points is a quadratic kernel with the appropriate bandwidth to match the optimized kernel values. The small dots are the value of the quadratic portion of the underlying function, for comparison.

distance of Δ between each data point (in Figure 10 $\Delta = 0.1$). Equation 42 is solved for $\mathbf{s}$, with the query at $\mathbf{x} = 0$. The mean squared error is the sum of the bias (Equation 54) squared and the variance (Equation 45). This quantity is minimized by adjusting the weights w_i. The resulting kernel shape $K(d) = w_i^2$ is shown in Figure 10. This kernel shape matches the quadratic kernel:

$$K(d) = \begin{cases} (1 - d^2) & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{62}$$

which has been described in Section 6.

Further numerical experimentation in one dimension revealed that the optimal scaling factor m for the one dimensional distance function is approximately:

$$m^2 \approx c\mathbf{H} \tag{63}$$

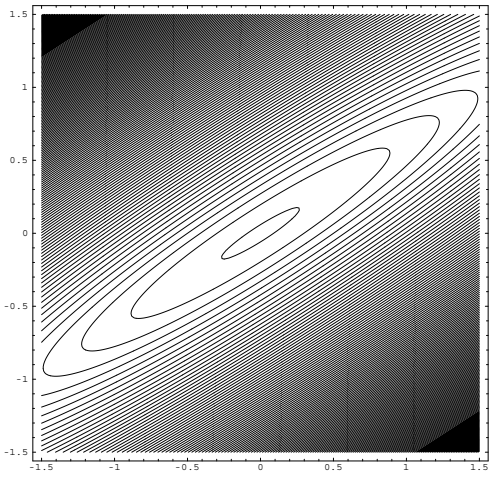


Figure 11: Contour plot of $f(\mathbf{x})$.

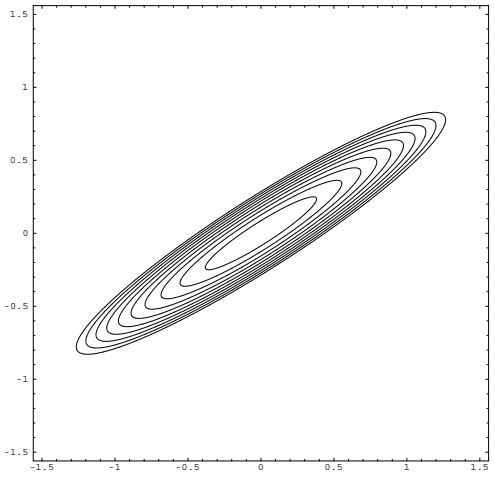


Figure 12: Contour plot of optimal kernel.

where c is a constant that takes into account issues such as data spacing Δ and the standard deviation of the additive noise:

$$c \propto \frac{\Delta^2}{\sigma} \tag{64}$$

The width of the resulting kernel is directly related to the optimal smoothing bandwidth.

In two dimensions we can explore optimization of the distance metric. Optimizing the values of the kernel at each of the data points is beyond our current computational resources, so we will assume the form of the kernel function is the quadratic kernel. We will choose a particular value for the Hessian $\mathbf{H}$ in Equation 53, and then optimize the scaling matrix $\mathbf{M}$ for the multidimensional distance function to minimize the mean squared error. We found that the optimal $\mathbf{M}$ approximately satisfies the following equation:

$$\mathbf{M}^{\mathrm{T}}\mathbf{M} \approx c\mathbf{H} \tag{65}$$

where c is the same as the one dimensional case. Figure 11 shows how the Hessian matrix $\mathbf{H}$ can orient the quadratic component in an arbitrary orientation. The distance function matrix $\mathbf{M}^{\mathrm{T}}\mathbf{M}$ needs to be a full matrix in order to allow the optimal kernel (Figure 12) to match the orientation of the quadratic component of $f(\mathbf{x})$ (Figure 11). For this numerical experiment $\mathbf{H}$ was chosen to be:

$$\mathbf{H} = \begin{pmatrix} 1.23851 & -1.77313 \\ -1.77313 & 2.86149 \end{pmatrix} \tag{66}$$

The optimal scaling matrix $\mathbf{M}$ was found by numerical search to be:

$$\mathbf{M} = \begin{pmatrix} 2.32597 & -3.33005 \\ 0.0 & 1.18804 \end{pmatrix} \tag{67}$$

and Equation 65 is approximately satisfied, as $(\mathbf{M}^{\mathrm{T}}\mathbf{M})\mathbf{H}^{-1}$ is almost a multiple of the identity matrix for $c = 4.37$.

$$(\mathbf{M}^{\mathrm{T}}\mathbf{M})\mathbf{H}^{-1} = \begin{pmatrix} 0.99949 & -0.0001 \\ 0.0008 & 1.0002 \end{pmatrix} \tag{68}$$

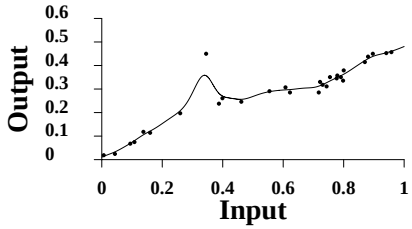


Figure 13: Locally weighted regression approximating a 1-dimensional dataset shown by the black dots. There is an outlier at $x \approx 0.33$.

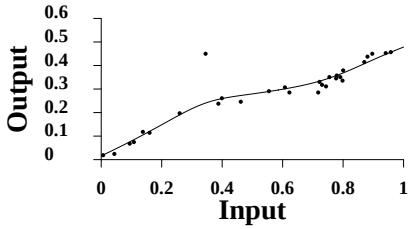


Figure 14: Locally weighted regression supplemented with outlier removal.

11 Noisy Training Data and Outliers

The averaging performed by the locally weighted regression process naturally filters out noise if the weighting function is not infinite at zero distance. The tuning process can optimize the noise filtering by adjusting fit parameters such as smoothing parameters, weighting function parameters, ridge regression parameters, and choice of local model structure. However, it is often useful to explicitly identify *outliers*: training points that are erroneous or whose noise is much larger than that of neighboring points. An example of the effect of an outlier is given in Figure 13 and the effect of outlier rejection is shown in Figure 14. Robust regression (see, for example Hampel et al., 1986) and cross validation allow extensions to locally weighted learners in which we can identify or reduce the effects of outliers. Outliers can be identified and removed globally, or they can be identified and ignored on a query by query basis. Query-based outlier detection allows training points to be ignored for some queries and used for other queries. Other areas that have been explored are detecting discontinuities and nonstationarity in the training data.

11.1 Global Weighting of Stored Points and Finding Outliers

It is possible to attach weights to stored points during the training process and during lookup that downweight points that are suspected of being unreliable (Aha and Kibler, 1989; Cost and Salzberg, 1993). These weights can multiply the weight based on the weighting function. Totally unreliable points can be assigned a weight of zero, leading them to be ignored. The reliability weights can be based on cross validation: whether a stored point correctly predicts or classifies its neighbors. Another approach is to only utilize stored points that have shown that they can reduce the cross validation error (Aha, 1990). Important issues are when the weighting decision is made and how often the decision is reevaluated. Global methods typically assign a weight to a point during training, in which case the decision is usually never reevaluated, or during an asynchronous database maintenance process, in which decisions are reevaluated each time the process cycles through the entire database.

11.2 Local Weighting of Stored Points and Finding Outliers

Local outlier detection methods do not label points as outliers for all queries, as do global methods. Points can be outliers for some queries and not outliers for others. We can generate weights for training data at query time based on cross validation using nearby points. The PRESS statistic (Myers, 1990) can be modified to serve as a local outlier detector in locally weighted regression. For this, we need the standardized individual PRESS residual (also called the Studentized residual):

$$e_{\text{PRESS}} = \frac{r_i}{\hat{\sigma}\sqrt{1 - \mathbf{z}_i^{\mathrm{T}}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \Lambda)^{-1}\mathbf{z}_i}} \tag{69}$$

This measure has zero mean and unit variance and assumes a locally normal distribution of the error. If, for a given data point it deviates from zero more than a certain threshold, the point can be called an outlier. A conservative threshold would be 1.96, discarding all points lying outside the 95% area of the normal distribution. In our applications, we used 2.57, cutting off all data outside the 99% area of the normal distribution.

11.3 Robust Regression Approaches

Data with outliers can be viewed as having additive noise with long-tailed symmetric distributions. Robust regression is useful for both global and local detection of outliers (Cleveland, 1979). A bisquare weighting function is used to additionally downweight points based on their residuals:

$$u_i = \begin{cases} \left(1 - \left(\frac{e_i}{6e_{\text{MED}}}\right)^2\right)^2 & \text{if } |e_i| < 6e_{\text{MED}} \\ 0 & \text{otherwise} \end{cases} \tag{70}$$

where e_{MED} is the median of the absolute value of the residuals e_i. The weights now become $w_i = u_i K(d(\mathbf{x}_i, \mathbf{q}))$. This process is repeated about 1-3 times to refine the estimates of u_i.

12 Tuning

Like most learning algorithms, locally weighted learning usually needs to be tuned to work well for a particular problem. Tuning means adjusting the parameters of the learning algorithm itself. The locally weighted fit criteria is

$$C(\mathbf{q}) = \sum_i \left[(f(\mathbf{x}_i, \beta) - y_i)^2 K\left(\frac{d(\mathbf{x}_i, \mathbf{q})}{h}\right)\right] \tag{71}$$

It includes several "fit" parameters: the bandwidth or smoothing parameter h, the distance metric $d()$, and the weighting or kernel function $K()$. There are additional fit parameters such as ridge regression parameters and outlier thresholds. There are several ways to tune these fit parameters.

- **Global tuning:** The fit parameters are set globally by an optimization process that typically minimizes cross validation error over all the data, and therefore constant size and shape volumes of data are used to answer queries.

- **Query-based local tuning:** The fit parameters are set on each query based on local information.

- **Point-based local tuning:** The weighted training criteria uses different fit parameters for each point $\mathbf{x}_i$: a bandwidth h_i, a distance metric $d_i()$, a weighting function $K_i()$, and possibly a weight $w_{\mathbf{x}_i}$:

$$C(\mathbf{q}) = \sum_i \left[(f(\mathbf{x}_i, \beta) - y_i)^2 w_{\mathbf{x}_i} K_i \left(\frac{d_i(\mathbf{x}_i, \mathbf{q})}{h_i} \right) \right] \tag{72}$$

  In typical implementations of this approach the fit parameters are computed in advance of the queries and are stored with the data points.

There are several approaches to computing the fit parameter values:

- **Plug-in approach:** The fit parameters can be set by a direct computation.

- **Optimization approaches:** The fit parameters can be set by an optimization process that either (Marron, 1988):
  - minimizes the training set error,
  - minimizes the test or validation set error,
  - minimizes the cross validation error (CV),
  - minimizes the generalized cross validation error (GCV) (Myers, 1990),
  - maximizes Akaike's information criterion (AIC),
  - or adjusts Mallow's C_p.

Fit parameters cannot be optimized in isolation. The combination of all fit parameters generates a particular fit quality. If one fit parameter is changed, typically the optimal values of other parameters change in response. If a locally constant model is used, then the smoothing parameter and distance function must reflect the flatness of the neighborhood in different directions. If the local model is a hyperplane, the smoothing parameter and distance function must reflect the second derivative of the neighborhood. If the local model is quadratic, it is the third spatial derivative of the data that must be dealt with.

For practical purposes it would be useful to have a clear understanding of how accurate the non-linear fit parameters should be for a good fit. Our intuition is that approximate values usually result in barely distinguishable performance from optimal parameters in practical use, although (Brockmann et al., 1993) states that this is not true for h in kernel regression.

The next section considers optimizing a single set of parameters for all possible future queries (global tuning). Section 12.2 considers optimizing multiple sets of parameters for specific queries (local tuning).

12.1 Global Tuning

Global cross-validation can be a particularly robust method for tuning parameters, because it does not make any special assumptions. Independent of the noise distribution, data distribution and underlying function, the cross-validation value is an unbiased estimate of how well a given set of parameters will perform on new data drawn from the same distribution as the old data. This robustness has lead to the use of global cross-validation in applications that attempt to achieve high autonomy by making few assumptions, such as the General Memory Based Learning (GMBL) system described in (Moore et al., 1992). GMBL performs large amounts of cross validation search to optimize feature subsets, the diagonal elements of the distance metric, the smoothing parameter, and the order of the regression.

12.1.1 Continuous Search

Continuous fit parameters make continuous search possible. Inevitably this is local hill climbing, with a large risk of getting stuck in local optima. The sum of the squared cross validation errors is minimized using a nonlinear parameter estimation procedure (e.g., MINPACK (More et al., 1980) or NL2SOL (Dennis et al., 1981)). As discussed in Section 9.3, in this locally weighted learning approach computing the cross validation error for a single point is no more computationally expensive than answering a query. This is quite different from parametric approaches such as a neural network, where a new model (network) must be trained for each cross validation training set with a particular point removed. In addition, if the local model is linear in the unknown parameters we can analytically take the derivative of the cross validation error with respect to the parameters to be estimated, which greatly speeds up the search process.

We can use the optimized distance metric to find which input variables are more or less important to the function being represented. Distance scaling factors that go to zero indicate directions that are irrelevant or are consistent with the local model structure, and that a global model will suffice for those directions. We can also interpret the ridge regression parameters. The ridge regression parameters for irrelevant terms in the local model become very large in the fit parameter optimization process. The effect of this is to force the corresponding estimated parameters β_i to the apriori values $\bar{\beta}_i$, which corresponds to a dimensionality reduction.

A relatively unexplored area is stochastic gradient descent approaches to optimizing fit parameters. Rather than use all the cross validation errors and their associated contributions to the derivative, why not use only a small random sample of the cross validation errors and their associated derivative contributions? Racine (1993) describes an approach to optimizing fit parameters based on partitioning the training data into subsets, calculating cross validation errors for each subset based only on data in the subset, and then averaging the results.

12.1.2 Discrete Search

Discrete search algorithms for good fit parameters is an active area of research. Maron and Moore (1996) describe "racing" techniques to find good fit parameter values. These

techniques compare a wide range of different types of models simultaneously, and handle models with discrete parameters. Bad models are quickly dropped from the race, which focuses computational effort on distinguishing between the good models. Typically any continuous fit parameters are discretized (Maron and Moore, 1996).

Techniques for selecting features in the distance metric and local model have been developed in statistics (Draper and Smith, 1981; Miller, 1990), including all subsets, forward regression, backwards regression, and stepwise regression. We have explored stepwise regression procedures to determine which terms of the local model are useful with similar results to the gradient based search described above. Feature selection is a hard problem because the features cannot be examined independently. The value of a feature depends on which other features are also selected. Thus the goal is to find a *set* of feature weights, not individual feature weights for each feature. In Maron and Moore (1996) a number of algorithms for doing this are described and compared, including methods based on Monte-Carlo sampling. Aha (1991) gives an algorithm that constructs new features, in addition to selecting features. Friedman (1994) gives techniques for query dependent feature construction.

12.1.3 Continuous vs. Discrete Search

Discrete search can explore settings for discrete fit parameters, and even select training algorithm features or function approximation methods (e.g., locally weighted regression, neural networks, rule-based systems). It would seem that continuous fit parameter optimization cannot make these choices. However, this is not the case. By blending the output of different approaches with a blending parameter α, continuous search can choose model order, algorithm features, and approximation method. For example, α could be optimized to blend two methods $f_1()$ and $f_2()$ in the following equation:

$$f(\mathbf{q}) = \alpha f_1(\mathbf{q}) + (1 - \alpha) f_2(\mathbf{q}) \tag{73}$$

Cleveland and Loader (1994a,c) present an approach to automatically choose the local model structure (i.e., order of the polynomial model) by blending polynomial models, where a non-integral model order indicates a weighted blend between two integral model orders. They use cross validation to optimize the local model order on each query.

12.2 Local Tuning

Local fit parameter optimization is referred to as "adaptive" or "variable" in the statistics literature, as in "adaptive bandwidth" or "variable bandwidth" smoothers. There are several reasons to consider local tuning, although it dramatically increases the number of degrees of freedom in the training process, leading to increased variance of the predictions and an increased risk of overfitting the data (Cleveland and Loader, 1994c):

- **Adaptation to the data density and distribution:** This adaptation is in addition to the adaptation provided by the locally weighted regression procedure itself (Bottou and Vapnik, 1992).

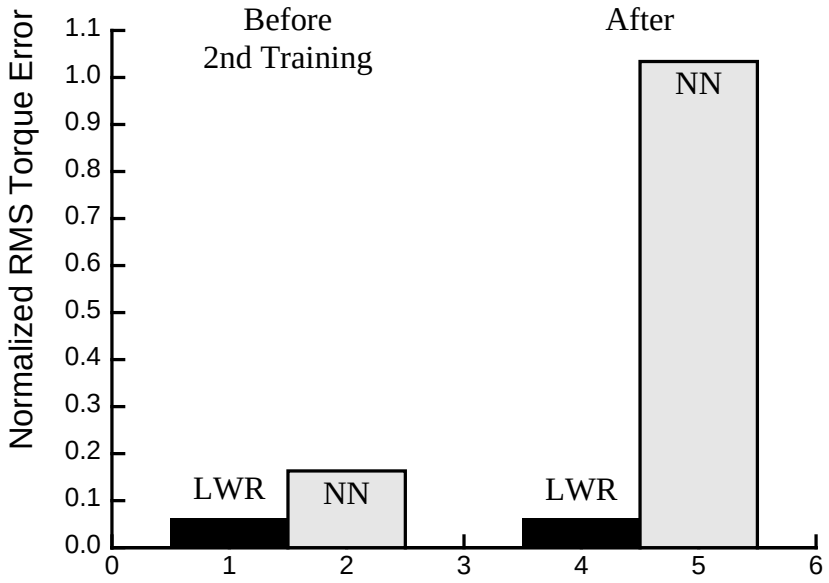


Figure 15: Performance of various methods on two joint arm dynamics.

- **Adaptation to variations in the noise level in the training data.** These variations are known as *heteroscedasticity*.
- **Adaptation to variations in the behavior of the underlying function.** The function may be locally planar in some regions and have high curvature in others.

"Plug-in" estimators have been derived and local (locally weighted) training set error, cross validation, or validation (test) set error can drive an optimization of the local model.

13 Interference

Negative interference between old and new training data is one of the most important motivations for exploring locally weighted learning. To illustrate the differences between global parametric representations and a locally-weighted learning approach, a sigmoidal feedforward neural network approach was compared to a locally weighted learning approach on the same problem. The architecture for the sigmoidal feedforward neural network was taken from (Goldberg and Pearlmutter, 1988, section 6) who modeled arm inverse dynamics. The ability of each of these methods to predict the torques of the simulated two joint arm at 1000 random points was compared (Atkeson, 1992). Figure 15 plots the normalized RMS prediction error. The points were sampled uniformly using ranges comparable to those used in Miller et al. (1987), which also looked at two joint arm inverse dynamics modeling. Initially, each method was trained on a training set of 1000 random samples, and then the predictions of the torques on a separate test set of 1000 random samples of the two joint arm dynamics function were assessed. The solid bar marked LWR at location 1 shows the test set error of a locally weighted regression with a quadratic local model. The light bar marked NN at location 2 shows the best test

set error of the neural network. Both methods generalize well on this problem (bars 1 and 2 have low error).

Each method was then trained on ten attempts to make a particular desired movement. Each method successfully learned the desired movement. After this second round of training, performance on the random test set was again measured (bars at locations 4 and 5). The sigmoidal feedforward neural network lost its memory of the full dynamics (the light bar at location 5 has a large error), and represented only the dynamics of the particular movements being learned in the second training set. This interference between new and previously learned data was not prevented by increasing the number of hidden units in the single layer network from 10 up to 100. The locally weighted learning method did not show this interference effect (solid bar at location 4).

The interference is caused by the failure of the neural network model structure to match the arm inverse dynamics structure perfectly. There is no noise in the data, and no concept drift, so these causes are eliminated as possible sources of the interference. It can be argued that the sigmoidal neural network forgot the original training data because we did not include that data in the second training data set (learning a specific movement). That is exactly our point; if all past data is retained to combat interference, then the method becomes a lazy learning method. In that case we argue that one should take advantage of the opportunity to locally weight the training procedure, and get better performance (Vapnik, 1992; Bottou and Vapnik, 1992; Vapnik and Bottou, 1993).

14 Implementing Locally Weighted Learning

There are several concerns about locally weighted learning systems, including whether locally weighted learning systems can answer queries fast enough and whether their speed will unacceptably degrade as the size of the database grows. This section explores these concerns. We discuss fast ways to find relevant data using either k-d trees in software, special purpose hardware, or massively parallel computers, and the current performance of our LWR implementation. Our goal is to minimize the need for compromises such as forgetting (discarding data) to keep the database size under a limit, instance averaging, which averages similar data, or maintaining an elaborate data structure of intermediate results to accelerate query processing. We will not discuss LWR acceleration approaches that are limited to low dimensional problems such as binning (Fan and Marron, 1994a; Turlach and Wand, 1995). Other discussions of fast implementations include Seifert et al. (1994) and Seifert and Gasser (1994).

14.1 Retrieving Relevant Data

The choice of method for storing experiences depends on what fraction of the experiences are used in each locally weighted regression and what computational technology is available. If all of the experiences are used in each locally weighted regression, then simply maintaining a list or array of experiences is sufficient. If only nearby experiences are included in the locally weighted regression, then an efficient method of finding nearest neighbors is required. Nearest neighbor lookup can be accelerated on a serial proces-

sor using the k-d tree data structure. Parallel processors and special purpose processors typically use parallel exhaustive search.

14.1.1 K-d Trees

Naively implemented search for a d dimensional nearest neighbor in a database of size n requires n distance computations. Nearest neighbor search can be implemented efficiently by means of a *k-d tree* (Bentley, 1975; Friedman et al., 1977; Bentley and Friedman, 1979; Bentley et al., 1980; Murphy and Selkow, 1986; Ramasubramanian and Paliwal, 1989; Broder, 1990; Samet, 1990; Sproull, 1991). A k-d tree is a binary data structure that recursively splits a d-dimensional space into smaller subregions. The search for a nearest neighbor proceeds by initially searching the k-d tree in the branches nearest the query point. Frequently, distance constraints mean there is no need to explore further branches. Figure 16 shows a k-d tree segmenting a two dimensional space. The shaded regions correspond to areas of the k-d tree that were not searched.

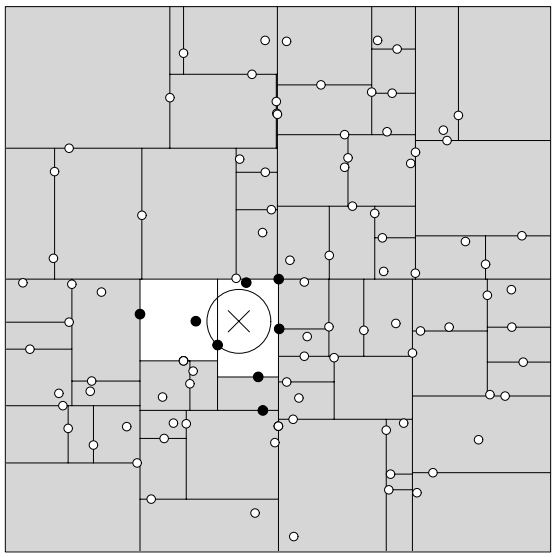


Figure 16: Generally during a nearest neighbor search only a few leaf nodes need to be inspected. The query point is marked by an x and the distance to the nearest neighbor is indicated by a circle. Black nodes are those inspected on the path to the leaf node.

The access time is asymptotically logarithmic in n, the size of the memory, although often overhead costs mean that nearly all the data points *will* be accessed in a supposed logarithmic search, for example, with eight dimensions or more and fewer than approximately 100,000 uniformly distributed data points. In fact, given uniformly distributed data points, the tree size for which logarithmic performance is noticeable increases exponentially with dimensionality. Two things can alleviate this problem. First, the data points are unlikely to be distributed uniformly. In fact, the less randomly distributed the training data is the better. Second, there are approximate algorithms that can find one or more nearby experiences, without guaranteeing they are the nearest, that *do* operate in logarithmic time. Empirically, these approximations do not greatly reduce prediction accuracy (Omohundro, 1987; Moore, 1990b). Bump trees (Omohundro, 1991) are another promising efficient approximation. Cleveland et al. (1988), Farmer and Sidorowich (1988a,b), Renka (1988), Grosse (1989), Moore (1990a), Cleveland and Grosse (1991), Karalič (1992), Townshend (1992), Loader (1994), Wess et al. (1994), Deng and Moore

(1994), Lowe (1995) and van der Smagt et al. (1994) have used trees in memory-based learning and locally weighted regression.

14.1.2 Special Purpose Devices

Special purpose hardware for finding nearest neighbors has a long history (Taylor, 1959, 1960; Steinbuch, 1961; Steinbuch and Piske, 1963; Kazmierczak and Steinbuch, 1963; Batchelor, 1974). These machines calculated either a Manhattan or Euclidean distance for all stored points, and then did comparisons to pick the winning point. The current version of this technology is the wafer scale memory–based reasoning devices proposed by Yasunaga and Kitano (1993). The devices allocate one processor per data point, and can handle approximately 1.7 million data points per 8 inch wafer. The designers have exploited the properties of memory-based learning in two ways. First, the resolution of the computed distance is not critical, so analog adders and multipliers are used for weighting and distance calculations instead of digital circuits, saving much space on the silicon for other processors. Second, the device is robust to faulty processors, in that a faulty processor only causes the loss of a single data point. The authors advocate simply ignoring processor failures, although it would be possible to map the faulty processors and skip them when loading data.

14.1.3 Massively Parallel Implementations

Many nearest neighbor systems have been implemented on massively parallel Connection Machines (Waltz, 1987). On a massively parallel computer, such as the CM1 and CM2 (Hillis, 1985), exhaustive search is often faster than using k-d trees, due to the limited number of experiences allocated to each processor. The Connection Machine can have up to 2^{16} (65536) processors, and can simulate a parallel computer with many more processors. Experiences are stored in the local memory associated with each processor. An experience can be compared to the desired experience in each processor, with the processors running in parallel, and then a hardwired global-OR bus can be used to find the closest match in constant time independent of the number of stored experiences. The search time depends linearly on the number of dimensions in the distance metric, and the distance metric can be changed easily or made to depend on the current query point.

The critical feature of the massively parallel computer system IXM2 is the use of associative memories in addition to multiple processors (Higuchi et al., 1991). There are 64 processors (Transputers) in the IXM2, but each processor has 4Kx40 bits of associative memory, which increases the effective number of processors to 256K. This architecture is well suited for memory-based learning where the distance metric involves exact matches of symbolic fields, as that is the operation the associative memory chips can support. Future associative memories might implement Euclidean distance as a basic operation. There have been implementations of memory-based translation and parsing on the IXM2 (Kitano and Higuchi, 1991a,b; Sumita et al., 1993; Kitano, 1993a,b).

The current generic parallel computer seems to be on the order of 100 standard microprocessors tightly connected with a communication network. Examples of this design are the CM5 and the SNAP system (Kitano et al., 1991). The details of the communication network are not critical to locally weighted learning, since the time critical processing

consists of broadcasting the query to the processors and determining which answer is the best, which can easily be done with a prespecified communication pattern. This form of communication is not difficult to implement. This machine does not have the thousands of processors that make exhaustive search the obvious nearest neighbor algorithm. The processors will probably maintain some sort of search data structure such as a k-d tree, although the local k-d trees may be too small for efficient search performance. Kitano et al. (1991) describe an implementation of memory-based reasoning on the SNAP system. This type of parallel computer is excellent for locally weighted learning, where the regression calculation dominates the lookup time if a large fraction of the points are used in each regression.

14.2 Implementing Locally Weighted Regression

Locally weighted learning minimizes the computational cost of training; new data points are simply stored in the memory. The price for trivial training costs is a more expensive lookup procedure. Locally weighted regression uses a relatively complex regression procedure to form the local model, and is thus more expensive than nearest neighbor and weighted average memory-based learning procedures. For each query a new local model is formed. The rate at which local models can be formed and evaluated limits the rate at which queries can be answered. We have implemented the locally weighted regression procedure on a 33MHz Intel i860 microprocessor. The peak computation rate of this processor is 66 MFlops. We have achieved effective computation rates of 15 MFlops on a learning problem with 10 input dimensions and 5 output dimensions, using a linear local model. This leads to a lookup time of approximately 15 milliseconds on a database of 1000 points, using exhaustive search. This time includes distance and weight calculation for all the stored points, forming the regression matrix, and solving the normal equations.

15 Applications of Locally Weighted Learning

The presence of the LOWESS and LOESS software in the S statistics package has lead to the use of locally weighted regression as a standard tool in many areas, including modeling biological motor control, feeding cycles in smokers and nonsmokers, lead-induced anemia, categories of tonal alignment in spoken English, and growth and sexual maturation during disease (Cleveland, 1979; Cleveland et al., 1992).

Atkeson et al. (1996) survey our own work in applying locally weighted learning to robot control. Zografski has explored the use of locally weighted regression in robot control and modeling time series, and also compared LWR to neural networks and other methods (Zografski, 1989, 1991, 1992; Zografski and Durrani, 1995). Gorinevsky and Connolly (1994) compared several different approximation schemes (neural nets, Kohonen maps, radial basis functions, and local polynomical fits) on simulated robot inverse kinematics with added noise, and showed that local polynomial fits were more accurate than all other methods. van der Smagt et al. (1994) learned robot kinematics using local linear models at the leaves of a tree data structure. Tadepalli and Ok (1996) apply local linear regression to reinforcement learning. Baird and Klopf (1993) apply nearest

neighbor techniques and weighted averaging to reinforcement learning and Thrun (1996) and Thrun and O'Sullivan (1996) apply similar techniques to robot learning. Connell and Utgoff (1987) interpolated a value function using locally weighted averaging to balance an inverted pendulum (a pole) on a moving cart. Peng (1995) performed the cart pole task using locally weighted regression to interpolate a value function. Aha and Salzberg (1993) explored nearest neighbor and locally weighted learning approaches to a tracking task in which a robot pursued and caught a ball. McCallum (1995) explored the use of lazy learning techniques in situations where states were not completely measured. Farmer and Sidorowich (1987, 1988a,b) apply locally weighted regression to modeling and prediction of chaotic dynamic systems. Huang (1996) uses nearest neighbor and weighted averaging techniques to cache simulation results and accelerate a movement planner.

Lawrence et al. (1996) compare neural networks and local regression methods on several benchmark problems. Local regression outperformed neural networks on half the benchmarks. Factors affecting performance included whether the data had differing density over the input space, noise level, dimensionality, and the nature of the function underlying the data.

Several researchers have applied locally weighted averaging and regression to free form 2D and 3D deformation, morphing, and image interpolation in computer graphics (Goshtasby, 1988; Wolberg, 1990; Ruprecht and Müller, 1992, 1993, 1994a; Ruprecht et al., 1994). Coughran, Jr. and Grosse (1991) describe using locally weighted regression for scientific visualization and auralization of data.

Ge et al. (1994) apply locally weighted regression to predict cell density in a fermentation process. They used nearest neighbor weighting and a tricube weighting function. They also used principal components and cross validation to select features globally. Locally weighted regression outperformed other methods, including a global nonlinear regression. Hammond (1991) used LWR to model fermentation as well.

Næs et al. (1990), Næs and Isaksson (1992) and Wang et al. (1994) apply locally weighted regression to analytical chemistry. They use global principal components to reduce the dimensionality of the inputs, and they use cross validation to set the number of components to use. They also explore several weighted Euclidean distance metrics, including weighting depending on the range of the data in principal component coordinates, weighting depending on how good that dimension is in predicting the output, and a distance metric that includes the output value. They use a quadratic local model and the tricube weighting function. They use cross validation to select the number of points to include in the local regression. They make the important point that optimal experiment design is quite different when using locally weighted regression as compared to global linear regression.

Tamada et al. (1993) apply memory-based learning to water demand forecasting. They select features using Akaike's Information Criterion (AIC), and use locally weighted averaging within a neighborhood. They use a default temporally local regression scheme if no points are found in the neighborhood. They use error rates to set feature weights and to perform outlier removal.

Townshend (1992) applies locally weighted regression to the analysis, modeling, coding, and prediction of speech signals. He uses a singular value decomposition to reduce the dimensionality of the regression to a fixed value D, determined from other criteria.

He uses the k closest points to form the local model. The distance to the nearest point is used as an estimate of the confidence in the prediction. A clustering process on the inputs and the outputs $(\mathbf{x}_i, y_i)$ is used to handle noise and one to many mapping problems. A k-d tree is used to speed up nearest neighbor search. This process lead to a significant improvement over a linear predictor.

Wijnberg and Johnson (1985) apply locally weighted regression to interpolating air quality measurements. They used cross validation to optimize the smoothing parameter globally, but did not find a well defined minimum for the smoothing parameter. Kozek (1992) describe using LWR to model automobile emissions.

Walden and Prescott (1983) use LWR to remove trends in time series involving climate data. Solow (1988) estimated the variance or noise level in time series climate data after having removed the mean using LWR.

Locally weighted regression has also been applied in economics and econometrics (Meese and Wallace, 1991; LeBaron, 1992). Meese and Rose (1990) used LWR to model exchange rates and conclude that no significant nonlinearity exists in the data. Diebold and Nason (1990) also used LWR to predict exchange rates, without any more success than other nonparametric regression techniques.

Turetsky et al. (1989) and Raz et al. (1989) use LWR to smooth biological evoked potential data, and explore approaches to choosing the smoothing parameter. Bottou and Vapnik (1992) apply locally weighted classification to optical character recognition (OCR). Rust and Bornman (1982) apply LWR to marketing data.

There have been a range of applications of locally weighted techniques in statistics (Cleveland, 1993b; Cleveland and Loader, 1994c). The idea of local fitting was extended to likelihood-based regression models by Tibshirani and Hastie (1987) and Hastie and Tibshirani (1990) applied locally weighted techniques to many distributional settings such as logistic regression and developed general fitting algorithms. Lejeune and Sarda (1992) applied locally weighted regression to estimation of distribution and density functions. Cleveland et al. (1993) applied locally weighted regression to density estimation, spectrum estimation, and predicting binary variables. Fan and Kreutzberger (1995) applied locally weighted regression to spectral density estimation.

16 Discussion

16.1 What Is A Local Learning Approach?

To explore the idea of local learning, it is useful to first consider what a global learner is. A global/distributed representation is typically characterized by:

1. Incrementally learning a single new training point affects many parameters.

2. A prediction or answer to a query also depends on many parameters.

1 and 2 are characteristics of distributed representations. An additional criterion:

3. There are many fewer parameters than data.

could serve as a definition of a global representation or model, and is a good predictor that 1 and 2 will be true for a particular method. However, it is also possible to have local methods with attribute 3, and not attributes 1 and 2, such as a low resolution tabular representation with non-overlapping cells. A part of the design space that has not been explored are learning algorithms with huge numbers of parameters that use distributed representations (1 and 2, but not 3).

There are at least three different views of what constitutes local learning: local representations, local selection, and locally weighted learning. This has lead to some confusion and convoluted terminology. In a local representation, each new data point affects a small subset of the parameters and answering a query involves a small subset of the parameters as well. This view of local learning stems from the distinction between local and distributed representations in neuroscience (Thorpe, 1995). Examples of local representations are lookup tables and exemplar/prototype based classifiers. It is not necessarily the case that the number of parameters in the representation be on the order of the number of data points (i.e., a considerable amount of local averaging can occur).

Local selection methods store all (or most) of the training data, and use a distance function to determine which stored points are relevant to the query. The function of local selection is to select a single output using nearest neighbor or using a distance-based voting scheme (k-nearest neighbor). Examples of these types of approaches are common, and include Stanfill and Waltz (1986) and Aha (1990).

Locally weighted learning stores the training data explicitly (as do local selection approaches), and only fits parameters to the training data when a query is known. The critical feature of locally weighted learning is that a criterion locally weighted with respect to the query location is used to fit some type of parametric model to the data (Vapnik, 1992; Bottou and Vapnik, 1992; Vapnik and Bottou, 1993). We have the paradoxical situation that seemingly global model structures (e.g., polynomials, multilayer sigmoidal neural nets) are being called local models because of the locally weighted training criterion. All of the data can be involved in training the local model, as long as distant data matters less than nearby data.

This paper explores locally weighted training procedures, which involves deferring processing the training data until a query is present, leading to the use of the terms *lazy learning* and *least commitment learning*. There are many global approaches and representations such as: rules, decision trees, and parametric models (e.g., polynomials, sigmoidal neural nets, radial basis functions, projection pursuit networks). All of the above approaches can be transformed into locally weighted approaches by using a locally weighted training criterion (Vapnik, 1992; Bottou and Vapnik, 1992; Vapnik and Bottou, 1993; Kozek, 1992), so the scope of locally weighted learning is quite broad. We will discuss locally weighted classification as an example.

16.2 Locally Weighted Classification

In classification, there are several ways to incorporate distance weighting. In k-nearest neighbor approaches, the number of occurrences of each class in the k closest points to the query are counted, and the class with the most occurrences (or votes) is predicted. Distance weighting could be used to weight the votes, so that nearby data points receive

more votes than distant points.

A second way to incorporate distance weighting in classifier training is to incorporate it into the cost criterion that is being minimized by training (Vapnik, 1992; Bottou and Vapnik, 1992; Vapnik and Bottou, 1993):

$$C(\mathbf{q}) = \sum_i [L(\hat{c}_i, c_{\text{true}i}) K(d(\mathbf{x}_i, \mathbf{q}))] \tag{74}$$

C is the cost to be minimized and $L(\hat{c}_i, c_{\text{true}i})$ is the cost of predicting class $\hat{c}_i$ on training point $\mathbf{x}_i$ when the true class is $c_{\text{true}i}$. $K()$ is the weighting or kernel function. A simple version of this approach is to select the k nearest points and just train a classifier on that data. In this case $K()$ is a uniform or boxcar kernel. The form of the classifier is not constrained in any way. Locally weighted learning specifies the form of the training criterion only, and not the form of the performance algorithm.

A third way to incorporate distance weighting is to treat classification as a regression problem, where there are decision functions for each class, and the decision function with the largest value at the query point determines the class of the query. Training these decision functions can be distance weighted as well:

$$C(\mathbf{q}) = \sum_i \left[\left(\sum_j (g_j(\mathbf{x}_i) - t_{ij})^2 \right) K(d(\mathbf{x}_i, \mathbf{q})) \right] \tag{75}$$

where $g_j()$ is the decision function for class j, and t_{ij} is the target for decision function $g_j()$ on training point i. Hastie and Tibshirani (1994) describe an approach in which global approaches to finding discriminants are localized by locally weighting the algorithm directly, rather than the criterion.

In this paper we described fitting simple linear models using distance weighted fit criterion. One can imagine using distance weighted criterion to train linear decision functions and linear discriminants to create local classifiers. It is also possible to train general models, such as logistic regression, to perform classification in a locally weighted fashion.

16.3 Requirements for Locally Weighted Learning

Locally weighted learning has several requirements:

- **Distance function:** Locally weighted learning systems require a measure of relevance. The major assumption made by locally weighted learning is that relevance can be measured using a measure of distance. Nearby training points are more relevant. There are many other possible measures of relevance, and also more general notions of similarity. The distance function $d(a, b)$ needs to input two objects and produce a number. The distance function does not need to satisfy the formal requirements for a distance metric.

- **Separable criterion:** Locally weighted learning systems compute a weight for each training point. To apply this weight, the training criterion cannot be a general function of the predictions of the training points:

$$C = L(\hat{y}_1, y_1, \hat{y}_2, y_2, ..., \hat{y}_n, y_n,) \tag{76}$$

but must be separable in some way. We use additive separability:

$$C = \sum_i [L(\hat{y}_i, y_i) K(d(\mathbf{x}_i, \mathbf{q}))] \tag{77}$$

although there are other forms of separability.

- **Enough data:** There needs to be enough data to compute statistics, which is also true of other statistical learning approaches. How much is enough? We have run robot learning experiments where performance improvements started to occur with on the order of ten points in the training set, although we typically collect between 100 and 1000 points during an experiment. The amount of training data needed is highly problem dependent.

- **Labelled data:** Each training point needs to have an associated output y_i. For classification this is a label, and for regression (function approximation) it is a number.

- **Representations:** Although the above requirements are enough for a system using nearest neighbor techniques, locally weighted regression requires that each object produces a fixed length vector of the values (symbolic or numeric) for a list of specified features:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{78}$$

However, more general representations can be handled by locally weighted learning approaches. For example, a more general training criterion is:

$$C = \sum_i \{L(f(X_i, \beta), Y_i) K(d(X_i, Q))\} \tag{79}$$

The inputs X_i, outputs Y_i, and query Q can be complex objects such as entire semantic networks, with the distance functions being graph matching algorithms or graph difference measuring algorithms, and $f()$ being a graph transformation with β as adjustable parameters (Elliot and Scott, 1991). Or the objects can be text computer files, with the inputs X in Japanese and the outputs Y in English, the distance functions can be the number of characters in the output of a file difference program such as the UNIX `diff`, and the local model $f()$ can be a machine translation program with adjustable parameters β. Typical parameters for an expert system might be strengths of rules, so changing β affects which rules are selected for application.

The input space distance $d()$ can be generalized to take into account the output space distance between the output values of the training data and a predicted output:

$$C = \sum_i \left\{ L(f(X_i, \beta), Y_i) K \left(d \left(\begin{pmatrix} X_i \\ Y_i \end{pmatrix}, \begin{pmatrix} Q \\ Y_{pred} \end{pmatrix} \right) \right) \right\} \tag{80}$$

This is useful when the function being approximated has several distinct outputs for similar inputs.

Although it has not yet been extensively explored by current research, it is possible for locally weighted learning systems to have stored objects that provide separate information to the query distance function (X_i) and to the local model ($\mathcal{X}_i$) (Hammond, 1991; Callan et al., 1991; Nguyen et al., 1993). In this case the training criterion might be:

$$C = \sum_i \{L(f(\mathcal{X}_i, \beta), Y_i) K(d(X_i, Q))\} \tag{81}$$

One example of this is to use measures of volatility of the stock market to measure distance between data points and a query $d(X_i, Q)$, but use price histories and other factors to form local (with respect to volatility) predictive models for future prices $f(\mathcal{X}_i, \beta)$ (LeBaron, 1990, 1992). Another example is to use nationality as the input to the distance function (requiring a distance calculation for symbolic values), and to use numeric features such as age, height, weight, and blood pressure to build a locally (with respect to the nationality distance) weighted regression to predict heart attack risk.

16.4 Future Research Directions

Our view of interesting areas of future research include:

- **Hybrid Tuning Algorithms:** We have developed independent continuous and discrete fit parameter optimization techniques. It is clear that a hybrid approach can do better than either approach alone. Parameters could initially be treated as discrete, and then more and more continuous optimization could be performed as optimal values were approached, for example. Another approach is for the racing algorithms to allow continuous tuning by each contestant during the race, rather than racing fixed sets of parameters.

- **New forms of local tuning:** So far research has focused on locally tuning bandwidth and smoothing parameters. More work needs to be done on locally tuning distance metrics, ridge regression parameters, outlier thresholds, etc., without overfitting.

- **Multiscale local tuning:** One dimensional fit parameters such as bandwidth and model order can be locally optimized using small neighborhoods. Multidimensional fit parameters such as the distance scale parameters in a distance matrix $\mathbf{M}$ or the set of ridge regression parameters need much larger neighborhoods and different kinds of regularization to be tuned locally. How should these different tuning processes interact?

- **Stochastic gradient approaches to continuous tuning:** Continuous optimization based on estimates of the gradient using small numbers of random queries rather than exhaustive query sets seems a promising approach to efficient tuning algorithms (Moore and Schneider, 1995).

- **Properties of massive cross-validation:** We have discussed the use of cross-validation, and why locally weighted learning is particularly well suited to its use. Better understanding of how much cross validation can take place before it is in danger of overfitting (which must be guarded by an extra level of cross-validation) would be desirable.

- **Probabilistic approaches:** We would like to explore further the analogies between locally weighted learning and probabilistic models, including Bayesian models (Rachlin et al., 1994; Ting and Cameron-Jones, 1994).

- **Forgetting:** So far, forgetting has not played an important role in our implementations of robot learning, as we have not run out of memory. However, we expect forgetting to play a more important role in the future, and expect it to be necessary to implement a principled approach to storage control.

- **Computational Techniques:** For enormous dataset sizes, new data management algorithms may be needed. They include principled ways to forget or coalesce old data, compactly represent high dimensional data clouds, ways of using samples of datasets instead of entire datasets, and, in the case of multi-gigabyte datasets hardware and software techniques for managing data on secondary storage.

- **Less Lazy Learning:** This review has focussed on a pure form of lazy learning, in which only the data is stored between queries. This purist approach will be too extreme in some circumstances, and most tuning algorithms for fit parameters store the optimized fit parameters in between queries. Substantial amounts of data compression can be achieved by building a set of local models at fixed locations, using the techniques described in this paper. In addition to computational speedup in the presence of large datasets there may be statistical advantages to compressing data instead of merely storing it all (Fritzke, 1995; Schaal and Atkeson, 1995).

17 Summary

This paper has surveyed locally weighted learning. Local weighting, whether by weighting the data or the error criterion, can turn global function approximation into powerful alternative approaches. By means of local weighting, unnecessary bias of global function fitting is reduced, higher flexibility is obtained, but desirable properties like smoothness and statistical analyzability are retained. We have concentrated on how linear regression behaves under local weighting, and surveyed the ways in which tools from conventional regression analysis in global regression can be used in locally weighted regression. A major question has concerned the notion of locality: what is a good choice of distance metric, how close within that metric should points be and how can these decisions be automatically made from the data. The field of local learning is of large interest in the statistics community, and we have provided entry points into that literature. Locally weighted learning is also rapidly increasing in popularity in the machine learning community and the outlook is promising for interesting statistical, computational and application-oriented development.

18 Acknowledgments

Support for C. Atkeson and S. Schaal was provided by the ATR Human Information Processing Research Laboratories. Support for C. Atkeson was provided under Air Force Office of Scientific Research grant F49-6209410362, and by a National Science Foundation Presidential Young Investigator Award. Support for S. Schaal was provided by the German Scholarship Foundation and the Alexander von Humboldt Foundation. Support for A. Moore was provided by the U. K. Science and Engineering Research Council, NSF Research Initiation Award # IRI-9409912, and a Research Gift from the 3M Corporation.

References

AAAI-9 (1991). *Ninth National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, Cambridge, MA.

Aha, D. W. (1989). Incremental, instance-based learning of independent and graded concept descriptions. In *Sixth International Machine Learning Workshop*, pages 387–391. Morgan Kaufmann, San Mateo, CA.

Aha, D. W. (1990). *A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Observations*. PhD dissertation, University of California, Irvine, Department of Information and Computer Science.

Aha, D. W. (1991). Incremental constructive induction: An instance-based approach. In *Eighth International Machine Learning Workshop*. Morgan Kaufmann, San Mateo, CA.

Aha, D. W. and Goldstone, R. L. (1990). Learning attribute relevance in context in instance-based learning algorithms. In *12th Annual Conference of the Cognitive Science Society*, pages 141–148.

Aha, D. W. and Goldstone, R. L. (1992). Concept learning and flexible weighting. In *14th Annual Conference of the Cognitive Science Society*, pages 534–539, Bloomington, IL. Lawrence Erlbaum Associates, Mahwah, NJ.

Aha, D. W. and Kibler, D. (1989). Noise-tolerant instance-based learning algorithms. In *Eleventh International Joint Conference on Artificial Intelligence*, pages 794–799. Morgan Kaufmann, San Mateo, CA.

Aha, D. W. and McNulty, D. M. (1989). Learning relative attribute weights for instance-based concept descriptions. In *11th Annual Conference of the Cognitive Science Society*, pages 530–537. Lawrence Erlbaum Associates, Mahwah, NJ.

Aha, D. W. and Salzberg, S. L. (1993). Learning to catch: Applying nearest neighbor algorithms to dyna mic control tasks. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pages 363–368, Ft. Lauderdale, FL.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Atkeson, C. G. (1990). Using local models to control movement. In Touretzky, D. S., editor, *Advances In Neural Information Processing Systems 2*, pages 316–323. Morgan Kaufman, San Mateo, CA.

Atkeson, C. G. (1992). Memory-based approaches to approximating continuous functions. In Casdagli and Eubank (1992), pages 503–521. Proceedings of a Workshop on Nonlinear Modeling and Forecasting September 17-21, 1990, Santa Fe, New Mexico.

Atkeson, C. G. (1996). Local learning. http://www.cc.gatech.edu/fac/Chris.Atkeson/local-learning/.

Atkeson, C. G., Moore, A. W., and Schaal, S. (1996). Locally weighted learning for control. *Artificial Intelligence Review*. in press.

Atkeson, C. G. and Reinkensmeyer, D. J. (1988). Using associative content-addressable memories to control robots. In *Proceedings of the 27th IEEE Conference on Decision and Control*, volume 1, pages 792–797, Austin, Texas. IEEE Cat. No.88CH2531-2.

Atkeson, C. G. and Reinkensmeyer, D. J. (1989). Using associative content-addressable memories to control robots. In *Proceedings, IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona.

Atkeson, C. G. and Schaal, S. (1995). Memory-based neural networks for robot learning. *Neurocomputing*, 9:243–269.

Baird, L. C. and Klopf, A. H. (1993). Reinforcement learning with high-dimensional, continuous actions. Technical Report WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base Ohio. http://kirk.usafa.af.mil/ baird/papers/index.html.

Barnhill, R. E. (1977). Representation and approximation of surfaces. In Rice, J. R., editor, *Mathematical Software III*, pages 69–120. Academic Press, New York, NY.

Batchelor, B. G. (1974). *Practical Approach To Pattern Classification*. Plenum Press, New York, NY.

Benedetti, J. K. (1977). On the nonparametric estimation of regression functions. *Journal of the Royal Statistical Society, Series B*, 39:248–253.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.

Bentley, J. L. and Friedman, J. H. (1979). Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409.

Bentley, J. L., Weide, B., and Yao, A. (1980). Optimal expected time algorithms for closest point problems. *ACM Transactions on Mathematical Software*, 6:563–580.

Blyth, S. (1993). Optimal kernel weights under a power criterion. *Journal of the American Statistical Association*, 88(424):1284–1286.

Bottou, L. and Vapnik, V. (1992). Local learning algorithms. *Neural Computation*, 4(6):888–900.

Bregler, C. and Omohundro, S. M. (1994). Surface learning with applications to lipreading. In Cowan et al. (1994), pages 43–50.

Brockmann, M., Gasser, T., and Herrmann, E. (1993). Locally adaptive bandwidth choice for kernel regression estimators. *Journal of the American Statistical Association*, 88(424):1302–1309.

Broder, A. J. (1990). Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23:171–178.

Callan, J. P., Fawcett, T. E., and Rissland, E. L. (1991). CABOT: An adaptive approach to case based search. In IJCAI 12 (1991), pages 803–808.

Casdagli, M. and Eubank, S., editors (1992). *Nonlinear Modeling and Forecasting*. Proceedings Volume XII in the Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley, New York, NY. Proceedings of a Workshop on Nonlinear Modeling and Forecasting September 17-21, 1990, Santa Fe, New Mexico.

Cheng, P. E. (1984). Strong consistency of nearest neighbor regression function estimators. *Journal of Multivariate Analysis*, 15:63–72.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836.

Cleveland, W. S. (1993a). Coplots, nonparametric regression, and conditionally parametric fits. Technical Report 19, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ. http://netlib.att.com/netlib/att/stat/doc/.

Cleveland, W. S. (1993b). *Visualizing Data*. Hobart Press, Summit, NJ. books@hobart.com.

Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610.

Cleveland, W. S., Devlin, S. J., and Grosse, E. (1988). Regression by local fitting: Methods, properties, and computational algorithms. *Journal of Econometrics*, 37:87–114.

Cleveland, W. S. and Grosse, E. (1991). Computational methods for local regression. *Statistics and Computing*, 1(1):47–62. ftp://cm.bell-labs.com/cm/cs/doc/91/4-04.ps.gz.

Cleveland, W. S., Grosse, E., and Shyu, W. M. (1992). Local regression models. In Chambers, J. M. and Hastie, T. J., editors, *Statistical Models in S*, pages 309–376. Wadsworth, Pacific Grove, CA. http://netlib.att.com/netlib/a/cloess.ps.Z.

Cleveland, W. S. and Loader, C. (1994a). Computational methods for local regression. Technical Report 11, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ. http://netlib.att.com/netlib/att/stat/doc/.

Cleveland, W. S. and Loader, C. (1994b). Local fitting for semiparametric (nonparametric) regression: Comments on a paper of Fan and Marron. Technical Report 8, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ. http://netlib.att.com/netlib/att/stat/doc/, 94.8.ps, earlier version is

94.3.ps.

Cleveland, W. S. and Loader, C. (1994c). Smoothing by local regression: Principles and methods. Technical Report 95.3, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ. http://netlib.att.com/netlib/att/stat/doc/.

Cleveland, W. S., Mallows, C. L., and McRae, J. E. (1993). ATS methods: Nonparametric regression for non-Gaussian data. *Journal of the American Statistical Association*, 88(423):821–835.

Connell, M. E. and Utgoff, P. E. (1987). Learning to control a dynamic physical system. In *Sixth National Conference on Artificial Intelligence*, pages 456–460, Seattle, WA. Morgan Kaufmann, San Mateo, CA.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78.

Coughran, Jr., W. M. and Grosse, E. (1991). Seeing and hearing dynamic loess surfaces. In *Interface'91 Proceedings*, pages 224–228. Springer-Verlag. ftp://cm.bell-labs.com/cm/cs/doc/91/4-07.ps.gz or 4-07long.ps.gz.

Cowan, J. D., Tesauro, G., and Alspector, J., editors (1994). *Advances In Neural Information Processing Systems 6*. Morgan Kaufman, San Mateo, CA.

Crain, I. K. and Bhattacharyya, B. K. (1967). Treatment of nonequispaced two dimensional data with a digital computer. *Geoexploration*, 5:173–194.

Deheuvels, P. (1977). Estimation non-paramétrique del la densité par histogrammes généralisés. *Revue Statistique Appliqué*, 25:5–42.

Deng, K. and Moore, A. W. (1994). Multiresolution instance-based learning. In *Fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981). An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software*, 7(3):369–383.

Devroye, L. (1981). On the almost everywhere convergence of nonparametric regression function estimates. *The Annals of Statistics*, 9(6):1310–1319.

Diebold, F. X. and Nason, J. A. (1990). Nonparametric exchange rate prediction? *Journal of International Economics*, 28(3-4):315–332.

Dietterich, T. G., Wettschereck, D., Atkeson, C. G., and Moore, A. W. (1994). Memory-based methods for regression and classification. In Cowan et al. (1994), pages 1165–1166.

Draper, N. R. and Smith, H. (1981). *Applied Regression Analysis*. John Wiley, New York, NY, 2nd edition.

Elliot, T. and Scott, P. D. (1991). Instance-based and generalization-based learning procedures applied to solving integration problems. In *Proceedings of the Eighth Conference of the Society for the Study of Artificial Intelligence*, pages 256–265, Leeds, England. Springer Verlag.

Epanechnikov, V. A. (1969). Nonparametric estimation of a multivariate probability density. *Theory of Probability and Its Applications*, 14:153–158.

Eubank, R. L. (1988). *Spline Smoothing and Nonparametric Regression*. Marcel Dekker, New York, NY.

Falconer, K. J. (1971). A general purpose algorithm for contouring over scattered data points. Technical Report NAC 6, National Physical Laboratory, Teddington, Middlesex, United Kingdon, TW11 0LW.

Fan, J. (1992). Design-adaptive nonparametric regression. *Journal of the American Statistical Association*, 87(420):998–1004.

Fan, J. (1993). Local linear regression smoothers and their minimax efficiencies. *Annals of Statistics*, 21:196–216.

Fan, J. (1995). Local modeling. EES Update: written for the Encyclopedia of Statistics Science, http://www.stat.unc.edu/faculty/fan/papers.html.

Fan, J. and Gijbels, I. (1992). Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, 20(4):2008–2036.

Fan, J. and Gijbels, I. (1994). Censored regression: Local linear approximations and their applications. *Journal of the American Statistical Association*, 89:560–570.

Fan, J. and Gijbels, I. (1995a). Adaptive order polynomial fitting: Bandwidth robustification and bias reduction.

Fan, J. and Gijbels, I. (1995b). Data-driven bandwidth selection in local polynomial fitting: Variable bandwidth and spatial adaptation. *Journal of the Royal Statistical Society, Series B*, 57:371–394.

Fan, J. and Gijbels, I. (1996). *Local Polynomial Modeling and its Applications*. Chapman and Hall,

London.

Fan, J. and Hall, P. (1994). On curve estimation by minimizing mean absolute deviation and its implications. *The Annals of Statistics*, 22(2):867–885.

Fan, J. and Kreutzberger, E. (1995). Automatic local smoothing for spectral density estimation. ftp://stat.unc.edu/pub/fan/spec.ps.

Fan, J. and Marron, J. S. (1993). Comment on [Hastie and Loader, 1993]. *Statistical Science*, 8(2):129–134.

Fan, J. and Marron, J. S. (1994a). Fast implementations of nonparametric curve estimators. *Journal of Computational and Graphical Statistics*, 3:35–56.

Fan, J. and Marron, J. S. (1994b). Rejoinder to discussion of Cleveland and Loader.

Farmer, J. D. and Sidorowich, J. J. (1987). Predicting chaotic time series. *Physical Review Letters*, 59(8):845–848.

Farmer, J. D. and Sidorowich, J. J. (1988a). Exploiting chaos to predict the future and reduce noise. In Lee, Y. C., editor, *Evolution, Learning, and Cognition*, pages 277–??? World Scientific Press, NJ. also available as Technical Report LA-UR-88-901, Los Alamos National Laboratory, Los Alamos, New Mexico.

Farmer, J. D. and Sidorowich, J. J. (1988b). Predicting chaotic dynamics. In Kelso, J. A. S., Mandell, A. J., and Schlesinger, M. F., editors, *Dynamic Patterns in Complex Systems*, pages 265–292. World Scientific, NJ.

Farwig, R. (1987). Multivariate interpolation of scattered data by moving least squares methods. In Mason, J. C. and Cox, M. G., editors, *Algorithms for Approximation*, pages 193–211. Clarendon Press, Oxford.

Fedorov, V. V., Hackl, P., and Müller, W. G. (1993). Moving local regression: The weight function. *Nonparametric Statistics*, 2(4):355–368.

Franke, R. and Nielson, G. (1980). Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15:1691–1704.

Friedman, J. H. (1984). A variable span smoother. Technical Report LCS 5, Stanford University, Statistics Department, Stanford, CA.

Friedman, J. H. (1994). Flexible metric nearest neighbor classification. http://playfair.stanford.edu/reports/friedman/.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226.

Fritzke, B. (1995). Incremental learning of local linear mappings. In *Proceedings of the International Conference on Artificial Neural Networks ICANN '95*, pages 217–222, Paris, France.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press, New York, NY, second edition.

Gasser, T. and Müller, H. G. (1979). Kernel estimation of regression functions. In Gasser, T. and Rosenblatt, M., editors, *Smoothing Techniques for Curve Estimation*, number 757 in Lecture Notes in Mathematics, pages 23–67. Springer-Verlag, Heidelberg.

Gasser, T. and Müller, H. G. (1984). Estimating regression functions and their derivatives by the kernel method. *Scandanavian Journal of Statistics*, 11:171–185.

Gasser, T., Müller, H. G., and Mammitzsch, V. (1985). Kernels for nonparametric regression. *Journal of the Royal Statistical Society, Series B*, 47:238–252.

Ge, Z., Cavinato, A. G., and Callis, J. B. (1994). Noninvasive spectroscopy for monitoring cell density in a fermentation process. *Analytical Chemistry*, 66:1354–1362.

Goldberg, K. Y. and Pearlmutter, B. (1988). Using a neural network to learn the dynamics of the CMU Direct-Drive Arm II. Technical Report CMU-CS-88-160, Carnegie-Mellon University, Pittsburgh, PA.

Gorinevsky, D. and Connolly, T. H. (1994). Comparison of some neural network and scattered data approximations: The inverse manipulator kinematics example. *Neural Computation*, 6:521–542.

Goshtasby, A. (1988). Image registration by local approximation methods. *Image and Vision Computing*, 6(4):255–261.

Grosse, E. (1989). LOESS: Multivariate smoothing by moving least squares. In Chui, C. K., Schumaker, L. L., and Ward, J. D., editors, *Approximation Theory VI*, pages 1–4. Academic Press, Boston, MA.

Hammond, S. V. (1991). Nir analysis of antibiotic fermentations. In Murray, I. and Cowe, I. A., editors,

Making Light Work: Advances in Near Infrared Spectroscopy, pages 584–589. VCH: New York, NY. Developed from the 4th International Conference on Near Infrared Spectroscopy, Aberdeen, Scotland, August 19-23, 1991.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). *Robust Statistics: The Approach Based On Influence Functions*. John Wiley, New York, NY.

Härdle, W. (1990). *Applied Nonparametric Regression*. Cambridge University Press, New York, NY.

Hastie, T. and Loader, C. (1993). Local regression: Automatic kernel carpentry. *Statistical Science*, 8(2):120–143.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Regression*. Chapman Hall, London.

Hastie, T. J. and Tibshirani, R. J. (1994). Discriminant adaptive nearest neighbor classification. ftp://playfair.Stanford.EDU/pub/hastie/dann.ps.Z.

Higuchi, T., Kitano, H., Furuya, T., Handa, K., Takahashi, N., and Kokubu, A. (1991). IXM2: A parallel associative processor for knowledge processing. In AAAI-9 (1991), pages 296–303.

Hillis, D. (1985). *The Connection Machine*. MIT Press, Cambridge, MA.

Huang, P. S. (1996). *Planning For Dynamic Motions Using A Search Tree*. MS thesis, University of Toronto, Graduate Department of Computer Science. http://www.dgp.utoronto.ca/people/psh/home.html.

IJCAI 12 (1991). *Twelfth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.

IJCAI 13 (1993). *Thirteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA.

Jabbour, K., Riveros, J. F. W., Landsbergen, D., and Meyer, W. (1987). ALFA: Automated load forecasting assistant. In *Proceedings of the 1987 IEEE Power Engineering Society Summer Meeting*, San Francisco, CA.

James, M. (1985). *Classification Algorithms*. John Wiley and Sons, New York, NY.

Jones, M. C., Davies, S. J., and Park, B. U. (1994). Versions of kernel-type regression estimators. *Journal of the American Statistical Association*, 89(427):825–832.

Karalič, A. (1992). Employing linear regression in regression tree leaves. In Neumann, B., editor, *ECAI 92: 10th European Conference on Artificial Intelligence*, pages 440–441, Vienna, Austria. John Wiley and Sons.

Katkovnik, V. Y. (1979). Linear and nonlinear methods of nonparametric regression analysis. *Soviet Automatic Control*, 5:25–34.

Kazmierczak, H. and Steinbuch, K. (1963). Adaptive systems in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-12:822–835.

Kibler, D., Aha, D. W., and Albert, M. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence*, 5:51–57.

Kitano, H. (1993a). Challenges of massive parallelism. In IJCAI 13 (1993), pages 813–834.

Kitano, H. (1993b). A comprehensive and practical model of memory-based machine translation. In IJCAI 13 (1993), pages 1276–1282.

Kitano, H. and Higuchi, T. (1991a). High performance memory-based translation on IXM2 massively parallel associative memory processor. In AAAI-9 (1991), pages 149–154.

Kitano, H. and Higuchi, T. (1991b). Massively parallel memory-based parsing. In IJCAI 12 (1991), pages 918–924.

Kitano, H., Moldovan, D., and Cha, S. (1991). High performance natural language processing on semantic network array processor. In IJCAI 12 (1991), pages 911–917.

Kozek, A. S. (1992). A new nonparametric estimation method: Local and nonlinear. *Interface*, 24:389–393.

Lancaster, P. (1979). Moving weighted least-squares methods. In Sahney, B. N., editor, *Polynomial and Spline Approximation*, pages 103–120. D. Reidel Publishing, Boston, MA.

Lancaster, P. and Šalkauskas, K. (1981). Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158.

Lancaster, P. and Šalkauskas, K. (1986). *Curve And Surface Fitting*. Academic Press, New York, NY.

Lawrence, S., Tsoi, A. C., and Black, A. D. (1996). Function approximation with neural networks and local methods: Bias, variance and smoothness. In *Australian Conference on Neural Networks, Canberra,*

Australia, Canberra, Australia. available from http://www.neci.nj.nec.com/homepages/lawrence and http://www.elec.uq.edu.au/~lawrence.

LeBaron, B. (1990). Forecast improvements using a volatility index. unpublished.

LeBaron, B. (1992). Nonlinear forecasts for the S&P stock index. In Casdagli and Eubank (1992), pages 381–393. Proceedings of a Workshop on Nonlinear Modeling and Forecasting September 17-21, 1990, Santa Fe, New Mexico.

Legg, M. P. C. and Brent, R. P. (1969). Automatic contouring. In *4th Australian Computer Conference*, pages 467–468.

Lejeune, M. (1984). Optimization in non-parametric regression. In *COMPSTAT 1984: Proceedings in Computational Statistics*, pages 421–426, Prague. Physica-Verlag Wien.

Lejeune, M. (1985). Estimation non-paramétrique par noyaux: Régression polynômial mobile. *Revue de Statistique Appliquée*, 23(3):43–67.

Lejeune, M. and Sarda, P. (1992). Smooth estimators of distribution and density functions. *Computational Statistics & Data Analysis*, 14:457–471.

Li, K. C. (1984). Consistency for cross-validated nearest neighbor estimates in nonparametric regression. *The Annals of Statistics*, 12:230–240.

Loader, C. (1994). Computing nonparametric function estimates. Technical Report 7, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ. Available by anonymous FTP from netlib.att.com in /netlib/att/stat/doc/94/7.ps.

Lodwick, G. D. and Whittle, J. (1970). A technique for automatic contouring field survey data. *Australian Computer Journal*, 2:104–109.

Lowe, D. G. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7:72–85.

Maron, O. and Moore, A. W. (1996). A racing algorithm: Model selection for memory based learners. *Artificial Intelligence Review*. in press.

Marron, J. S. (1988). Automatic smoothing parameter selection: A survey. *Empirical Economics*, 13:187–208.

McCallum, R. A. (1995). Instance-based utile distinctions for reinforcement learning with hidden state. In Prieditis and Russell (1995), pages 387–395.

McIntyre, D. B., Pollard, D. D., and Smith, R. (1968). Computer programs for automatic contouring. Technical Report Kansas Geological Survey Computer Contributions 23, University of Kansas, Lawrence, KA.

McLain, D. H. (1974). Drawing contours from arbitrary data points. *The Computer Journal*, 17(4):318–324.

Medin, D. L. and Shoben, E. J. (1988). Context and structure in conceptual combination. *Cognitive Psychology*, 20:158–190.

Meese, R. and Wallace, N. (1991). Nonparametric estimation of dynamic hedonic price models and the construction of residential housing price indices. *American Real Estate and Urban Economics Association Journal*, 19(3):308–332.

Meese, R. A. and Rose, A. K. (1990). Nonlinear, nonparametric, nonessential exchange rate estimation. *The American Economic Review*, May:192–196.

Miller, A. J. (1990). *Subset Selection in Regression*. Chapman and Hall, London.

Miller, W. T., Glanz, F. H., and Kraft, L. G. (1987). Application of a general learning algorithm to the control of robotic manipulators. *International Journal of Robotics Research*, 6:84–98.

Mohri, T. and Tanaka, H. (1994). An optimal weighting criterion of case indexing for both numeric and symbolic attributes. In Aha, D. W., editor, *AAAI-94 Workshop Program: Case-Based Reasoning, Working Notes*, pages 123–127. AAAI Press, Seattle, WA.

Moore, A. W. (1990a). Acquisition of Dynamic Control Knowledge for a Robotic Manipulator. In *Seventh International Machine Learning Workshop*. Morgan Kaufmann, San Mateo, CA.

Moore, A. W. (1990b). Efficient Memory-based Learning for Robot Control. PhD. Thesis; Technical Report No. 209, Computer Laboratory, University of Cambridge.

Moore, A. W., Hill, D. J., and Johnson, M. P. (1992). An empirical investigation of brute force to choose features, smoothers, and function approximators. In Hanson, S., Judd, S., and Petsche, T., editors, *Computational Learning Theory and Natural Learning Systems*, volume 3. MIT Press, Cambridge,

MA.

Moore, A. W. and Schneider, J. (1995). Memory-based stochastic optimization. To appear in the proceedings of NIPS-95, Also available as Technical Report CMU-RI-TR-95-30, ftp://ftp.cs.cmu.edu/afs/cs.cmu.edu/project/reinforcement/papers/memstoch.ps.

More, J. J., Garbow, B. S., and Hillstrom, K. E. (1980). User guide for MINPACK-1. Technical Report ANL-80-74, Argonne National Laboratory, Argonne, Illinois.

Müller, H.-G. (1987). Weighted local regression and kernel methods for nonparametric curve fitting. *Journal of the American Statistical Association*, 82:231–238.

Müller, H.-G. (1993). Comment on [Hastie and Loader, 1993]. *Statistical Science*, 8(2):134–139.

Murphy, O. J. and Selkow, S. M. (1986). The efficiency of using k-d trees for finding nearest neighbors in discrete space. *Information Processing Letters*, 23:215–218.

Myers, R. H. (1990). *Classical and Modern Regression With Applications*. PWS-KENT, Boston, MA.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability and Its Applications*, 9:141–142.

Næs, T. and Isaksson, T. (1992). Locally weighted regression in diffuse near-infrared transmittance spectroscopy. *Applied Spectroscopy*, 46(1):34–43.

Næs, T., Isaksson, T., and Kowalski, B. R. (1990). Locally weighted regression and scatter correction for near-infrared reflectance data. *Analytical Chemistry*, 62(7):664–673.

Nguyen, T., Czerwinksi, M., and Lee, D. (1993). COMPAQ Quicksource: Providing the consumer with the power of artificial intelligence. In *Proceedings of the Fifth Annual Conference on Innovative Applications of Artificial Intelligence*, pages 142–150, Washington, DC. AAAI Press.

Nosofsky, R. M., Clark, S. E., and Shin, H. J. (1989). Rules and exemplars in categorization, identification, and recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15:282–304.

Omohundro, S. M. (1987). Efficient Algorithms with Neural Network Behaviour. *Journal of Complex Systems*, 1(2):273–347.

Omohundro, S. M. (1991). Bumptrees for Efficient Function, Constraint, and Classification Learning. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann.

Palmer, J. A. B. (1969). Automatic mapping. In *4th Australian Computer Conference*, pages 463–466.

Pelto, C. R., Elkins, T. A., and Boyd, H. A. (1968). Automatic contouring of irregularly spaced data. *Geophysics*, 33:424–430.

Peng, J. (1995). Efficient memory-based dynamic programming. In Prieditis and Russell (1995), pages 438–446.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1988). *Numerical Recipes in C*. Cambridge University Press, New York, NY.

Prieditis, A. and Russell, S., editors (1995). *Twelfth International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Rachlin, J., Kasif, S., Salzberg, S., and Aha, D. W. (1994). Towards a better understanding of memory-based reasoning systems. In Cohen, W. W. and Hirsh, H., editors, *Eleventh International Conference on Machine Learning*, pages 242–250. Morgan Kaufmann, San Mateo, CA. now updated and submitted as an Artificial Intelligence Journal technical note: Kasif, S., Salzberg, S., Waltz, D., Rachlin, J., & Aha, D. W. (1995). "Towards a framework for memory-based reasoning".

Racine, J. (1993). An efficient cross-validation algorithm for window width selection for nonparametric kernel regression. *Communications in Statistics: Simulation and Computation*, 22(4):1107–1114.

Ramasubramanian, V. and Paliwal, K. K. (1989). A generalized optimization of the k-d tree for fast nearest-neighbour search. In *International Conference on Acoustics, Speech, and Signal Processing*.

Raz, J., Turetsky, B. I., and Fein, G. (1989). Selecting the smoothing parameter for estimation of smoothly changing evoked potential signals. *Biometrics*, 45:851–871.

Renka, R. J. (1988). Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software*, 14(2):139–152.

Ruppert, D. and Wand, M. P. (1994). Multivariate locally weighted least squares regression. *The Annals of Statistics*, 22(3):1346–1370.

Ruprecht, D. and Müller, H. (1992). Image warping with scattered data interpolation methods. Technical Report 443, Universität Dortmund, Fachbereich Informatik, D-44221 Dortmund, Germany. Available for anonymous FTP from ftp-ls7.informatik.uni-dortmund.de in pub/reports/ls7/rr-443.ps.Z.

Ruprecht, D. and Müller, H. (1993). Free form deformation with scattered data interpolation methods. In Farin, G., Hagen, H., and Noltemeier, H., editors, *Geometric Modelling (Computing Suppl. 8)*, pages 267–281. Springer Verlag. Available for anonymous FTP from ftp-ls7.informatik.uni-dortmund.de in pub/reports/iif/rr-41.ps.Z.

Ruprecht, D. and Müller, H. (1994a). Deformed cross-dissolves for image interpolation in scientific visualization. *The Journal of Visualization and Computer Animation*, 5(3):167–181. Available for anonymous FTP from ftp-ls7.informatik.uni-dortmund.de in pub/reports/ls7/rr-491.ps.Z.

Ruprecht, D. and Müller, H. (1994b). A framework for generalized scattered data interpolation. Technical Report 517, Universität Dortmund, Fachbereich Informatik, D-44221 Dortmund, Germany. Available for anonymous FTP from ftp-ls7.informatik.uni-dortmund.de in pub/reports/ls7/rr-517.ps.Z.

Ruprecht, D., Nagel, R., and Müller, H. (1994). Spatial free form deformation with scattered data interpolation methods. Technical Report 539, Fachbereich Informatik der Universität Dortmund, 44221 Dortmund, Germany. Accepted for publication by *Computers & Graphics*, Available for anonymous FTP from ftp-ls7.informatik.uni-dortmund.de in pub/reports/ls7/rr-539.ps.Z.

Rust, R. T. and Bornman, E. O. (1982). Distribution-free methods of approximating nonlinear marketing relationships. *Journal of Marketing Research*, XIX:372–374.

Sabin, M. A. (1980). Contouring – a review of methods for scattered data. In Brodlie, K., editor, *Mathematical Methods in Computer Graphics and Design*, pages 63–86. Academic Press, New York, NY.

Saitta, L., editor (1996). *Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA.

Schaal, S. and Atkeson, C. G. (1994). Assessing the quality of learned local models. In Cowan et al. (1994), pages 160–167.

Schaal, S. and Atkeson, C. G. (1995). From isolation to cooperation: An alternative view of a system of experts. NIPS95 proceedings, in press.

Scott, D. W. (1992). *Multivariate Density Estimation*. Wiley, New York, NY.

Seber, G. A. F. (1977). *Linear Regression Analysis*. John Wiley, New York, NY.

Seifert, B., Brockmann, M., Engel, J., and Gasser, T. (1994). Fast algorithms for nonparametric curve estimation. *Journal of Computational and Graphical Statistics*, 3(2):192–213.

Seifert, B. and Gasser, T. (1994). Variance properties of local polynomials. http://www.unizh.ch/biostat/manuscripts.html.

Shepard, D. (1968). A two-dimensional function for irregularly spaced data. In *23rd ACM National Conference*, pages 517–524.

Solow, A. R. (1988). Detecting changes through time in the variance of a long-term hemispheric temperature record: An application of robust locally weighted regression. *Journal of Climate*, 1:290–296.

Specht, D. E. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576.

Sproull, R. F. (1991). Refinements to nearest-neighbor searching in k-d trees. *Algorithmica*, 6:579–589.

Stanfill, C. (1987). Memory-based reasoning applied to English pronunciation. In *Sixth National Conference on Artificial Intelligence*, pages 577–581.

Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228.

Steinbuch, K. (1961). Die lernmatrix. *Kybernetik*, 1:36–45.

Steinbuch, K. and Piske, U. A. W. (1963). Learning matrices and their applications. *IEEE Transactions on Electronic Computers*, EC-12:846–862.

Stone, C. J. (1975). Nearest neighbor estimators of a nonlinear regression function. In *Computer Science and Statistics: 8th Annual Symposium on the Interface*, pages 413–418.

Stone, C. J. (1977). Consistent nonparametric regression. *The Annals of Statistics*, 5:595–645.

Stone, C. J. (1980). Optimal rates of convergence for nonparametric estimators. *The Annals of Statistics*, 8:1348–1360.

Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10(4):1040–1053.

Sumita, E., Oi, K., Furuse, O., Iida, H., Higuchi, T., Takahashi, N., and Kitano, H. (1993). Example-

based machine translation on massively parallel processors. In IJCAI 13 (1993), pages 1283–1288.

Tadepalli, P. and Ok, D. (1996). Scaling up average reward reinforcement learning by approximating the domain models and the value function. In Saitta (1996). http://www.cs.orst.edu:80/~tadepall/research/publications.html.

Tamada, T., Maruyama, M., Nakamura, Y., Abe, S., and Maeda, K. (1993). Water demand forecasting by memory based learning. *Water Science and Technology*, 28(11–12):133–140.

Taylor, W. K. (1959). Pattern recognition by means of automatic analogue apparatus. *Proceedings of The Institution of Electrical Engineers*, 106B:198–209.

Taylor, W. K. (1960). A parallel analogue reading machine. *Control*, 3:95–99.

Thorpe, S. (1995). Localized versus distributed representations. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 549–552. The MIT Press, Cambridge, MA.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems (NIPS) 8*. http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/thrun/publications.html.

Thrun, S. and O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC algorithm. In Saitta (1996). http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/thrun/publications.html.

Tibshirani, R. and Hastie, T. (1987). Local likelihood estimation. *Journal of the American Statistical Association*, 82:559–567.

Ting, K. M. and Cameron-Jones, R. M. (1994). Exploring a framework for instance based learning and naive Bayesian classifiers. In *Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence*, Armidale, Australia. World Scientific.

Tou, J. T. and Gonzalez, R. C. (1974). *Pattern Recognition Principles*. Addison-Wesley, Reading, MA.

Townshend, B. (1992). Nonlinear prediction of speech signals. In Casdagli and Eubank (1992), pages 433–453. Proceedings of a Workshop on Nonlinear Modeling and Forecasting September 17-21, 1990, Santa Fe, New Mexico.

Tsybakov, A. B. (1986). Robust reconstruction of functions by the local approximation method. *Problems of Information Transmission*, 22:133–146.

Tukey, J. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

Turetsky, B. I., Raz, J., and Fein, G. (1989). Estimation of trial-to-trial variation in evoked potential signals by smoothing across trials. *Psychophysiology*, 26(6):700–712.

Turlach, B. A. and Wand, M. P. (1995). Fast computation of auxiliary quantities in local polynomial regression. http://netec.wustl.edu/~adnetec/WoPEc/agsmst/agsmst95009.html.

van der Smagt, P., Groen, F., and van het Groenewoud, F. (1994). The locally linear nested network for robot manipulation. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 2787–2792. ftp://ftp.fwi.uva.nl/pub/computer-systems/aut-sys/reports/SmaGroGro94b.ps.gz.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances In Neural Information Processing Systems 4*, pages 831–838. Morgan Kaufman, San Mateo, CA.

Vapnik, V. and Bottou, L. (1993). Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 5(6):893–909.

Walden, A. T. and Prescott, P. (1983). Identification of trends in annual maximum sea levels using robust locally weighted regression. *Estuarine, Coastal and Shelf Science*, 16:17–26.

Walters, R. F. (1969). Contouring by machine: A user's guide. *American Association of Petroleum Geologists Bulletin*, 53(11):2324–2340.

Waltz, D. L. (1987). Applications of the Connection Machine. *Computer*, 20(1):85–97.

Wand, M. P. and Jones, M. C. (1993). Comparison of smoothing parameterizations in bivariate kernel density estimation. *Journal of the American Statistical Association*, 88:520–528.

Wand, M. P. and Jones, M. C. (1994). *Kernel Smoothing*. Chapman and Hall, London.

Wand, M. P. and Schucany, W. R. (1990). Gaussian-based kernels for curve estimation and window width selection. *Canadian Journal of Statistics*, 18:197–204.

Wang, Z., Isaksson, T., and Kowalski, B. R. (1994). New approach for distance measurement in locally weighted regression. *Analytical Chemistry*, 66(2):249–260.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 26:359–372.

Weisberg, S. (1985). *Applied Linear Regression*. John Wiley and Sons.

Wess, S., Althoff, K.-D., and Derwand, G. (1994). Using k-d trees to improve the retrieval step in case-based reasoning. In Wess, S., Althoff, K.-D., and Richter, M. M., editors, *Topics in Case-Based Reasoning*, pages 167–181. Springer-Verlag, New York, NY. Proceedings of the First European Workshop, EWCBR-93.

Wettschereck, D. (1994). *A Study Of Distance-Based Machine Learning Algorithms*. PhD dissertation, Oregon State University, Department of Computer Science, Corvalis, OR.

Wijnberg, L. and Johnson, T. (1985). Estimation of missing values in lead air quality data sets. In Johnson, T. R. and Penkala, S. J., editors, *Quality Assurance in Air Pollution Measurements*. Air Pollution Control Association, Pittsburgh, PA. TR-3: Transactions: An APCA International Specialty Conference.

Wolberg, G. (1990). *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA.

Yasunaga, M. and Kitano, H. (1993). Robustness of the memory-based reasoning implemented by wafer scale integration. *IEICE Transactions on Information and Systems*, E76-D(3):336–344.

Zografski, Z. (1989). *Neuromorphic, Algorithmic, and Logical Models for the Automatic Synthesis of Robot Action*. PhD dissertation, University of Ljubljana, Ljubljana, Slovenia, Yugoslavia.

Zografski, Z. (1991). New methods of machine learning for the construction of integrated neuromorphic and associative-memory knowledge bases. In Zajc, B. and Solina, F., editors, *Proceedings, 6th Mediterranean Electrotechnical Conference*, volume II, pages 1150–1153, Ljubljana, Slovenia, Yugoslavia. IEEE catalog number 91CH2964-5.

Zografski, Z. (1992). Geometric and neuromorphic learning for nonlinear modeling, control and forecasting. In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, pages 158–163, Glasgow, Scotland. IEEE catalog number 92CH3110-4.

Zografski, Z. and Durrani, T. (1995). Comparing predictions from neural networks and memory-based learning. In *Proceedings, ICANN '95/NEURONIMES '95: International Conference on Artificial Neural Networks*, pages 221–226, Paris, France.