Memory-Based Approaches to Approximating Continuous Functions

Christopher G. Atkeson
The Artificial Intelligence Laboratory and
the Brain and Cognitive Sciences Department
Massachusetts Institute of Technology
NE43-771, 545 Technology Square, Cambridge, MA 02139
cga@ai.mit.edu

December 20, 1994

Abstract: This paper explores the use of locally weighted regression in memory-based robot learning. A local model is formed to answer each query, using a weighted regression in which close points (similar experiences) are weighted more than distant points (less relevant experiences). This approach implements a philosophy of modeling a complex function with many simple local models. The paper explains how an appropriate distance metric or measure of similarity can be found, and how the distance metric is used. It also explains how irrelevant input variables and terms in the local model are detected. An example from the control of a robot arm is used to compare the performance of this approach with that of other approaches that have been used for robot control and learning.

1 Introduction

A common problem in motor learning is approximating a continuous function from samples of the function's inputs and outputs. This paper explores a memory-based algorithm that simply remembers experiences (samples) and builds a local model to answer any particular query (an input for which the function's output is desired). Our approach is to model complex functions using simple local models. This approach avoids the difficult problem of finding an appropriate structure for a global model. A key idea is to form a training set for the local model after a query to be answered is known. This approach allows us to include in the training set only relevant experiences (nearby samples) and to weight the experiences according to their relevance to the query. We form a local model of the portion of the function near the query point, much as a Taylor series models a function in a neighborhood of a point. This local model is then used to predict the output of the function for that query. After answering the query, the local model is discarded and a new local model is created to answer the next query.

The memory-based architecture can represent nonlinear functions which are smooth, yet has simple training rules with a single global optimum for building a local model in response to a query. This allows complex nonlinear models to be identified (trained) quickly. Currently we are using polynomials as the local models. Since the polynomial local models are linear in the unknown parameters, we can estimate these parameters using a linear regression. We use cross validation to choose an appropriate distance metric and weighting function, and to help find irrelevant input variables and terms in the local model. The local modeling approach minimizes interference between old and new data, and allows the range of generalization to depend on the density of the samples.

In order to explore the feasibility of the memory-based approach and identify research issues and problems, we used the approach to model and control a simulated planar two-joint robot arm moving in a horizontal plane. The memory-based approach performed surprisingly well in these simple evaluations. The simulated arm was able to follow a desired trajectory after only a few practice movements. The cross validation identified irrelevant input variables and terms in the local model in the arm dynamics problem.

We also compared the performance of memory-based motor learning with that of other approaches such as CMAC and a three layer feedforward neural network. Our memory-based approach generalizes more effectively than CMAC, and has much less interference between old and new data than the feedforward neural network model structure.

2 Roles of Function Approximation in Learning Control

Memory-Based modeling is one approach to approximating functions. There are many others. In order to understand why memory-based modeling might be useful I will outline some potential roles of function approximation in learning control. In this outline I will talk about general "learning". Learning problems that have been studied extensively include refining kinematic models, refining dynamic models, learning a feedforward command to follow a particular trajectory more accurately, and learning models of particular tasks. Function approximation can play a role in all of these forms of learning. I will use the following notation that can be applied to all of these types of learning problems. A command $\mathbf{c}$ is given to a plant in state $\mathbf{x}$ which results in output $\mathbf{p}$:

$$\mathbf{p} = \mathbf{f}(\mathbf{x}, \mathbf{c}) \tag{1}$$

A unique inverse for the true plant may exist:

$$\mathbf{c} = \mathbf{f}^{-1}(\mathbf{x}, \mathbf{p}) \tag{2}$$

2.1 Forming an inverse model of the plant

One application of learning control is to directly learn an inverse model of the plant

$$\mathbf{c} = \widehat{\mathbf{f}}^{-1}(\mathbf{x}, \mathbf{p}) \tag{3}$$

from samples $(\mathbf{x}_i, \mathbf{c}_i, \mathbf{p}_i)$. This approach assumes a unique inverse exists. Once the inverse model is formed, a command to achieve a particular goal can be found with a single query. It is reasonable to expect that there will be as many data points to learn from as there will be queries when using an inverse model in on-line learning. Each query leads to an actual command applied and a new data point to be learned. This suggests that function approximation methods should be efficient to train, and that expensive iterative searches for parameters in nonlinear global models may not be appropriate in this context, although the global model may be inexpensive to evaluate for any particular query. Another important issue with this approach is generating sufficiently rich training data so that the inverse model makes accurate predictions and the learning process converges. There are a wide range of teaching strategies to generate the training data, including random experimentation and using a crude feedback controller for initial control.

2.2 Forming a forward model of the plant

In cases where a unique inverse for the plant does not exist a forward model can be used in an iterative search for the appropriate command to apply to achieve a particular goal. For example, the following three step process can be used to find a command that achieves a goal $\mathbf{p}_d$:

1. Guess an initial command $\mathbf{c}_0$. An approximate inverse model could be used:
$$\mathbf{c}_0 = \widehat{\mathbf{f}}^{-1}(\mathbf{x}, \mathbf{p}_d) \tag{4}$$

2. Use the forward model to predict the performance that will result from applying a command (mental simulation):
$$\mathbf{p}_i = \widehat{\mathbf{f}}(\mathbf{x}, \mathbf{c}_i) \tag{5}$$

3. Use derivatives of the forward model (the Jacobian $\mathbf{J} = \partial \mathbf{p} / \partial \mathbf{c}$) to refine the command. Gradient descent is one approach (Jordan 1988):
$$\mathbf{c}_{i+1} = \mathbf{c}_i - \delta \mathbf{c}_i \tag{6}$$
$$\delta \mathbf{c}_i = \widehat{\mathbf{J}}^T(\mathbf{p}_i - \mathbf{p}_d) \tag{7}$$
More aggressive nonlinear equation solving (root finding) techniques can be used, such as variants of the Newton Raphson method, which involve using pseudoinverse techniques to solve
$$\widehat{\mathbf{J}} \delta \mathbf{c}_i = (\mathbf{p}_i - \mathbf{p}_d) \tag{8}$$
for $\delta \mathbf{c}_i$. If $\widehat{\mathbf{J}}$ is almost singular, these corrections may be invalid because they are too large. Various schemes can be used to limit the size of the command corrections.

Any number of iterations of steps 2 and 3 can be performed, until the predicted command converges or the forward model is queried outside the range of the training data. At this point the command should be applied, new training data obtained, and the above process repeated. In this approach the forward model is likely to be evaluated many times relative to the number of training points, which suggests that a function approximation method that is cheap to evaluate and to find derivatives should be used, even if it is expensive to train.

The approach of using a forward model and mental simulation may give this method a wider range of convergence than methods that form inverse models. We are currently exploring this issue. In cases where more than one command can achieve the desired output, the solution chosen by these methods depends on the starting point of the learning process. Additional constraints, or a criteria to be optimized can be added to force a unique choice of command independent of the starting point of the search. Generating initial training data, and active exploration during the learning process are important issues that have not been carefully examined. Random or deliberate experiments, based on the observed performance improvement rate, could be used.

2.3 Function optimization

Often a goal in learning is to optimize a particular criteria, rather than achieve a particular output. Function approximation techniques can be used to represent the cost function directly, and to speed the search for extreme points (Devroye 1978, Schagen 1984, Powell 1987). A linear local model can be used to estimate the first derivatives (gradient) and a quadratic local model can be used to estimate the second derivatives (Hessian) of the function at the current point in the optimization procedure. These estimates can be used in gradient descent, or in a Gauss Newton method which uses estimates of second derivatives. This optimization process may be constrained in that some particular constraints on the output must be satisfied as well as a criteria optimized, a combination of root finding and optimization.

2.4 Optimization over time: dynamic programming

In situations where commands must be optimized over time, dynamic programming provides a framework to perform the optimization (Jacobson and Mayne 1970). Let us consider a discrete time problem although the continuous time problem is closely related. A forward model of the plant

$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is learned as well as the single step cost function $L(\mathbf{x}_k, \mathbf{u}_k)$. Assume an estimate of the cost to go $V_{k+1}(\mathbf{x}_{k+1})$ has been constructed. Bellman's principle of optimality gives us a rule for finding the cost to go V_k and the optimal control $\mathbf{u}_k$ at time step k, given an initial state $\mathbf{x}_k$:

$$V_k(\mathbf{x}_k) = \min_{\mathbf{u}_k}[L_k(\mathbf{x}_k, \mathbf{u}_k) + V_{k+1}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))] \tag{9}$$

For smooth problems this minimization can be performed by continuous optimization techniques. As described in the previous section, local linear forward models of the plant and local quadratic models of the one step cost and cost to go can be used to optimize the control (Jacobson and Mayne 1970).

Thus, goals for function approximation in learning control go beyond being able to represent the training set and generalize appropriately. Other desired criteria include:

- Fast training based on a continuous stream of incoming data, rather than a fixed training set.
- Ability to find first (and potentially second) derivatives of the learned function.
- Ability to tell where in the input space the function is accurately approximated. This is typically based on the local density of samples, and an estimate of the local variance of the outputs. This ability is used in iterative use of the model to determine when to terminate search and collect more data.
- Ability to find good initial guesses for search or optimization procedures. Memory-based approaches typically scan the entire data set. If the data set is represented as a set of parameters, this is more difficult.

3 Related Work

Two forms of modeling previous experience are those approaches that represent the experiences directly, as in this study, and those that represent the experiences using parameters or weights, as in many table-based schemes, perceptrons and multilayer connectionist or model neural networks. For a more extensive review of related work see (Atkeson 1989).

3.1 Direct Storage Of Experience

Memory-based modeling has a long history. Approaches which represent previous experiences directly and use a similar experience or similar experiences to form a local model are often referred to as nearest neighbor or k-nearest neighbor approaches. Local models (often polynomials) have been used for many years to smooth time series (Sheppard 1912, Sherriff 1920, Whittaker and Robinson 1924, Macauley 1931) and interpolate and extrapolate from limited data. Barnhill (1977) and Sabin (1980) survey the use of nearest neighbor interpolators to fit surfaces to arbitrarily spaced points. Eubank (1988) surveys the use of nearest neighbor estimators 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. Farmer and Sidorowich (1988a, 1988b) survey the use of nearest neighbor and local model approaches in modeling chaotic dynamic systems.

An early use of direct storage of experience was in pattern recognition. Fix and Hodges (1951, 1952) suggested that a new pattern could be classified by searching for similar patterns among a set of stored patterns, and using the categories of the similar patterns to classify the new pattern. Steinbuch and Taylor proposed a neural network implementation of the direct storage of experience and nearest-neighbor search process for pattern recognition (Steinbuch 1961, Taylor 1959), and pointed out that this approach could be used for control (Steinbuch and Piske 1963). Stanfill and Waltz (1986) proposed using directly stored experience to learn pronunciation, using a Connection Machine and parallel search to find relevant experience. They have also applied their approach to medical diagnosis (Waltz 1987) and protein structure prediction.

Nearest neighbor approaches have also been used in nonparametric regression and fitting surfaces to data. Often, a group of similar experiences, or nearest neighbors, is used to form a local model, and then that model is used to predict the desired value for a new point. Local models are formed for each new access to the memory. Watson (1964), Royall (1966), Crain and Bhattacharyya (1967), Cover (1968), and Shepard (1968) proposed using a weighted average of a set of nearest neighbors. Gordon and Wixom (1978) and Barnhill, Dube, and Little (1983) analyze such weighted average schemes. Crain and Bhattacharyya (1967), Falconer (1971), and McLain (1974) suggested using a weighted regression 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. McIntyre, Pollard, and Smith (1968), Pelto, Elkins, and Boyd (1968), Legg and Brent (1969), Palmer (1969), Walters (1969), Lodwick and Whittle (1970), Stone (1975), and Franke and Nielson (1980) suggested fitting a polynomial surface to a set of nearest neighbors, also using distance weighted regression. Stone scaled the values in each dimension when the experiences where stored. The standard deviations of each dimension of previous experiences were used as the scaling factors, so that the range of values in each dimension were approximately equal. This affects the distance metric used to measure closeness of points. Cleveland (1979) proposed using robust regression procedures to eliminate outlying or erroneous points in the regression process. A program implementing a refined version of this approach (LOESS) is available by sending electronic mail containing the single line, *send dloess from a*, to the address netlib@research.att.com (Grosse 1989). Cleveland, Devlin and Grosse (1988) analyze the statistical properties of the LOESS algorithm and Cleveland and Devlin (1988) show examples of its use. Stone (1977, 1982), Devroye (1981), Lancaster (1979), Lancaster and Šalkauskas (1981), Cheng (1984), Li (1984), Farwig (1987), and Müller (1987) provide analyses of nearest neighbor approaches. Franke (1982) compares the performance of nearest neighbor approaches with other methods for fitting surfaces to data.

4 How Does Locally Weighted Regression Work?

There are several stages in the version of locally weighted regression developed in this research.

1. Initially, experiences are simply stored in memory.

2. To answer any particular query a weighted linear regression is performed. The weights in the regression depend on parameters used to calculate a distance metric and the weighting function, and stabilize the solution to the regression. I will refer to these parameters as "fit parameters".

3. The fit parameters can be optimized using cross validation.

4.1 Storing The Experiences

The choice of the method of 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.

K-d trees (Friedman, Bentley, and Finkel 1977) are a tree data structure used to speed up the search for a best match, and are an appropriate technique for finding nearest neighbors on a serial computer. However, the number of data points that must be examined to find the best match grows exponentially with the number of dimensions used in the distance metric, in the worst case. The actual performance of k-d trees depends on the distribution of the data, the less randomly distributed the better. Farmer and Sidorowich (1987, 1988a, 1988b) used k-d trees and a variety of generalization functions to model chaotic dynamic systems. Cleveland and Devlin (1988) and Cleveland, Devlin, and Grosse (1988) used k-d trees in the LOESS program. Moore (1990) used k-d trees to model a robot manipulator.

On a parallel computer, such as the Connection Machine (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. This approach is similar to many Connection Machine algorithms that find a best match or set of nearest neighbors (Waltz 1987). The search time depends linearly on the number of dimensions in the distance metric, and the distance metric can be easily changed or made to depend on the current query point.

We have implemented both of these forms of storing and accessing the experiences, using standard workstations for the implementation of k-d trees, and using the Connection Machine to do parallel exhaustive search.

4.2 Performing The Regression

As an example of modeling a function using locally weighted regression, we will consider a problem from motor control and robotics, two-joint arm inverse dynamics. We will predict torques at the shoulder, τ_1, and elbow, τ_2, on the basis of joint positions, θ_1 and θ_2, joint velocities, $\dot{\theta}_1$ and $\dot{\theta}_2$, and joint accelerations, $\ddot{\theta}_1$ and $\ddot{\theta}_2$. We use this example because we already know the idealized function, and will be able to assess how well the locally weighted regression procedure is doing and interpret the parameters used to improve the fit. In an actual application a structured model (An, Atkeson, and Hollerbach 1988, for example) would be used to fit the dynamics data, and the locally weighted regression would be used to fit the errors (residuals) of the parametric model.

We have a query point $(\theta_1^*, \theta_2^*, \dot{\theta}_1^*, \dot{\theta}_2^*, \ddot{\theta}_1^*, \ddot{\theta}_2^*)$ for which we want to predict the shoulder and elbow torques. We will first show how an unweighted regression can be used to form a global model. Then we will show how a weighted regression can be used to form a local model appropriate to answer this particular query. For the purposes of this example we will assume a quadratic model is used in the regression. In this dynamics example there are 28 terms in the quadratic model:

$$\begin{array}{ccccccc}
1 & \theta_1, & \theta_2, & \dot{\theta}_1, & \dot{\theta}_2, & \ddot{\theta}_1, & \ddot{\theta}_2, \\
 & \theta_1 * \theta_1, & \theta_1 * \theta_2, & \theta_1 * \dot{\theta}_1, & \theta_1 * \dot{\theta}_2, & \theta_1 * \ddot{\theta}_1, & \theta_1 * \ddot{\theta}_2, \\
 & & \theta_2 * \theta_2, & \theta_2 * \dot{\theta}_1, & \theta_2 * \dot{\theta}_2, & \theta_2 * \ddot{\theta}_1, & \theta_2 * \ddot{\theta}_2, \\
 & & & \dot{\theta}_1 * \dot{\theta}_1, & \dot{\theta}_1 * \dot{\theta}_2, & \dot{\theta}_1 * \ddot{\theta}_1, & \dot{\theta}_1 * \ddot{\theta}_2, \\
 & & & & \dot{\theta}_2 * \dot{\theta}_2, & \dot{\theta}_2 * \ddot{\theta}_1, & \dot{\theta}_2 * \ddot{\theta}_2, \\
 & & & & & \ddot{\theta}_1 * \ddot{\theta}_1, & \ddot{\theta}_1 * \ddot{\theta}_2, \\
 & & & & & & \ddot{\theta}_2 * \ddot{\theta}_2
\end{array}$$

where 1 represents the constant term in the model.

Let us assume we have 1000 samples of the two joint arm dynamics function. The equation to be solved is

$$\mathbf{X}\beta = \mathbf{y} \tag{10}$$

where $\mathbf{X}$ is a 1000 by 28 data matrix, in which each row has the 28 terms of the quadratic model corresponding to a point (sample of the function), and each column corresponds to a particular term in the quadratic model. β is the vector of 28 estimated parameters of the quadratic model, and $\mathbf{y}$ is the vector of 1000 torques from the 1000 points included in the regression.

An unweighted regression finds the solution to the normal equations:

$$\left(\mathbf{X}^T\mathbf{X}\right)\beta = \mathbf{X}^T\mathbf{y} \tag{11}$$

The estimated parameters are used, with the query point, to predict the torques for the query point.

However, we assume the global quadratic model is not the correct model structure for predicting the torques. These structural modeling errors imply that different sets of parameters are estimated by the regression, given different data sets. The data set can be tailored to the query point by emphasizing nearby points in the regression. The origin of the input data is first shifted by subtracting the query point from each data point. Then each data point is given a weight.

Unweighted regression gives distant points equal influence with nearby points on the ultimate answer to the query. To weight similar points more, a weighted regression is used. Each row of $\mathbf{X}$ and $\mathbf{y}$ is multiplied by a weight:

$$w_i = (d_i^2)^{-p} \tag{12}$$

where w_i is the weight for row i (corresponding to the ith point), d_i is the Euclidean distance of point i to the query point, and p is a parameter that determines how local the regression will be (the rate of dropoff of the weights with distance). d_i^2 is calculated for each point in the following way (in this example):

$$\begin{aligned} d^2 &= m_1^2(\theta_1 - \theta_1^*)^2 + m_2^2(\theta_2 - \theta_2^*)^2 + m_3^2(\dot{\theta}_1 - \dot{\theta}_1^*)^2 \\ &\quad + m_4^2(\dot{\theta}_2 - \dot{\theta}_2^*)^2 + m_5^2(\ddot{\theta}_1 - \ddot{\theta}_1^*)^2 + m_6^2(\ddot{\theta}_2 - \ddot{\theta}_2^*)^2 \end{aligned} \tag{13}$$

The superscript * indicates the query point, and the m_i are the components of the distance metric.

A potential problem is that the data points may be distributed in such a way as to make the regression matrix $\mathbf{X}$ nearly singular. A technique known as ridge regression (Draper and Smith 1981) is used to prevent problems due to a singular data matrix. The following equation, with $\mathbf{X}$ and $\mathbf{y}$ already weighted, is solved for β:

$$\left(\mathbf{X}^T\mathbf{X} + \Lambda\right)\beta = \mathbf{X}^T\mathbf{y} \tag{14}$$

where Λ is a diagonal matrix with small positive diagonal elements λ_i^2. This is equivalent to adding i extra rows to $\mathbf{X}$, each having a single non-zero element, λ_i, in the ith column. Adding additional rows can be viewed as adding "fake" data, which, in the absence of sufficient real data, biases the parameter estimates to zero (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).

4.3 Optimizing The Fit Parameters

For the example problem, we have introduced 34 free parameters into the local regression process: the weighting function dropoff parameter p, the 6 elements of the distance metric m_i, and the 27 variable diagonal elements of Λ (the ridge regression parameters λ_i). The element of Λ corresponding to the constant term, λ_1, is held fixed.

A cross validation approach is used to choose values for these fit parameters. The sum of the squared cross validation error is minimized using a nonlinear parameter estimation procedure (MINPACK (More, Garbow, and Hillstrom 1980) or NL2SOL (Dennis, Gay, and Welsch 1981), for example). We can take the derivative of the cross validation error with respect to the parameters to be estimated, which greatly speeds up the search process (See Appendix).

The cross validation to optimize the fit parameters may be done globally, using all the experiences in the memory to produce one set of fit parameters. Different fit parameters can be used for different outputs. The cross validation may also be done locally, either with each query, or separately for different regions of the input space, producing different sets of fit parameters specialized for particular queries. We have only experimented with global cross validation so far.

We can use the optimized distance metric to find which input variables are irrelevant to the function being represented. In the horizontal two-joint arm inverse dynamics problem, m_1, the weight on the distances in the θ_1 direction typically drops to zero, Indicating that the input variable θ_1 is irrelevant to predicting τ_1 and τ_2. This is actually the case, as θ_1 does not appear in the true dynamics equations for an arm operating in a horizontal plane.

We can also interpret the ridge regression parameters, λ_i. Since the arm dynamics are linear in acceleration, the terms in the local model that are quadratic in acceleration ($\ddot{\theta}_1^2$, $\ddot{\theta}_1 * \ddot{\theta}_2$, $\ddot{\theta}_2^2$) are not relevant to predicting torques. Similarly the products of velocity and acceleration ($\dot{\theta}_1 * \ddot{\theta}_1$, $\dot{\theta}_1 * \ddot{\theta}_2$, $\dot{\theta}_2 * \ddot{\theta}_1$, $\dot{\theta}_2 * \ddot{\theta}_2$) are also not relevant to the dynamics. The ridge regression parameter for each of these terms becomes very large in the parameter optimization. The effect of this is to force the estimated parameter β_i for these terms to be zero and the terms to have no effect on the regression.

We have also explored stepwise regression procedures to determine which terms of the local model are useful (Draper and Smith 1981) with similar results.

5 Performance Comparisons

Two methods, CMAC (Albus 1975ab) and sigmoidal feedforward neural networks, were compared to the approach explored in this paper. The parameters for the CMAC approach were taken from Miller, Glanz, and Kraft (1987) who used the CMAC to model arm dynamics. The architecture for the sigmoidal feedforward neural network was taken from Goldberg and Pearlmutter (1988, section 6) who also modeled arm dynamics.

The ability of each of these methods to predict the torques of the simulated two joint arm at 1000 random points was compared. Figure 1 plots the normalized RMS prediction error. The points were sampled uniformly using ranges comparable to those used in (Miller et al 1987). Initially, each method was trained on a training set of 1000 random samples of the two joint arm dynamics function, 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 (points 1, 3, and 5). Each method was then trained on 10 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 (points 2, 4, and 6).

The data indicate that the locally weighted regression approach (filled in circles) and the sigmoidal feedforward network approach (asterisks) both generalize well on this problem (points 3 and 5 have low error). The CMAC (diamonds) did not generalize well on this problem (point 1 has a large error), although it represented the original training set with a normalized RMS error of 0.000001. A variety of CMAC resolutions were explored, ranging from a basic CMAC cell size covering the entire range of data to a cell size covering

Figure 1: Performance of various methods on two joint arm dynamics.

a fifth of the data range in each dimension. A cell size covering one half the data ranges in each dimension generalized best (the data shown here).

After training on a different training set (the attempts to make a particular desired movement), the sigmoidal feedforward neural network lost its memory of the full dynamics (point 4), 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 other methods explored did not show this interference effect (points 2 and 6).

6 Conclusions

It will be necessary to try out this memory-based approach to modeling on real data from several problems in order to more fully assess the potential of the approach. We are currently attempting to use these methods for both learning control (see An, Atkeson, and Hollerbach 1988 for further references on learning control) and task level learning (Aboaf, Drucker, and Atkeson 1989). Several points can be made at this early stage of the research.

This approach to modeling is appropriate for situations where new data is being collected continuously. Storing new data is cheap: the new data point is simply stored in a memory. However, making a prediction using this approach is much more expensive than just evaluating a global model at a point, as a new local model is fit for each query. If there is a fixed training set followed by many queries, other approaches will be preferable. If there is a continuously growing training set intermixed with queries, a memory-based approach seems worth exploring. The local modeling approach, because of its retention of all the data in memory, reduces interference between old and new data, and allows the range of generalization to depend on the density of the samples.

The cross validation approach to optimizing the fit parameters reduces the number of arbitrary choices that need to be made before the training data is collected. However, like other modeling approaches, the choice of representation of the data (number and selection of dimensions to be measured, etc.) play a large role in determining the success of the approach.

One concern with memory-based approaches is that more data will eventually be collected than will fit in the memory. In many problems and applications this is not an issue. However, algorithms for forgetting data and algorithms that determine whether or not new data is worth storing remain to be explored.

7 Appendix: Optimizing The Fit Parameters

The cross validation error for point i is the difference of y_i and the prediction of y_i generated by removing point i from the memory and performing a locally weighted regression using the current set of fit parameters with point i as the query point. The derivative of the cross validation error for point i is given by the derivative of the prediction for point i with respect to the fit parameters, since y_i itself is a known fixed quantity.

The dependence on the distance metric occurs indirectly through the weights:

$$\frac{\partial \hat{y}_i}{\partial m_k} = \sum_j \left(\frac{\partial \hat{y}_i}{\partial w_j} \frac{\partial w_j}{\partial m_k} \right) \tag{15}$$

$\hat{y}_i$ is the prediction for the ith cross validated point, m_k is the weight for the kth dimension in the distance metric, and w_j is the weight for the jth point or row in $\mathbf{X}$.

$$\frac{\partial \hat{y}_i}{\partial w_j} = \frac{2e_j\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_j^T\right]_l}{w_j} \tag{16}$$

where $\mathbf{X}^T\mathbf{X}$ is computed from the weighted regression matrix and includes the ridge regression parameters, $\mathbf{x}_j$ is the jth row of $\mathbf{X}$, and e_j is the jth residual for the current weighted regression (Belsley, Kuh, and Welsch 1980). $\mathbf{X}$, $\mathbf{x}_j$, and e_j are all evaluated with the current set of fit parameters. The l index indicates that the element of the vector in brackets corresponding to the constant term is selected (since the origin is moved to the query point, $\hat{y}_i$ is the constant term estimated by the ith regression).

$$\frac{\partial w_j}{\partial m_k} = -2w_j m_k p \frac{\delta_{jk}^2}{d_j^2} \tag{17}$$

where p is the power term, w_j is the weight for the jth point, m_k is the metric for the kth dimension, δ_{jk} is the difference between the jth point and the desired point in the kth dimension, and d_j^2 is the squared total distance between the jth point and the desired point.

The dependence on the power parameter in the weighting function also occurs indirectly through the weights:

$$\frac{\partial \hat{y}_i}{\partial p} = \sum_j \left(\frac{\partial \hat{y}_i}{\partial w_j}\frac{\partial w_j}{\partial p}\right) \tag{18}$$

$$\frac{\partial w_j}{\partial p} = -w_j \ln(d_j^2) \tag{19}$$

The ridge regression parameters can be treated as weights directly:

$$\frac{\partial \hat{y}_i}{\partial \lambda_k} = \frac{2e_k\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\right]_l}{\lambda_k} \tag{20}$$

$\mathbf{x}_k$ is a row with a single non-zero element λ_k in the kth position (the kth fake data point), and e_k is the residual for that row of $\mathbf{X}$.

8 Acknowledgments

Support was provided under Office of Naval Research contract N00014-88-K-0321 and under Air Force Office of Scientific Research grant AFOSR-89-0500. Support for CGA was provided by a National Science Foundation Engineering Initiation Award and Presidential Young Investigator Award, an Alfred P. Sloan Research Fellowship, the W. M. Keck Foundation Assistant Professorship in Biomedical Engineering, and a Whitaker Health Sciences Fund MIT Faculty Research Grant.

9 References

Aboaf, E.W., S.M. Drucker, and C.G. Atkeson (1989) "Task-Level Robot Learning: Juggling a Tennis Ball More Accurately", *Proceedings, IEEE International Conference on Robotics and Automation,* May 14-19, 1989, Scottsdale, Arizona.

Albus, J.S. (1975a) "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)." *ASME J. Dynamic Systems, Meas., Control,* 97(September):220-227.

Albus, J.S. (1975b) "Data Storage in the Cerebellar Model Articulation Controller (CMAC)." *ASME J. Dynamic Systems, Meas., Control,* 97(September):228-233.

An, C.H., C.G. Atkeson, and J.M. Hollerbach (1988) *Model-Based Control of a Robot Manipulator* MIT Press, Cambridge, MA.

Atkeson, C.G. (1989) "Learning Arm Kinematics And Dynamics", *Annual Review of Neuroscience,* 12:157-183.

Barnhill, R.E. (1977) "Representation And Approximation of Surfaces", in *Mathematical Software III,* J.R. Rice, ed. Academic Press, New York, pp. 69-120.

Barnhill, R.E., R.P. Dube, and F.F. Little (1983) "Properties of Shepard's Surfaces", Rocky Mountain Journal of Mathematics 13(2): 365-382.

Belsley, D. A., E. Kuh, and R. E. Welsch (1980) *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity,* John Wiley & Sons, New York.

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. and S.J. Devlin (1988) "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting", *Journal of the American Statistical Association* 83, 596-610.
Cleveland, W.S., S.J. Devlin and E. Grosse (1988) "Regression by Local Fitting: Methods, Properties, and Computational Algorithms", *Journal of Econometrics* 37, 87-114.
Cover, T.M. (1968) "Estimation by the Nearest Neighbor Rule", *IEEE Transactions on Information Theory,* IT-14, 50-55.
Crain, I.K. and B.K. Bhattacharyya (1967) "Treatment of nonequispaced two dimensional data with a digital computer", *Geoexploration,* 5:173-194.
Dennis, J.E., D.M. Gay, and R.E. Welsch (1981) "An Adaptive Nonlinear Least-Squares Algorithm", *ACM Transactions on Mathematical Software,* 7(3) 1981.
Devroye, L.P. (1978) "The Uniform Convergence of Nearest Neighbor Regression Function Estimators and Their Application in Optimization", *IEEE Transactions on Information Theory,* IT-24, 142-151.
Devroye, L.P. (1981) "On the Almost Everywhere Convergence of Nonparametric Regression Function Estimates", The Annals of Statistics, 9(6):1310-1319.
Draper, N.R. and H. Smith (1981) *Applied Regression Analysis, 2nd ed.,* John Wiley, New York.
Eubank, R.L. (1988) *Spline Smoothing and Nonparametric Regression,* Marcel Dekker, New York, pp. 384-387.
Falconer, K.J. (1971) "A general purpose algorithm for contouring over scattered data points", Nat. Phys. Lab. Report NAC 6.
Farmer, J.D., and J.J. Sidorowich (1987) "Predicting Chaotic Time Series," *Physical Review Letters,* 59(8): 845-848.
Farmer, J.D., and J.J. Sidorowich (1988a) "Exploiting Chaos to Predict the Future and Reduce Noise." Technical Report LA-UR-88-901, Los Alamos National Laboratory, Los Alamos, New Mexico.

Farmer, J.D., and J.J. Sidorowich (1988b) "Predicting Chaotic Dynamics." in *Dynamic Patterns in Complex Systems,* J.A.S. Kelso, A.J. Mandell, and M.F. Schlesinger, (eds.) World Scientific, New Jersey, pp. 265-292.
Farwig, R. (1987) "Multivariate Interpolation of Scattered Data by Moving Least Squares Methods", in J.C. Mason and M.G. Cox (eds), *Algorithms for Approximation,* Clarendon Press, Oxford, pp. 193-211.
Fix, E. and J.L. Hodges, Jr. (1951) "Discriminatory analysis, Nonparametric regression: consistency properties", Project 21-49-004, Report No. 4. USAF School of Aviation Medicine Randolph Field, Texas. Contract AF-41-(128)-31, Feb. 1951
Fix, E. and J.L. Hodges, Jr. (1952) "Discriminatory analysis: small sample performance", Project 21-49-004, Rep. 11 USAF School of Aviation Medicine Randolph Field, Texas. Aug. 1952.
Franke, R. (1982) "Scattered Data Interpolation: Tests of Some Methods." *Mathematics of Computation,* 38(157):181-200.
Franke, R. and G. Nielson (1980) "Smooth Interpolation of Large Sets of Scattered Data", *International Journal Numerical Methods Engineering,* 15:1691-1704.
Friedman, J.H., J.L. Bentley, and R.A. Finkel (1977) "An Algorithm for Finding Best Matches in Logarithmic Expected Time." ACM Trans. on Mathematical Software, 3(3), Sept., pp. 209-226.
Goldberg, K.Y. and B. Pearlmutter (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, August 1988.
Gordon, W.J. and J.A. Wixom (1978) "Shepard's Method of Metric Interpolation to Bivariate and Multivariate Interpolation", *Mathematics of Computation,* 32(141):253-264.
Grosse, E. (1989) "LOESS: Multivariate Smoothing by Moving Least Squares" in C.K. Chui, L.L. Schumaker, and J.D. Ward (eds.), *Approximation Theory VI* pp. 1-4. Academic Press, Boston.
Hillis, D. (1985) *The Connection Machine.* MIT Press, Cambridge, Mass., 1985.
Kazmierczak, H. and K. Steinbuch (1963) "Adaptive Systems in Pattern Recognition," *IEEE Trans. on Electronic Computers,* EC-12, December, pp. 822-835.

Jacobson, D.H. and D.Q. Mayne (1970) *Differential Dynamic Programming* American Elsevier Publishing Company, New York.
Jordan, M.I., and Rosenbaum, D.A. (1988) "Action" (Tech. Rep. 88-26). Computer and Information Science, University of Massachusetts, Amherst.
Lancaster, P. (1979) "Moving Weighted Least-Squares Methods", in B.N. Sahney (ed.), *Polynomial and Spline Approximation,* 103-120. D. Reidel Publishing, Boston.
Lancaster, P. and K. Šalkauskas (1981) "Surfaces Generated by Moving Least Squares Methods", *Mathematics of Computation,* 37(155):141-158.
Lancaster, P. and K. Šalkauskas (1986) *Curve And Surface Fitting* Academic Press, New York.
Legg M.P.C. and R.P. Brent (1969) "Automatic Contouring," *Proc. 4th Australian Computer Conference,* 467-468.
Li, K.C. (1984) "Consistency for Cross-Validated Nearest Neighbor Estimates in Nonparametric Regression", *The Annals of Statistics,* 12:230-240.
Lodwick, G.D., and J. Whittle (1970) "A technique for automatic contouring field survey data", *Australian Computer Journal,* 2:104-109.
Macauley, F.R. (1931) *The Smoothing of Time Series,* National Bureau of Economic Research, New York.
McIntyre, D.B., D.D. Pollard, and Roger Smith (1968) "Computer Programs For Automatic Contouring", *Kansas Geological Survey Computer Contributions 23*, University of Kansas, Lawrence, Kansas.
McLain, D.H. (1974) "Drawing Contours From Arbitrary Data Points", *The Computer Journal,* 17(4):318-324.
Miller, W.T., F.H. Glanz, and L.G. Kraft (1987) "Application of a general learning algorithm to the control of robotic manipulators", *International Journal of Robotics Research,* 6:84-98.
Moore, A.W. (1990) "Acquisition of Dynamic Control Knowledge for a Robotic Manipulator." in *Machine Learning: Proceedings of the Seventh International Conference (1990),* edited by B.W. Porter, and R.J. Mooney, pp. 244-252, Morgan Kaufmann, San Mateo, CA.
More, J.J., B.S. Garbow, and K.E. Hillstrom (1980) "User Guide for MINPACK-1", 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.
Palmer, J.A.B. (1969) "Automated mapping," *Proc. 4th Australian Computer Conference,* 463-466.
Pelto, C.R., T.A. Elkins, and H.A. Boyd (1968) "Automatic contouring of irregularly spaced data", *Geophysics,* 33:424-430.
M. J. D. Powell (1987) "Radial basis functions for multivariable interpolation: A review". In J. C. Mason and M. G. Cox (ed.), *Algorithms for Approximation*, 143-167. Clarendon Press, Oxford.
Royall, R.M. (1966) "A class of nonparametric estimators of a smooth regression function", Ph. D. dissertation and Tech Report No. 14, Public Health Service Grant USPHS-5T1 GM 25-09, Department of Statistics, Stanford University.
Sabin, M.A. (1980) "Contouring – A Review of Methods for Scattered Data", in *Mathematical Methods in Computer Graphics and Design,* K.W. Brodlie (ed.), Academic Press, New York. pp. 63-86.
Schagen, I.P. (1984) "Sequential Exploration of Unknown Multi-dimensional Functions as an Aid to Optimization", *IMA Journal of Numerical Analysis* 4:337-347.
Seber, G.A.F. (1977) *Linear Regression Analysis,* John Wiley, New York.
Shepard, D. (1968) "A two-dimensional function for irregularly spaced data", *Proceedings of 23rd ACM National Conference,* 517-524.
Sheppard, W. F. (1912) "Reduction of Errors by Means of Negligible Differences," *Proceedings of the Fifth International Congress of Mathematicians,* Vol II, pp. 348-384, E. W. Hobson and A. E. H. Love (eds), Cambridge University Press.
Sherriff, C. W. M. (1920) "On a Class of Graduation Formulae," *Proceedings of the Royal Society of Edinburgh,* XL: 112-128.
Stanfill, C., and D. Waltz (1986) "Toward Memory-Based Reasoning." *Communications of the ACM*, vol. 29 no. 12, December, pp. 1213-1228.
Steinbuch, K (1961) "Die lernmatrix," *Kybernetik,* 1:36-45.
Steinbuch, K. and U. A. W. Piske (1963) "Learning Matrices and Their Applications," *IEEE Trans. on Electronic Computers,* EC-12, December, pp. 846-862.
Steinbuch, K. and B. Widrow (1965) "A Critical Comparison of Two Kinds of Adaptive Classification Networks," *IEEE Trans. on Electronic*

Computers, EC-14, October, pp. 737-740.
Stone, C.J. (1975) "Nearest Neighbor Estimators of a Nonlinear Regression Function", *Proc. of Computer Science and Statistics: 8th Annual Symposium on the Interface* pp. 413-418.
Stone, C.J. (1977) "Consistent Nonparametric Regression", *The Annals of Statistics* 5, 595-645.
Stone, C.J. (1982) "Optimal Global Rates of Convergence for Nonparametric Regression", *The Annals of Statistics,* 10(4):1040-1053.
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.
Taylor, W.K. (1964) "Cortico-thalamic organization and memory", *Proc. Royal Society B,* 159:466-478.
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, January.
Watson, G.S. (1964) "Smooth Regression Analysis", *Sankhyā: The Indian Journal of Statistics, Series A,* 26:359-372.
Whittaker, E., and G. Robinson (1924) *The Calculus of Observations,* Blackie & Son, London.