Scalable Techniques from Nonparametric Statistics for Real Time Robot Learning

Stefan Schaal [‡⊕]
sschaal@usc.edu
www-slab.usc.edu/sschaal

Christopher G. Atkeson [*#]
cga@cc.gatech.edu
www.cc.gatech.edu/fac/Chris.Atkeson

Sethu Vijayakumar [‡⊕]
sethu@usc.edu
www-slab.usc.edu/sethu

[‡]Computer Science and Neuroscience, HNB-103, University of Southern California, Los Angeles, CA 90089-2520
[*]College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280
[⊕]Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan
[#]ATR Human Information Processing Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan

***Abstract**: Locally weighted learning (LWL) is a class of techniques from nonparametric statistics that provides useful representations and training algorithms for learning about complex phenomena during autonomous adaptive control of robotic systems. This paper introduces several LWL algorithms that have been tested successfully in real-time learning of complex robot tasks. We discuss two major classes of LWL, memory-based LWL and purely incremental LWL that does not need to remember any data explicitly. In contrast to the traditional belief that LWL methods cannot work well in high-dimensional spaces, we provide new algorithms that have been tested on up to 90 dimensional learning problems. The applicability of our LWL algorithms is demonstrated in various robot learning examples, including the learning of devil-sticking, pole-balancing by a humanoid robot arm, and inverse-dynamics learning for a seven and a 30 degree-of-freedom robot. In all these examples, the application of our statistical neural networks techniques allowed either faster or more accurate acquisition of motor control than classical control engineering.*

***Keywords**: Nonparametric Regression • Locally Weighted Learning • Motor Control • Internal Models • Incremental Learning*

1 Introduction

The necessity for self-improvement in control systems is becoming more apparent as fields such as robotics, factory automation, and autonomous vehicles become impeded by the complexity of inventing and programming satisfactory control laws. Learned models of complex tasks can aid the design of appropriate control laws for these tasks, which often involve decisions based on streams of information from sensors and actuators, and where data is relatively plentiful. Learning also seems to be the only viable research approach toward the generation of flexible autonomous robots that can perform multiple tasks (Schaal, 1999), with the hope of creating an autonomous humanoid robot at some point and to approach human abilities in sensorimotor control.

When approaching a learning problem, many alternative learning methods are available from the neural network, statistical, and machine learning literature. The current focus in learning research lies on increasingly more sophisticated algorithms for the off-line analysis of finite data sets, without severe constraints on the computational complexity of the algorithms. Examples of such algorithms include the revival of Bayesian inference (Bishop, 1995, Williams and Rasmussen, 1996) and the new algorithms developed in the framework of structural risk minimization (Vapnik, 1982, Cortes and Vapnik, 1995). Mostly, these methods target problems in classification and diagnostics, although several extensions to regression problems exist (e.g., Vapnik, Golowich, and Smola, 1996).

In motor learning, however, special constraints need to be taken into account when approaching a learning task. Most learning problems in motor learning require regression networks, for instance, as in the learning of internal models, coordinate transformations, control policies, or evaluation functions in reinforcement learning. Data in motor learning is usually not limited to a finite data set—whenever the robot moves new data are generated and should be included in the learning network. Thus, computationally inexpensive training methods are important in this domain, and on-line learning would be preferred. Among the most significant additional problems of motor learning is that the distributions of the learning data may change continuously. Input distributions change due to the fact that a flexible movement system may work on different tasks at different times, thus creating different kinds of training data. Moreover, the input-output relationship of the data—the conditional distribution—may change when the learning system changes its physical properties or when learning involves nonstationary training data as in reinforcement learning. Such changing distributions easily lead to catastrophic interference in many neural network paradigms, i.e., the unlearning of useful information when training on new data (Schaal and Atkeson, 1998). As a last element, motor learning tasks of complex motor systems can be rather high dimensional in the number of input dimensions, thus increasing the need for efficient learning algorithms. The current trend in learning research is largely orthogonal to the problems of motor learning.

In this paper, we advocate locally weighted learning methods (LWL) for motor learning, a learning technique derived from nonparametric statistics (Atkeson and Schaal, 1995, Cleveland and Loader, 1995, Hastie and Tibshirani, 1994). LWL provides a principled approach to learning models of complex phenomena, dealing with large amounts of data, training quickly, and avoiding interference between multiple tasks during control of complex systems (Atkeson, Moore, and Schaal, 1997a, Atkeson, Moore, and Schaal, 1997b). LWL methods can even deal successfully with high dimensional input data that have redundant and irrelevant inputs while keeping the computational complexity of the algorithms linear in the number of inputs. LWL methods come in two different strategies. Memory-based LWL is a "lazy learning" method (Aha, 1997) that simply stores all training data in memory and uses efficient lookup and interpolation techniques when a prediction for a new input has to be generated. This kind of LWL is useful when data need to be interpreted in flexible ways, for instance, as for-

ward *or* inverse transformations. Memory-based LWL is therefore a "least commitment" approach and very data efficient. Non-memory-based LWL has essentially the same statistical properties as memory-based LWL, but it avoids storing data in memory by using recursive system identification techniques (Ljung and Söderström, 1986). In this way, non-memory-based LWL caches the information about training data in compact representations, at the cost that a flexible re-evaluation of data becomes impossible, but lookup times for new data become significantly faster.

In the following, we will describe four LWL algorithms that are the most suitable to robot learning problems. The goal of the next section is to provide clear pseudo-code explanations of these algorithms. Afterwards, we will illustrate the successful application of some of the methods to real-time robot learning, involving dexterous manipulation tasks such as devil sticking and pole balancing with an anthropomorphic robot arm, and classical problems like the learning of high-dimensional inverse dynamics models.

2 Locally Weighted Learning

In all our algorithms we assume that the data generating model for our regression problems has the standard form $y = f(\mathbf{x}) + \varepsilon$, where $\mathbf{x} \in \Re^n$ is a n-dimensional input vector, the noise term has mean zero, $E\{\varepsilon\} = 0$, and, for simplicity, the output is one-dimensional.

The key concept of our LWL methods is to approximate nonlinear functions by means of piecewise linear models (Cleveland, 1979), similar to a first-order Taylor series expansion. Locally linear models have been demonstrated to be an excellent statistical compromise among the possible local polynomials that can be fit to data (Hastie and Loader, 1993). The key problem in LWL is to determine the region of validity in which a local model can be trusted, and how to fit the local model in this region.

In all following algorithms, we compute the region of validity, called a *receptive field*, of each linear model from a Gaussian kernel:

$$w_k = \exp\left(-\frac{1}{2}\left(\mathbf{x} - \mathbf{c}_k\right)^T \mathbf{D}_k \left(\mathbf{x} - \mathbf{c}_k\right)\right) \tag{1}$$

where $\mathbf{c}_k$ is the center of the k^{th} linear model, and $\mathbf{D}_k$ corresponds to a positive semi-definite distance metric that determines the size and shape of region of validity of the linear model. Other kernel functions are possible (Atkeson et al., 1997a) but add only minor differences to the quality of function fitting.

2.1 Locally Weighted Regression

The most straightforward LWL algorithm with locally linear models is memory-based Locally Weighted Regression (LWR) (Atkeson, 1989, Atkeson, 1992, Atkeson and Schaal, 1995). Training of LWR is very fast: it just requires adding new training data to

the memory. Only when a prediction is needed for a query point $\mathbf{x}_q$, the following weighted regression analysis is performed:

The LWR Algorithm:

Given:

A query point $\mathbf{x}_q$ and p training points $\{\mathbf{x}_i, y_i\}$ in memory

Compute Prediction:

a) compute diagonal weight matrix $\mathbf{W}$

$$\text{where } w_{ii} = \exp\left(-\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_q\right)^T \mathbf{D}\left(\mathbf{x}_i - \mathbf{x}_q\right)\right)$$

b) build matrix $\mathbf{X}$ and vector $\mathbf{y}$ such that (2)

$$\mathbf{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_p)^T \text{ where } \tilde{\mathbf{x}}_i = \left[\left(\mathbf{x}_i - \mathbf{x}_q\right)^T \; 1\right]^T$$

$$\mathbf{y} = \left(y_1, y_2, \ldots, y_p\right)^T$$

c) compute locally linear model

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

d) the prediction for $\mathbf{x}_q$ is thus

$$\hat{y}_q = \beta_{n+1}$$

β_{n+1} denotes the (n+1)$^{\text{th}}$ element of the regression vector β. The computational complexity of LWR is proportional to $O(pn^2)$. Since normally most of the p training data points receive an approximately zero weight as they are too far away from the query point, the computational complexity of LWR can be reduced significantly, particularly when exploiting efficient data structures like *kd*-trees for keeping the data in memory (Moore, 1990). Thus, LWR can be applied efficiently in real-time for problems that are not too high dimensional in the number of inputs n and that do not accumulate too much data in one particular area of the input space.

The only open parameter in (2) is the distance metric $\mathbf{D}$, introduced in Equation (1). After there is a significant amount of data, $\mathbf{D}$ can be optimized by leave-one-out cross validation. To avoid too many open parameters, $\mathbf{D}$ is usually assumed to be a global diagonal matrix $\mathbf{D} = h \cdot diag([n_1, n_2, \ldots, n_n])$, where h is a scale parameter, and the n_i normalize the range of the input dimensions, e.g., by the variance of each input dimension $n_i = 1/\sigma_i^2$. Leave-one-out crossvalidation is thus performed only as a one-dimensional search over the parameter h:

Leave – One – Out Cross Validation:

Given:

a set H of reasonable values h_r

Algorithm:

For all $h_r \in H$:

$sse_r = 0$

For i = 1:p (3)

a) $\mathbf{x}_q = \mathbf{x}_i$

b) temporarily exclude $\{\mathbf{x}_i, y_i\}$ from training data

c) compute LWR prediction $\hat{y}_q$ with reduced data

d) $sse_r = sse_r + \left(y_i - \hat{y}_q\right)^2$

Choose the optimal h_r^* such that $h_r^* = \min_r \{sse_r\}$

Of course, at an increased computational cost, leave-one-out cross validation can also be performed treating all coefficients of the distance metric as open parameters, usually by using gradient descent methods (Atkeson, 1992; Atkeson and Schaal, 1995; Lowe, 1995).

2.2 Locally Weighted Partial Least Squares

Under two circumstances it is necessary to enhance the LWR algorithm above: if the number of input dimensions grows large, or if there are redundant input dimensions such that the matrix inversion in (2) becomes numerically unstable. There is a computational efficient technique from the statistics literature, Partial Least Squares Regression (PLS) (Wold, 1975, Frank and Friedman, 1993, Schaal, Vijayakumar, and Atkeson, 1998, Vijayakumar and Schaal, 2000), that is ideally suited to reduce the computational complexity of LWR and to avoid numerical problems. The essence of PLS is to fit linear models using a hierarchy of univariate regressions along selected projections in input space. The projections are chosen according to the correlation of input and output data, and the algorithm assures that subsequent projections are orthogonal in input space. It is straightforward to derive a locally weighted PLS algorithm (LWPLS), as shown in Equation (4). The only steps in LWPLS that may look unusual at a first glance are the ones indicated by (*) and (**) in Equation (4). At these steps, the input data is regressed against the current projection s (*), and subsequently, the input space is reduced (**), This procedure ensures that the next projection direction $\mathbf{u}_{i+1}$ is guaranteed to be orthogonal with respect to all previous projection directions.

The LWPLS Algorithm:

Given:

a query point $\mathbf{x}_q$ and p training points $\{\mathbf{x}_i, y_i\}$ in memory

Compute Prediction:

a) compute diagonal weight matrix $\mathbf{W}$

where $w_{ii} = \exp\left(-\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_q\right)^T \mathbf{D}\left(\mathbf{x}_i - \mathbf{x}_q\right)\right)$

b) build the matrix $\mathbf{X}$ and vector $\mathbf{y}$ such that

$$\bar{\mathbf{x}} = \sum_{i=1}^{p} w_{ii}\mathbf{x}_i \Big/ \sum_{i=1}^{p} w_{ii};\ \beta_0 = \sum_{i=1}^{p} w_{ii} y_i \Big/ \sum_{i=1}^{p} w_{ii};$$

$$\mathbf{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_p)^T \text{ where } \tilde{\mathbf{x}}_i = \left(\mathbf{x}_i - \bar{\mathbf{x}}\right)$$

$$\mathbf{y} = \left(\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_p\right)^T \text{ where } \tilde{y}_i = \left(y_i - \beta_0\right)$$

c) recursively compute locally linear model

Initialize: $\mathbf{Z}_0 = \mathbf{X}$, $\mathbf{res}_0 = \mathbf{y}$

For i = 1:r (4)

$$\mathbf{u}_i = \mathbf{Z}_{i-1}^T \mathbf{W}\,\mathbf{res}_{i-1}$$

$$\mathbf{s}_i = \mathbf{Z}_{i-1}\mathbf{u}_i$$

$$\beta_i = \frac{\mathbf{s}_i^T \mathbf{W}\,\mathbf{res}_{i-1}}{\mathbf{s}_i^T \mathbf{W}\mathbf{s}_i}$$

$$\mathbf{p}_i = \frac{\mathbf{s}_i^T \mathbf{W}\mathbf{Z}_{i-1}}{\mathbf{s}_i^T \mathbf{W}\mathbf{s}_i} \quad (*)$$

$$\mathbf{res}_i = \mathbf{res}_{i-1} - \mathbf{s}_i \beta_i$$

$$\mathbf{Z}_i = \mathbf{Z}_{i-1} - \mathbf{s}_i \mathbf{p}_i \quad (**)$$

d) the prediction for $\mathbf{x}_q$ is thus

Initialize: $\mathbf{z}_0 = \mathbf{x}_q - \bar{\mathbf{x}}, \hat{y}_q = \beta_0$

For i = 1:r

$$s_i = \mathbf{z}_{i-1}^T \mathbf{u}_i$$

$$\hat{y}_q \leftarrow \hat{y}_q + s_i \beta_i$$

$$\mathbf{z}_i = \mathbf{z}_{i-1} - s_i \mathbf{p}_i^T$$

There is a remarkable property of LWPLS: if the input data is locally statistically independent (i.e., has a diagonal covariance matrix) and is approximately locally linear, LWPLS will find an optimal linear approximation for the data with a *single* projection

(Vijayakumar and Schaal, 2000). This statement is true since LWPLS will immediately find the optimal projection direction, i.e., the gradient of the data. An important question is thus how many projections r should be chosen if the input data are not statistically independent. Typically, the squared error res_i^2 at iteration i should be significantly lower than that of the previous step. Thus, a simple heuristic to stop adding projections is to require that for every new projection, the squared error be reduced at least by a certain ratio:

$$\frac{res_i^2}{res_{i-1}^2} < \phi, \text{ where } \phi \in [0,1] \tag{5}$$

We usually use $\phi = 0.5$ for all our learning tasks. Thus, as in LWR, the only open parameter in LWPLS becomes the distance metric $\mathbf{D}$, which can be optimized according to the strategy in (3).

The computational complexity of LWPLS is $O(rnp)$. If one assumes that most of the data has a zero weight and that only a fixed number of projections are needed to achieve a good fit, the computational complexity tends towards linear in the number of input dimensions. This property constitutes a significant saving over the more than quadratic cost of LWR, particularly in high dimensional input spaces. Additionally, the correlation step to select the projection direction eliminates irrelevant and redundant input dimensions and results in excellent numerical robustness of LWPLS.

2.3 Receptive Field Weighted Regression

Two points of concern remain with LWR and LWPLS. If the learning system receives a large, possibly never ending stream of input data, as is typical in online robot learning, both memory requirements to store all data as well as the computational cost to evaluate algorithms (2) or (4) become too large. Under these circumstances, a non-memory-based version of LWL is desirable such that each new data point is incrementally incorporated in the learning system and lookup speed becomes accelerated.

A first approach to an incremental LWL algorithm was suggested in previous work (Schaal and Atkeson, 1998), using LWR as the starting point. The idea of the algorithm is straightforward: instead of postponing the computation of a local linear model until a prediction needs to be made, local models are built continuously in the entire support area of the input data at selected points in input space (see details below). The prediction for a query point is then formed as the weighted average of the predictions of all local models:

$$\hat{y}_q = \frac{\sum_{k=1}^{K} w_k \hat{y}_{q,k}}{\sum_{k=1}^{K} w_k} \tag{6}$$

The weights in (6) are computed according to the weighting kernel of each local model in Equation (7). Incremental updates of the parameters of the linear models can be accomplished with recursive least squares techniques (Ljung and Söderström, 1986). Thus,

the LWR algorithm from (2) becomes the Receptive Field Weighted Regression (RFWR) algorithm (Schaal and Atkeson, 1998) as given in the following equations (7).

The RFWR Algorithm:

Given:

A training point $(\mathbf{x}, y)$

Update all K local models:

$$w_k = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x}-\mathbf{c}_k)\right) \tag{7}$$

$$\beta_k^{n+1} = \beta_k^n + w_k \mathbf{P}_k^{n+1} \tilde{\mathbf{x}} e_{cv,k} \quad \text{where} \quad \tilde{\mathbf{x}} = \begin{bmatrix} (\mathbf{x}-\mathbf{c}_k)^T & 1 \end{bmatrix}$$

$$\text{and} \quad \mathbf{P}_k^{n+1} = \frac{1}{\lambda}\left(\mathbf{P}_k^n - \frac{\mathbf{P}_k^n \tilde{\mathbf{x}} \tilde{\mathbf{x}}^T \mathbf{P}_k^n}{\frac{\lambda}{w_k} + \tilde{\mathbf{x}}^T \mathbf{P}_k^n \tilde{\mathbf{x}}}\right) \quad \text{and} \quad e_{cv,k} = \left(y - \beta_k^{nT} \tilde{\mathbf{x}}\right)$$

Compute Prediction for query point $\mathbf{x}_q$ **:**

$$\hat{y}_k = \beta_k^T \tilde{\mathbf{x}}_q$$

In the above equations, $\lambda \in [0,1]$ is a forgetting factor that determines how much old data in the regression parameters will be forgotten, similar as in recursive system identification techniques (Ljung and Söderström, 1986). The variables $\mathbf{P}_k$ cache the inverse of the covariance matrix of the input variables and needs to be initialized as $\mathbf{P}_k = \mathbf{I}\, r$, where $\mathbf{I}$ is the identity matrix and r is usually a large number, e.g., 10^{10} .

Thus, as in all the previous LWL algorithms, the only remaining open parameter is the distance metric $\mathbf{D}$. In contrast to the algorithm in (3) that only determined $\mathbf{D}$ as a global parameter to be used everywhere in input space, it is now possible to optimize the $\mathbf{D}_k$ for every local model individually. In (Schaal and Atkeson, 1998) we developed an incremental optimization of $\mathbf{D}$ by means of gradient descent in a stochastic leave-one-out crossvalidation criterion. The following equations summarize the update formulae. Since the gradient descent update is the same for all local models, the index k is omitted.

RFWR Gradient descent update of D:

Cost Function for Gradient Descent:

$$J = \frac{1}{\sum_{i=1}^{p} w_i} \sum_{i=1}^{p} \frac{w_i \left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|^2}{\left(1 - w_i \tilde{\mathbf{x}}_i^T \mathbf{P} \tilde{\mathbf{x}}_i\right)^2} + \gamma \sum_{i,j=1}^{n} D_{ij}^2$$

Recursive Update Algorithm:

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \quad \text{with } \mathbf{D} = \mathbf{M}^T\mathbf{M}, \text{where } \mathbf{M} \text{ is upper triangular}$$

$$W^{n+1} = \lambda W^n + w$$

$$E^{n+1} = \lambda E^n + w\, e_{cv}^2$$

$$\mathbf{H}^{n+1} = \lambda \mathbf{H}^n + \frac{w\,\tilde{\mathbf{x}}\, e_{cv}}{1-h}, \quad \text{where} \quad h = w\,\tilde{\mathbf{x}}^T \mathbf{P}^{n+1}\,\tilde{\mathbf{x}}$$

$$\mathbf{R}^{n+1} = \lambda \mathbf{R}^n + \frac{w^2\, e_{cv}^2 \tilde{\mathbf{x}}\,\tilde{\mathbf{x}}^T}{1-h}$$

$$\frac{\partial J}{\partial \mathbf{M}} \approx \frac{\partial w}{\partial \mathbf{M}} \sum_{i=1}^{p} \frac{\partial J_{1,i}}{\partial w} + \frac{w}{W^{n+1}} \frac{\partial J_2}{\partial \mathbf{M}} \tag{8}$$

where:

$$\frac{\partial w}{\partial M_{rl}} = -\frac{1}{2} w (\mathbf{x}-\mathbf{c})^T \frac{\partial \mathbf{D}}{\partial M_{rl}} (\mathbf{x}-\mathbf{c}), \quad \frac{\partial J_2}{\partial M_{rl}} = 2\gamma \sum_{i,j=1}^{n} D_{ij} \frac{\partial D_{ij}}{\partial M_{rl}}$$

$$\frac{\partial D_{ij}}{\partial M_{rl}} = \delta_{lj} M_{ri} + \delta_{il} M_{rj} \quad (\delta \text{ is the Kronecker operator})$$

$$\sum_{i=1}^{p} \frac{\partial J_{1,i}}{\partial w} \approx -\frac{E^{n+1}}{\left(W^{n+1}\right)^2} +$$

$$\frac{1}{W^{n+1}} \left(e_{cv}^2 - \left(2\,\mathbf{P}^{n+1}\,\tilde{\mathbf{x}} \left(y - \tilde{\mathbf{x}}^T \beta^{n+1} \right) \right) \otimes \mathbf{H}^n - \left(2\,\mathbf{P}^{n+1}\,\tilde{\mathbf{x}}\,\tilde{\mathbf{x}}^T \mathbf{P}^{n+1} \right) \otimes \mathbf{R}^n \right)$$

where the operator $\otimes$ denotes an element-wise multiplication of two homomorphic matrices or vectors with a subsequent summation of all coefficients, $\mathbf{Q} \otimes \mathbf{V} = \Sigma Q_{ij} V_{ij}$. All recursive variables are initialized with zeros, except for the initial distance metric that is set to a manually chosen default value $\mathbf{D}=\mathbf{D}_{def}$.

The above learning rules can be embedded in an incremental learning system that automatically allocates new locally linear models as needed (Schaal and Atkeson, 1998):

Initialize RFWR with no receptive field (RF);
For every new training sample (**x**,*y*):
  For *k*=1 to *K*:
    calculate the activation from (1)
    update according to (7) and (8)
  end;
  If no linear model was activated by more than w_{gen};
    create a new RF with **c**=**x**, **D**=$\mathbf{D}_{def}$
  end;
end;

In this pseudo-code algorithm, w_{gen} is a threshold that determines when to create a new receptive field, e.g., w_{gen}=0.1, and $\mathbf{D}_{def}$ is the initial (usually diagonal) distance metric in Equation (1). The decomposition of $\mathbf{D}$ into $\mathbf{M}^T\mathbf{M}$ can be achieved with a Cholesky decomposition (Press, Flannery, Teukolsky, and Vetterling, 1989).

2.4 Locally Weighted Projection Regression

While RFWR is a powerful algorithm for incremental function approximation, it becomes computationally too expensive in high dimensional spaces. For such cases, LWPLS can be reformulated as an incremental algorithm, too, called Locally Weighted Projection Regression (LWPR). The update equations for one local model become:

Locally Weighted Projection Regression

Given: A training point $(\mathbf{x}, y)$

Update the means of inputs and output:

$$\mathbf{x}_0^{n+1} = \frac{\lambda W^n \mathbf{x}_0^n + w\,\mathbf{x}}{W^{n+1}}$$

$$\beta_0^{n+1} = \frac{\lambda W^n \beta_0^{n+1} + w\,y}{W^{n+1}}$$

where $W^{n+1} = \lambda W^n + w$

Update the local model:

Initialize: $\mathbf{z}_0 = \mathbf{x} - \mathbf{x}_0^{n+1}$, $res_0 = y - \beta_0^{n+1}$

For $i = 1{:}\,r$,

$$\text{a)} \quad \mathbf{u}_i^{n+1} = \lambda\,\mathbf{u}_i^n + w\,\mathbf{z}_{i-1}\,res_{i-1} \tag{9}$$

$$\text{b)} \quad s_i = \mathbf{z}_{i-1}^T \mathbf{u}_i^{n+1}$$

$$\text{c)} \quad SS_i^{n+1} = \lambda\,SS_i^n + w\,s_i^2$$

$$\text{d)} \quad SR_i^{n+1} = \lambda\,SR_i^n + w\,s_i^2\,res_{i-1}$$

$$\text{e)} \quad SZ_i^{n+1} = \lambda\,SZ_i^n + w\,\mathbf{z}_{i-1}\,s_i$$

$$\text{f)} \quad \beta_i^{n+1} = SR_i^{n+1} / SS_i^{n+1}$$

$$\text{g)} \quad \mathbf{p}_i^{n+1} = SZ_i^{n+1} / SS_i^{n+1}$$

$$\text{h)} \quad \mathbf{z}_i = \mathbf{z}_{i-1} - s_i \mathbf{p}_i^{n+1}$$

$$\text{i)} \quad res_i = res_{i-1} - s_i \beta_i^{n+1}$$

$$\text{j)} \quad SSE_i^{n+1} = \lambda\,SSE_i^n + w\,res_i^2$$

In the above equations, the variables *SS*, *SR*, and *SZ* are memory terms that enable us to achieve the univariate regression in step f) in a recursive least squares fashion, i.e., a fast Newton-like method as in RFWR (cf. Equation (7)). The other steps are incremental

counterparts of the LWPLS algorithm above. Step j) computes the sum of squared errors that is used to determine when to stop adding projections according to (5). Predictions for a query point are formed exactly as in Eqn (4)-d.

In analogy with RFWR, an incremental update of the distance metric $\mathbf{D}$ can be derived based on stochastic one-leave-out cross validation. Due to the hierarchical fitting procedure of PLS, the one-leave-out crossvalidation cost function can only be formulated in approximation by treating all projections as independent:

LWPR Gradient descent update of D:

Cost Function for Gradient Descent:

$$J = \frac{1}{\sum_{i=1}^{p} w_i} \sum_{i=1}^{p} \sum_{k=1}^{r} \frac{w_i res_{k,i}^2}{\left(1 - w_i \frac{s_{k,i}^2}{\mathbf{s}_k^T \mathbf{W} \mathbf{s}_k}\right)^2} + \gamma \sum_{i,j=1}^{n} D_{ij}^2$$

Recursive Update Algorithm:

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \quad \text{with} \quad \mathbf{D} = \mathbf{M}^T \mathbf{M}, \text{where } \mathbf{M} \text{ is upper triangular}$$

$$W^{n+1} = \lambda W^n + w$$

$$E_k^{n+1} = \lambda E_k^n + w\, res_{k-1}^2$$

$$\mathbf{H}_k^{n+1} = \lambda \mathbf{H}_k^n + \frac{w\, s_k\, res_{k-1}}{1 - h_k}, \quad \text{where} \quad h_k = w \frac{s_k^2}{SS_k^{n+1}} \tag{10}$$

$$\mathbf{R}_k^{n+1} = \lambda \mathbf{R}_k^n + \frac{w^2\, res_{k-1}^2 s_k^2}{1 - h_k}$$

$$\frac{\partial J}{\partial \mathbf{M}} \approx \frac{\partial w}{\partial \mathbf{M}} \sum_{i=1}^{p} \sum_{k=1}^{r} \frac{\partial J_{1,r,i}}{\partial w} + \frac{w}{n\, W^{n+1}} \frac{\partial J_2}{\partial \mathbf{M}}$$

where:

$$\frac{\partial w}{\partial M_{rl}} = -\frac{1}{2} w (\mathbf{x} - \mathbf{c})^T \frac{\partial \mathbf{D}}{\partial M_{rl}} (\mathbf{x} - \mathbf{c}), \quad \frac{\partial J_2}{\partial M_{rl}} = 2\gamma \sum_{i,j=1}^{n} D_{ij} \frac{\partial D_{ij}}{\partial M_{rl}}$$

$$\frac{\partial D_{ij}}{\partial M_{rl}} = \delta_{lj} M_{ri} + \delta_{il} M_{rj} \quad (\delta \text{ is the Kronecker operator})$$

$$\sum_{i=1}^{p} \sum_{k=1}^{r} \frac{\partial J_{1,r,i}}{\partial w} \approx \sum_{k=1}^{r} \left[-\frac{E_k^{n+1}}{\left(W^{n+1}\right)^2} + \frac{1}{W^{n+1}} \left(res_{k-1}^2 - 2 \frac{res_k s_k}{SS_k^{n+1}} \mathbf{H}_k^n - 2 \left(\frac{s_k}{SS_k^{n+1}} \right)^2 \mathbf{R}_k^n \right) \right]$$

Thus, the ensuing learning algorithm is the same as for RFWR by using the update equations (9) and (10) instead of (7) and (8), and by initializing a new receptive field with r=2, i.e., two initial projections in order to allow deciding whether to add new projections according to Equation (5). For a diagonal distance metric $\mathbf{D}_k$ and under the

assumption that r remains small, the computational complexity of the update of all parameters of LWPR is linear in the number of input dimensions. Thus, LWPR constitutes the computationally most efficient algorithm in the series of Locally Weighted Learning algorithms, and, as will be demonstrated below, even works successfully in very high-dimensional input spaces.

3 Empirical Evaluations

The following examples illustrate how our LWL techniques have been applied to various robot learning problems. In common to all these application is that the learning algorithm acquired an internal model of a dynamic system, and that this model was used subsequently in designing a controller for a robot task. Such model-based control and learning has also become increasingly more an accepted hypothesis of how biological movements acquire and perform skilled movement (Kawato, 1999; Wolpert, Miall, and Kawato, 1998).

3.1 Learning Devil-Sticking

Devil sticking is a juggling task where a center stick is batted back and forth between two handsticks (Figure 1a). Figure 1b shows a sketch of our devil-sticking robot. The robot uses its top two joints to perform planar devil-sticking; more details can be found in (Schaal and Atkeson, 1994). The task of the robot is to learn a continuous left-right-left-etc. juggling pattern. For learning, the task is modeled as a discrete function that maps impact states on one hand to impact states on the other hand. A state is given as a five dimensional vector $\mathbf{x} = (p, \theta, \dot{x}, \dot{y}, \dot{\theta})^T$, comprising impact position, angle, and velocities of the center of the devil stick and angular velocity (Figure 1b), respectively. The task command $\mathbf{u} = (x_h, y_h, \dot{\theta}_t, v_x, v_y)^T$ is given by a catch position (x_h, y_h), an angular trigger velocity $(\dot{\theta}_t)$when to start the throw, and the two dimensional throw direction (v_x, v_y). In order to compute appropriate Linear Quadratic Regulator (LQR) controllers for this task (Dyer and McReynolds, 1970), the robot learns the nonlinear mapping between the current state and command, and the next state, i.e., a 10 dimensional input to five dimensional output function. This task is ideally suited for LWR as it is not too high dimensional and new training data are only generated at about 1-2Hz, i.e., whenever the center stick hits one of the handsticks. Moreover, memory-based learning also supports efficient search of the state-action space for statistically good new commands (Schaal and Atkeson, 1994). As a result, successful (i.e., more than 1000 consecutive hits) devil sticking could be achieved in about 40-80 trials, corresponding to about 300-800 training points in memory (Figure 2). This is a remarkable learning rate given that humans need about one week of 1 hour practicing a day before they learn to juggle the devilstick. The final performance of the robot was also quite superior to our early attempts to implement the task based on classical system identification and controller design, which only accomplished up to 100 consecutive hits.

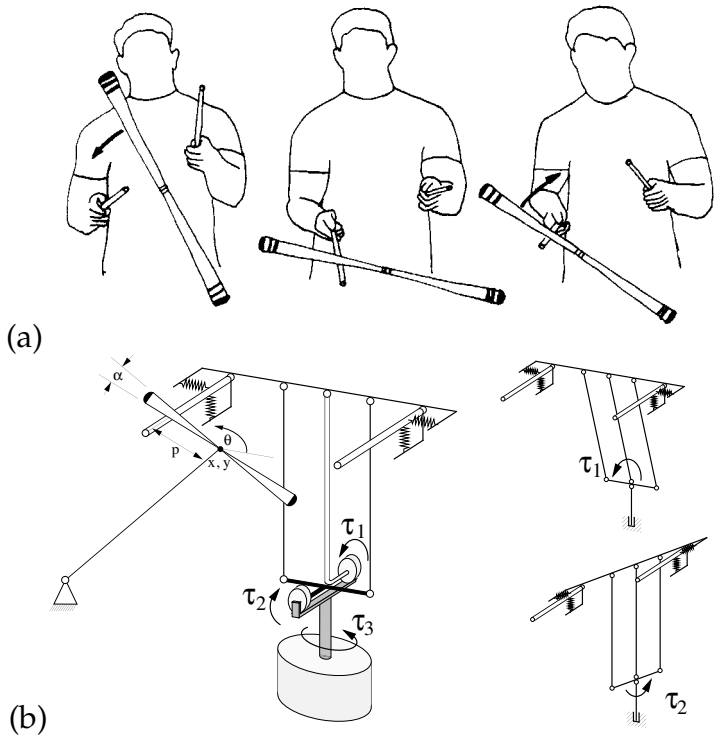


Figure 1: (a) an illustration of devil sticking, (b) sketch of our devil sticking robot: the flow of force from each motor into the robot is indicated by different shadings of the robot links, and a position change due to an application of motor 1 or motor 2, respectively, is indicated in the small sketches

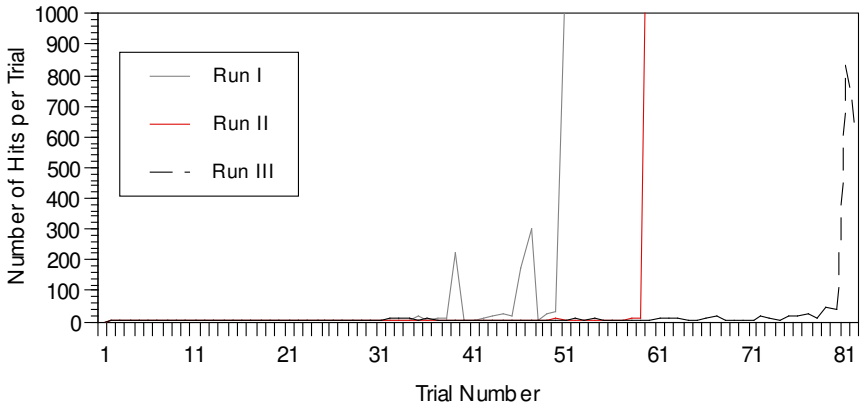


Figure 2: Learning curves of devil sticking for three learning runs.

3.2 Learning Pole Balancing

We implemented learning of the task of balancing a pole on a fingertip with a 7-degree-of-freedom anthropomorphic robot arm (Figure 3a). The low-level robot controller ran in a compute-torque mode (Craig, 1986) at 480Hz out of 8 parallel processors located in a VME bus, running the real-time operating system vxWorks. The inverse dynamics

model of the robot had been estimated using established parameter estimation techniques from control theory (An, Atkeson, and Hollerbach, 1988). The goal of learning was to generate appropriate task level commands, i.e., Cartesian accelerations of the fingertip, to keep the pole upright. Task level commands were converted to actuator space by means of the extended Jacobian method (Baillieul and Martin, 1990). As input, the robot received data from its color-tracking-based stereo vision system with more than 60ms processing delays. Learning was implemented on-line using RFWR. The task of RFWR was to acquire a discrete time forward dynamics model of the pole that was both used to compute an LQR controller and to realize a Kalman predictor to eliminate the delays in visual input. The forward model had 12 input dimensions (3 positions of the lower pole end, 2 angular positions, the corresponding 5 velocities, and 2 horizontal accelerations of the fingertip) that are mapped to 10 outputs, i.e., the next state of the pole. The robot only received training data when it actually moved.

Figure 3b shows the results of learning. It took about 10-20 trials before learning succeeded in reliable performance longer than one minute. We also explored learning from demonstration, where a human demonstrated how to balance a pole for 30 seconds while the robot was learning the forward model by just "watching". From the demonstration, the robot could extract the forward dynamics model of the pole-balancing task and synthesize the LQR controller. Now learning was reliably accomplished in one *single* trial, using a large variety of physically different poles and using demonstrations from arbitrary people in the laboratory.

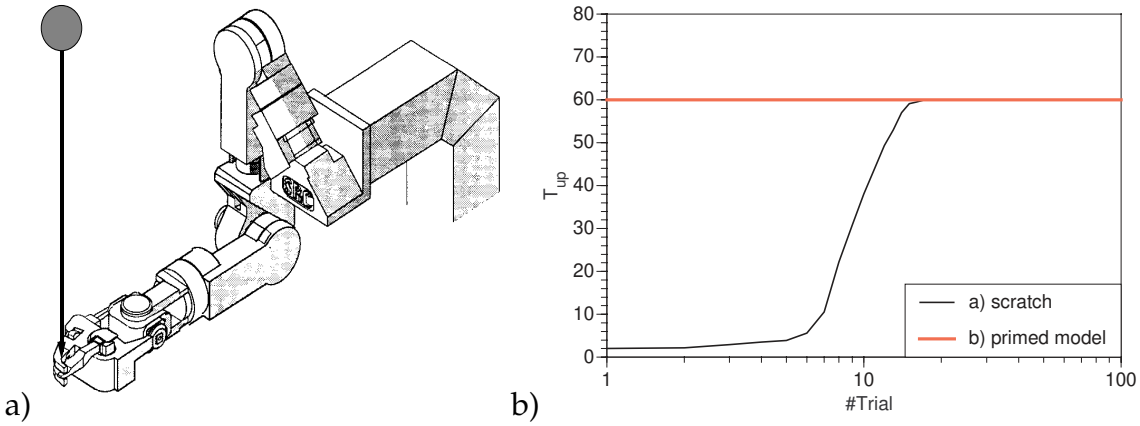


Figure 3: a) Sarcos Dexterous Robot Arm; b) Smoothed average of 10 learning curves of the robot for pole balancing. The trials were aborted after successful balancing of 60 seconds. We also tested long term performance of the learning system by running pole balancing for over an hour—the pole was never dropped.

3.3 Inverse Dynamics Learning

As all our anthropomorphic robots are rather light-weight, compliant, and driven by direct-drive hydraulic motors, using methods from rigid body dynamics to estimate their

inverse models resulted in rather inaccurate performance due to unknown nonlinearities in these systems. Therefore, the goal of the following learning experiments was to approximate the inverse dynamics of a 7-degree-of-freedom anthropomorphic robot arm (Figure 3a) from a data set consisting of 45,000 data points, collected at 100Hz from the actual robot performing various rhythmic and discrete movement tasks (this corresponds to 7.5 minutes of data collection). The data consisted of 21 input dimensions: 7 joint positions, velocities, and accelerations. The goal of learning was to approximate the appropriate torque command of the shoulder motor in response to the input vector. To increase the difficulty of learning, we added 29 irrelevant dimensions to the inputs with $N(0,0.05^2)$ Gaussian noise. 5,000 data points were excluded from the training data as a test set.

The high dimensional input space of this learning problem requires an application of LWPR. Figure 4 shows the learning results in comparison to parametric estimation of the inverse dynamics based on rigid body dynamics (An et al., 1988), and in comparison to a sigmoidal feedforward neural network with 30 hidden units trained with Levenberg-Marquardt optimization. From the very beginning, LWPR outperformed the parametric model. Within about 500,000 training points (about 30 minutes training on a 500Mhz DEC Alpha Computer), LWPR converged to the excellent result of $nMSE$=0.042. It employed an average of only 3.8 projections per local model despite the fact that the input dimensionality was 50. During learning, the number of local models increased by a factor of 6 from about 50 initial models to about 310 models. This increase is due to the adjustment of the distance metric $\mathbf{D}$ in Equation (10) that was initialized to form a rather large kernel. Since this large kernel oversmoothes the data, LWPR reduced the kernel size, and in response more kernels needed to be allocated. The sigmoidal neural network achieved the same learning results as LWPR, however, it required one night of training and could only operate in batch mode.

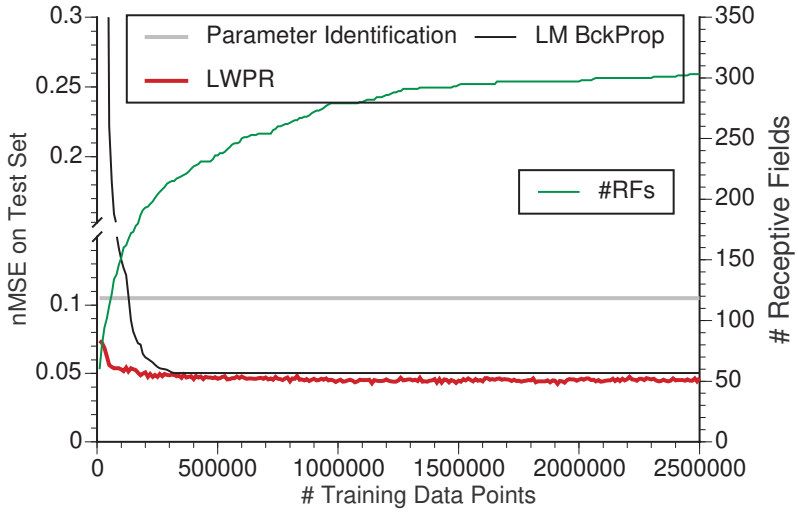


Figure 4: Learning curve for learning the inverse dynamics model of the robot from a 50 dimensional data set that included 29 irrelevant dimensions.

We also applied LWPR to an even more complex robot, a 30 DOFs humanoid robot as shown in Figure 5a. Again, we learned the inverse dynamics model for the shoulder motor, however, this time involving 90 input dimensions (i.e., 30 positions, 30 velocities, and 30 accelerations of all DOFs). The learning results, shown in Figure 5b, are similar to Figure 4. Very quickly, LWPR outperformed the inverse dynamics model estimated from rigid body dynamics and settled at a result that was more than three times more accurate. The huge learning space required more than 2000 local models, using about 2.5 local projections on average. In our real-time implementation of LWPR on this robot, the learned models achieve by far better tracking performance than the parameter estimation techniques.

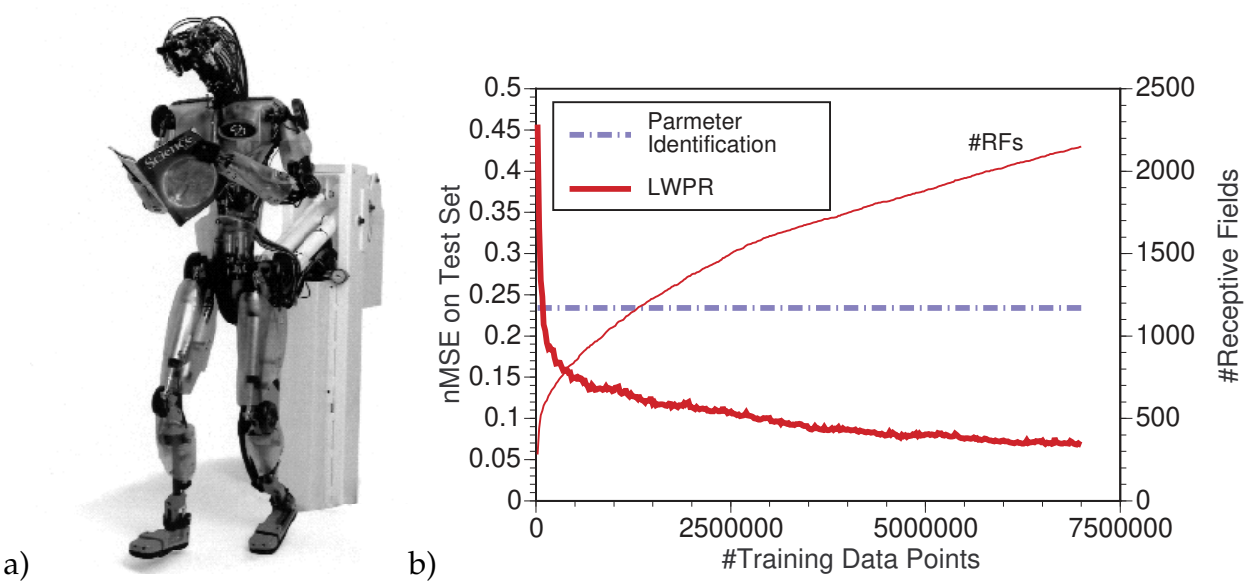


Figure 5: a) Humanoid robot in our laboratory; b) inverse dynamics learning for the right shoulder motor of the humanoid.

4 Conclusions

This paper presented Locally Weighted Learning algorithms for real-time robot learning. The algorithms are easy to implement, use sound statistical learning techniques, converge quickly to accurate learning results, and can be implemented in a purely incremental fashion. We demonstrated that the latest version of our algorithms is capable of dealing with high dimensional input spaces that even have redundant and irrelevant input dimensions while the computational complexity of an incremental update remained linear in the number of inputs. In several examples, we demonstrated how LWL algorithms were applied successfully to complex learning problems with actual robots. From the view point of function approximation, LWL algorithms are competitive methods of supervised learning of regression problem and achieve results that are comparable with state-of-the-art learning techniques. However, what makes the presented algo-

rithms special is their learning speed, numerical robustness in high dimensional spaces, and ability to learn incrementally. To the best of our knowledge, there is currently no comparable learning framework that combines all the required properties for real-time motor learning as well as Locally Weighted Learning. Our learning results demonstrate that autonomous learning can be applied successfully to rather complex robotic systems, and that learning can achieve performance that outperforms traditional techniques from control theory.

5 Acknowledgments

This work was made possible by award 9710312 and 9711770 of the National Science Foundation, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Human Information Processing Research Laboratories.

6 References

Aha, D (1997). Lazy Learning. Artificial Intelligence Review:325-337.

An, CH, Atkeson, CG, Hollerbach, JM (1988). Model-based control of a robot manipulator. Cambridge, MA: MIT Press.

Atkeson, CG (1989). Using local models to control movement. In D. Touretzky (Ed.), Advances in Neural Information Processing Systems 1. San Mateo, CA: Morgan Kaufmann, pp 157-83.

Atkeson, CG (1992). Memory-based approaches to approximating continuous functions. In M. Casdagli, & S. Eubank (Eds.), Nonlinear Modeling and Forecasting. Redwood City, CA: Addison Wesley, pp 503-521.

Atkeson, CG, Moore, AW, Schaal, S (1997a). Locally weighted learning. Artificial Intelligence Review 11:11-73.

Atkeson, CG, Moore, AW, Schaal, S (1997b). Locally weighted learning for control. Artificial Intelligence Review 11:75-113.

Atkeson, CG, Schaal, S (1995). Memory-based neural networks for robot learning. Neurocomputing 9:1-27.

Baillieul, J, Martin, DP (1990). Resolution of kinematic redundancy, Proceedings of Symposia in Applied Mathematics: American Mathematical Society, pp 49-89.

Bishop, CM (1995). Neural networks for pattern recognition. New York: Oxford University Press.

Cleveland, WS (1979). Robust locally weighted regression and smoothing scatterplots. Journal of the American Statistical Association 74:829-836.

Cleveland, WS, Loader, C (1995). Smoothing by local regression: Principles and methods: AT&T Bell Laboratories Murray Hill NY.

Cortes, C, Vapnik, V (1995). Support vector networks. Machine Learning 20:273-297.

Craig, JJ (1986). Introduction to robotics. Reading, MA: Addison-Wesley.

Dyer, P, McReynolds, SR (1970). The computation and theory of optimal control. New York: Academic Press.

Frank, IE, Friedman, JH (1993). A statistical view of some chemometric regression tools. Technometrics 35:109-135.

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

Hastie, TJ, Tibshirani, RJ (1994). Nonparametric regression and classification: Part I: Nonparametric regression. In V. Cherkassky, J. H. Friedman, & H. Wechsler (Eds.), From Statistics to Neural Networks: Theory and Pattern Recognition Applications. ASI Proceedings, subseries F, Computer and Systems Sciences: Springer, pp 120-143.

Kawato, M (1999). Internal models for motor control and trajectory planning. Curr Opin Neurobiol 9:718-727.

Ljung, L, Söderström, T (1986). Theory and practice of recursive identification: Cambridge MIT Press.

Lowe, DG (1995). Similarity metric learning for a variable-kernel classifier. Neural Comput 7:72-85.

Moore, AW (1990). Efficient memory-based learning for robot control: Computer Laboratory University of Cambridge October 1990.

Press, WP, Flannery, BP, Teukolsky, SA, Vetterling, WT (1989). Numerical recipes in C – The art of scientific computing. Cambridge, MA: Press Syndiacate University of Cambridge.

Schaal, S (1999). Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences 3:233-242.

Schaal, S, Atkeson, CG (1994). Robot juggling: An implementation of memory-based learning. Control Systems Magazine 14:57-71.

Schaal, S, Atkeson, CG (1998). Constructive incremental learning from only local information. Neural Comput 10:2047-2084.

Schaal, S, Vijayakumar, S, Atkeson, CG (1998). Local dimensionality reduction. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), Advances in Neural Information Processing Systems 10. Cambridge, MA: MIT Press, pp 633-639.

Vapnik, V, Golowich, S, Smola, A (1996). Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.), Advances in Neural Information Processing Systems 9. Cambridge, MA: MIT Press, pp 281-287.

Vapnik, VN (1982). Estimation of dependences based on empirical data. Berlin: Springer.

Vijayakumar, S, Schaal, S (2000). Locally weighted projection regression, Proceedings of the Seventeenth International Conference (ICML 2000). Stanford, CA.

Williams, CKI, Rasmussen, CE (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), Advances in Neural Information Processing Systems 8. Cambridge, MA: MIT Press, pp 514-520.

Wold, H (1975). Soft modeling by latent variables: the nonlinear iterative partial least squares approach. In J. Gani (Ed.), Perspectives in Probability and Statistics, Papers in Honour of M. S. Bartlett. London: Academic Press, pp 520-540.

Wolpert, DM, Miall, RC, Kawato, M (1998). Internal models in the cerebellum. Trends in Cognitive Sciences 2:338-347.

Locally Weighted Projection Regression : An *O(n)* Algorithm for Incremental Real Time Learning in High Dimensional Space

Sethu Vijayakumar SETHU@USC.EDU
Stefan Schaal SSCHAAL@USC.EDU
Dept. of Computer Science & Neuroscience and Kawato Dynamic Brain Project
HEDCO Neuroscience Bldg HNB 103, University of Southern California, Los Angeles, CA 90089-2520, USA

Abstract

Locally weighted projection regression is a new algorithm that achieves nonlinear function approximation in high dimensional spaces with redundant and irrelevant input dimensions. At its core, it uses locally linear models, spanned by a small number of univariate regressions in selected directions in input space. This paper evaluates different methods of projection regression and derives a nonlinear function approximator based on them. This nonparametric local learning system i) learns rapidly with second order learning methods based on incremental training, ii) uses statistically sound stochastic cross validation to learn iii) adjusts its weighting kernels based on local information only, iv) has a computational complexity that is linear in the number of inputs, and v) can deal with a large number of - possibly redundant - inputs, as shown in evaluations with up to 50 dimensional data sets. To our knowledge, this is the first truly incremental spatially localized learning method to combine all these properties.

1. Introduction

Nonlinear function approximation with high dimensional input data remains a nontrivial problem. An ideal algorithm for such tasks needs to eliminate redundancy in the input data, detect irrelevant input dimensions, keep the computational complexity low, and, of course, achieve accurate function approximation and generalization. A route to accomplish these goals can be sought in techniques of projection regression. Projection Regression(PR) copes with high dimensional inputs by decomposing multivariate regressions into a superposition of single variate regressions along particular projections in input space. The major difficulty of PR lies in how to select efficient projections, i.e., how to achieve the best fitting result with as few as possible one dimensional regressions.

Previous work has focused on finding good *global* projections for fitting nonlinear one-dimensional functions. Among the best known algorithms is projection pursuit regression (Friedman & Stutzle, 1981), and its generalization in form of Generalized Additive Models (Hastie & Tibshirani, 1990). Sigmoidal neural networks can equally be conceived of as a method of projection regression, in particular when new projections are added sequentially, e.g., as in Cascade Correlation (Fahlman & Lebiere, 1990).

In this paper we suggest an alternative method of projection regression, focusing on finding efficient *local* projections. Local projections can be used to accomplish local function approximation in the neighborhood of a given query point. Such methods allow to fit locally simple functions, e.g., low order polynomials, along the projection, which greatly simplifies the function approximation problem. Local projection regression can thus borrow most of its statistical properties from the well established methods of locally weighted learning and nonparametric regression (Hastie & Loader, 1993; Atkeson, Moore & Schaal, 1997). Counterintuitive to the curse of dimensionality (Scott, 1992), local regression methods can work successfully in high dimensional spaces as shown in a recent work (Vijayakumar & Schaal, 1998). In the above work, using techniques of principal component regression (Schaal, Vijayakumar & Atkeson, 1998), the observation that globally high dimensional movement data usually lie on locally low dimensional distributions was exploited. However, principal component regression does not address an efficient selection of local projections, nor is it well suited to detect irrelevant input dimensions. This paper will explore methods that can remedy these shortcoming. We will introduce a novel algorithm, covariance projection regression, that generalizes principal component regression to a family of algorithms capable of discovering efficient projections for locally weighted linear regression and compare it to partial least squares regression–one of the most successful global linear projection regression methods. Empirical evaluations highlight

Table 1. Pseudocode implementation of PLS, PCR and CPR projection regression

PLS/PCR/CPR Pseudocode

1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\mathbf{y}_{res} = \mathbf{y}$
2. **for** $i = 1$ **to** k **do**
   (a) $\mathbf{X}_t = \mathbf{X}_{res}\mathbf{T}$, where $\mathbf{T}$ is a diagonal weight matrix.
   (b) If **[PLS]:** $\mathbf{u}_i = \mathbf{X}_t^T \mathbf{y}_{res}$. If **[PCR/CPR]:** $\mathbf{u}_i = [eigenvector(\mathbf{X}_t^T \mathbf{X}_t)]_{max}$.
   (c) $\beta_i = \mathbf{s}_i^T \mathbf{y}_{res}/(\mathbf{s}_i^T \mathbf{s}_i)$ where $\mathbf{s}_i = \mathbf{X}_{res}\mathbf{u}_i$.
   (d) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{s}_i\beta_i$.
   (e) $\mathbf{X}_{res} = \mathbf{X}_{res} - \mathbf{s}_i\mathbf{p}_i{}^T$ where $\mathbf{p}_i = \mathbf{X}_{res}^T \mathbf{s}_i/(\mathbf{s}_i^T \mathbf{s}_i)$.

the pros and cons of the different methods. Finally, we embed one of the projection regression algorithms in an incremental nonlinear function approximation (Vijayakumar & Schaal, 1998). In several evaluations, the resulting incremental learning system demonstrates high accuracy for function fitting in high dimensional spaces, robustness towards irrelevant inputs, as well as low computational complexity.

2. Linear Projection Regression for Dimensionality Reduction

In this section we will outline several PR algorithms that fit linear functions along the individual projections. Later, by spatially localizing these algorithms, they can serve as the core of nonlinear function approximation techniques. We assume that our data is generated by the standard linear regression model $y = \mathbf{x} * \beta + \epsilon$, where $\mathbf{x}$ is a vector of input variables and y is the scalar, mean-zero noise contaminated output. Without loss of generality, both inputs and output are assumed to be mean zero. For notational convenience, all input vectors are summarized in the rows of the matrix $\mathbf{X}$=$[\mathbf{x}_1\ \mathbf{x}_2\ ...\mathbf{x}_M]^T$ and the corresponding outputs are the elements of the vector $\mathbf{y}$. M is the number of training data and N is the dimensionality of the input data. All the PR techniques considered here project the input data $\mathbf{X}$ onto k orthogonal directions $\mathbf{u}_1, ..., \mathbf{u}_k$ along which they carry out univariate linear regressions - hence, the name projection regression. If the linear model of the data was known, it would be straightforward to determine the optimal projection direction: it is given by the vector of regression coefficients β, i.e., the gradient; along this direction, a single univariate regression would suffice to obtain an optimal regression result.

2.1 Partial Least Squares

Partial least squares (PLS) (Wold, 1975; Frank & Friedman, 1993), a technique extensively used in chemometrics, recursively computes orthogonal projections of the input data and performs single variable regressions along these projections on the residuals of the previous iteration step. Table 1 illustrates PLS in a pseudocode implementation. It should be noted that for PLS, the matrix $\mathbf{T}$ in step 2a of the algorithm needs to be the identity matrix. The key ingredient in PLS is to use the direction of maximal correlation between the residual error and the input data as the projection direction at every regression step. Additionally, PLS regresses the inputs of the previous step against the projected inputs $\mathbf{s}$ in order to ensure the orthogonality of all the projections $\mathbf{u}$ (Step 2d,2e). Actually, this additional regression could be avoided by replacing $\mathbf{p}$ with $\mathbf{u}$ in Step 2e, similar to techniques used in principal component analysis(Sanger, 1989). However, using this regression step leads to better performance of the algorithm. This effect is due to the fact that PLS chooses the most effective projections if the input data has a spherical distribution: with only one projection, PLS will find the direction of the gradient and achieve optimal regression results. The regression step in 2e modifies the input data $\mathbf{X}_{res}$ such that each resulting data vectors have coefficients of minimal magnitude and, hence, push the distribution of $\mathbf{X}_{res}$ to become more spherical.

2.2 Principal Component Regression

A computationally efficient technique of dimensionality reduction for linear regression is Principal Component Regression (PCR) (Massy, 1965; Vijayakumar & Schaal, 1998). PCR projects the input data onto its principal components and performs univariate regressions in these directions. Only those k principal components are used that correspond to the largest eigenvalues of the input covariance matrix. The algorithm for PCR is almost identical to PLS, with $\mathbf{T}$ again being the identity matrix. Only Step 2b in Table 1 is different, but this difference is essential. PCR chooses projection $\mathbf{u}$ solely based on the input distribution. Although this can be interpreted as a method that maximizes the confidence in the univariate regressions, it

is prone to choose quite inefficient projections.

2.3 Covariant Projection Regression

In this section, we introduce a new algorithm which has the flavour of both PCR and PLS. Covariant Projection Regression(CPR) transforms the input data in Step 2a (Table 1) by a (diagonal) weight matrix $\mathbf{T}$ with the goal to elongate the distribution in the direction of the gradient of the input/output relation of the data. Subsequently, the major principal component of this deformed distribution is chosen as the direction of a univariate regression (Step 2b). In contrast to PCR, this projection now reflects not only the influence of the input distribution but also that of the regression outputs. As in PLS, if the input distribution is spherical, CPR will obtain an optimal regression result with a single univariate fit, irrespective of the actual input dimensionality.

CPR is actually a family of algorithms depending on how the weights in $\mathbf{T}$ are chosen. Here, we consider two such options:

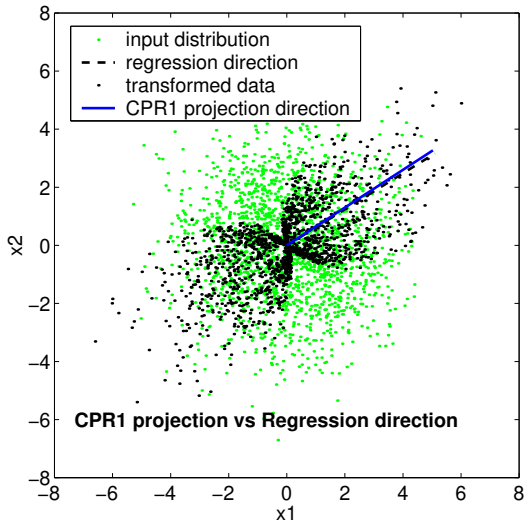
Weighting scheme CPR1: $T_{ii}^{cpr1} = \frac{1}{\sigma_{\mathbf{x}_{res}}}\left(\frac{|y_{res,i}|}{\|\mathbf{x}_{res,i}\|}\right)^p$. See Fig.1(a).

Weighting scheme CPR2: $T_{ii}^{cpr2} = \frac{1}{\|\mathbf{x}_{res,i}\|}\left(\frac{|y_{res,i}|}{\|\mathbf{x}_{res,i}\|}\right)^p$. See Fig.1(b).

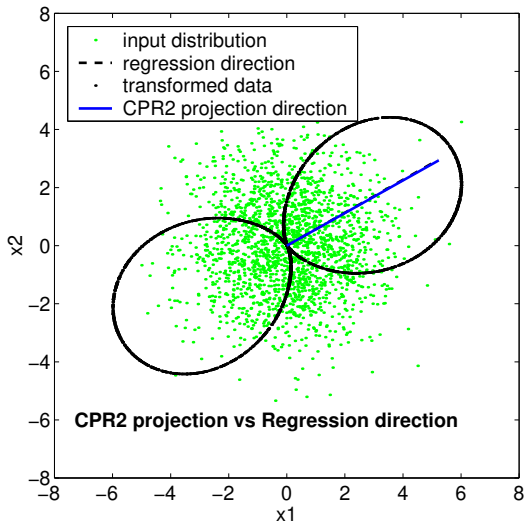
CPR1 spheres the input data and then weights it by the "slope" of each data point, taken to the power p for increasing the impact of the input/output relationships. CPR2 is a variant that, although a bit idiosyncratic, had the best average performance in practice: CPR2 first maps the input data onto a unit-(hyper)sphere, and then stretches the distribution along the direction of maximal slope, i.e., the regression direction (Fig.1) – this method is fairly insensitive to noise in the input data. Fig.1 shows the effect of transforming a gaussian input distribution by the CPR weighting schemes. Additionally, the figure also compares the regression gradient against the projection direction extracted by CPR. As can be seen, for gaussian distributions CPR finds the optimal projection direction with a single projection.

2.4 Monte Carlo evaluations for performance comparison

In order to evaluate the candidate methods, linear data sets, consisting of 1000 training points and 2000 test points, with 5 input dimension and 1 output were generated at random. The outputs were calculated from random linear coefficients, and gaussian noise was added. Then, the input data was augmented with 5 additional constant dimensions and rotated and stretched to have random variances in all dimensions. For some test cases, 5 more input dimensions



(a)



(b)

Figure 1. CPR projection under two different weighting schemes

with random noise was added afterwards to explore the effect of irrelevant inputs on the regression performance. Empirically, we determined $p = 4$ as a good choice for CPR.

The simulations considered the following permutations:

1. Low noise [1] (r^2=0.99) and High noise ($r^2 = 0.9$) in output data.
2. With and without irrelevant (non-constant) input dimensions.

Each algorithm was run 100 times on random data sets of each of the 4 combinations of test conditions. Results were

[1] Noise is parametrised by the coefficient of determination (r^2). We add noise scaled to the output variance, i.e. $\sigma_{noise} = c.\sigma_y$, where $c = \sqrt{\frac{1}{r^2} - 1}$. The best normalized mean squared error (nMSE) achievable by a learning system under this noise level is $1 - r^2$.

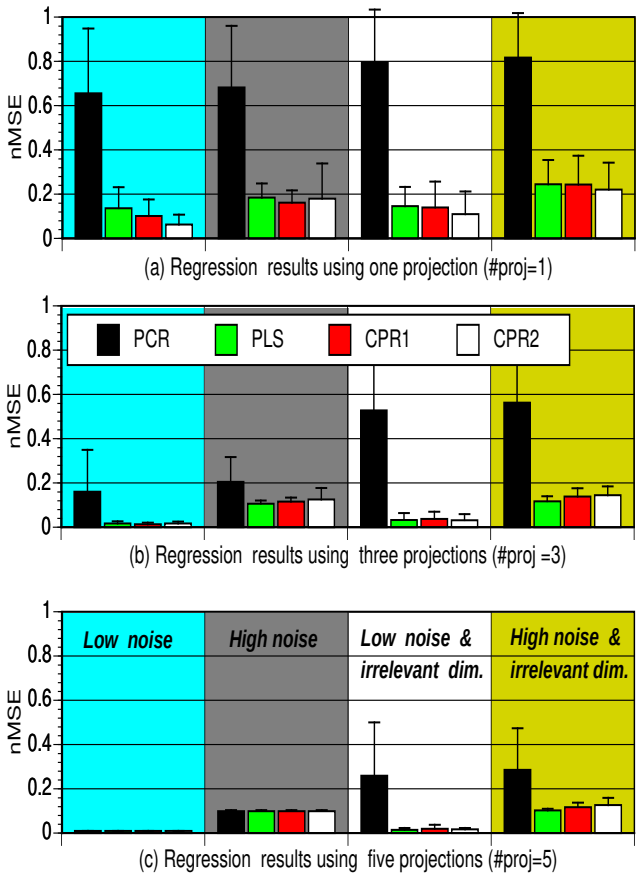


Figure 2. Simulation Results for PCR/PLS/CPR1/CPR2. The subplots show results for projection dimensions 1, 3 and 5. Each of the subplots have four additional conditions made of permutations of : (i) low and high noise (ii) With and without irrelevant(non constant) inputs.

compiled such that the number of projection dimensions k employed by the methods varied from one to five. Fig.2 show the summary results.

It can be seen that on average the PLS and CPR methods outperform the PCR methods by a large margin, even more in the case when irrelevant inputs were included. This can be attributed to the fact that PCR's projections solely rely on the input distributions. In cases where irrelevant inputs have high variance, PCR will thus choose inappropriate projection directions. For low noise cases ($r^2 = 0.99$), CPR performs marginally better than PLS, especially during the first projections. For high noise cases ($r^2 = 0.9$), PLS seems be slightly better. Amongst the CPR candidates, CPR2 seems to have a slight advantage over CPR1 in low noise cases, while the advantage is flipped with larger noise. Summarizing, it can be said that CPR and PLS both perform very well. In contrast to PCR, they accomplish excellent regression results with relatively few projections since their choice of projections does not just try to span the input distribution but rather the gradient of the data.

3. Locally Weighted Projection Regression

Going from linear regression to nonlinear regression can be accomplished by localizing the linear regressions (Vijayakumar & Schaal, 1998; Atkeson, Moore & Schaal, 1997). The key concept here is to approximate nonlinear functions by means of piecewise linear models. Of course, in addition to learning the local linear regression, we must also determine the locality of the particular linear model from the data.

3.1 The LWPR network

In this section, we briefly outline the schematic layout of the LWPR learning mechanism. Fig. 3.1 shows the associated local units and the inputs which feed into it. Here, a weighting kernel (determining the locality) is defined that computes a weight $w_{k,i}$ for each data point $(\mathbf{x}_i, y_i)$ according to the distance from the center $\mathbf{c}_k$ of the kernel in each local unit. For a gaussian kernel, $w_{k,i}$ becomes

$$w_{k,i} = exp(-\frac{1}{2}(\mathbf{x}_i - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x}_i - \mathbf{c}_k)), \qquad (1)$$

where $\mathbf{D}_k$ corresponds to a distance metric that determines the size and shape of region of validity of the linear model. Here we assume that there are K local linear models combining to make the prediction. Given an input vector $\mathbf{x}$, each linear model calculates a prediction y_k. The total output of the network is the weighted mean of all linear models:

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k y_k}{\sum_{k=1}^{K} w_k}, \qquad (2)$$

as shown in Fig. 3.1. The parameters that need to be learned includes the dimensionality reducing transformation (or projection directions) $u_{i,k}$, the local regression parameter $\beta_{i,k}$ and the distance metric D_k for each local module.

3.2 Learning the projection directions and local regression

Previous work (Schaal & Atkeson, 1997) computed the outputs of each linear model y_k by traditional recursive least squares regression over all the input variables. Learning in such a system, however, required more than $O(n^2)$ (where n is the number of input dimensions) computations which became infeasible for more than 10 dimensional input spaces. However, using the PLS/CPR framework, we are able to reduce the computational burden in each local linear model by applying a sequence of one-dimensional regressions along selected projections u_r in input space (note

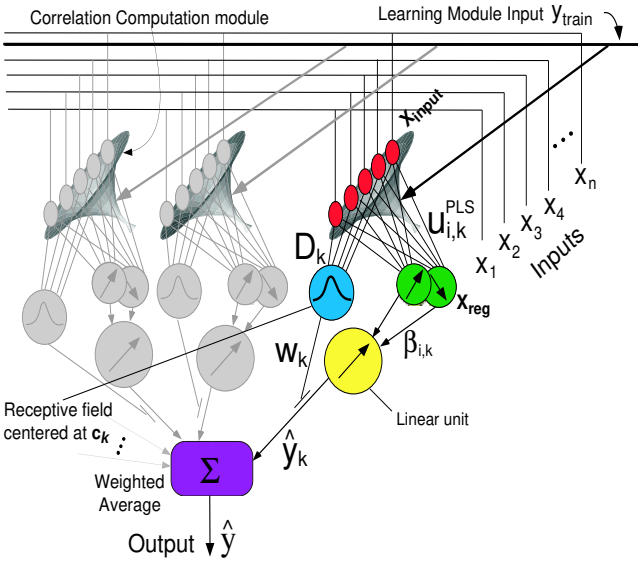


Figure 3. Information processing unit of LWPR

that we drop the index k from now on unless it is necessary to distinguish explicitly between different linear models) as shown in Table 1. The important ingredient of PLS is to choose projections according to the correlation of the input data with the output data. The Locally Weighted Projection Regression (LWPR) algorithm, shown in Table 2, uses an incremental locally weighted version of PLS to determine the linear model parameters.

In Table 2, $\lambda \in [0, 1]$ is a forgetting factor that determines how much older data in the regression parameters will be forgotten, as used in the recursive system identification techniques (Ljung & Soderstrom, 1986). The variables SS, SR, and SZ are memory terms that enable us to do the univariate regression in step (f) in a recursive least squares fashion, i.e., a fast Newton-like method. Step (g) regresses the projection from the current projected data s and the current input data $\mathbf{z}$. This step guarantees that the next projection of the input data for the next univariate regression will result in a u_{i+1} that is orthogonal to u_i. Thus, for $r = N$, the entire input space would be spanned by the projections u_i and the regression results would be identical to that of a traditional linear regression. Once again, we emphasize the important properties of the local projection scheme. First, if all the input variables are statistically independent, PLS will find the optimal projection direction u_i in a single iteration - here, the optimal projection direction corresponds to the gradient of the assumed locally linear function to be approximated. Second, choosing the projection direction from correlating the input and the output data in Step (a) automatically excludes irrelevant input dimensions, i.e., inputs that do not contribute to the output. And third, there is no danger of numerical problems in PLS due to redundant input dimensions as the univariate regressions will never be singular.

Incremental PLS Pseudocode

Given: A training point $(\mathbf{x}, y)$.

Update the means of input and output:

$$\mathbf{x}_0^{n+1} = \frac{\lambda W^n \mathbf{x}_0^n + w\mathbf{x}}{W^{n+1}}$$
$$\beta_0^{n+1} = \frac{\lambda W^n \beta_0^n + wy}{W^{n+1}}$$

where $W^{n+1} = \lambda W^n + w$ and
$\mathbf{x}_0^0 = \mathbf{0}$, $\mathbf{u}_i^0 = \mathbf{0}$, $\beta_0^0 = 0$, $W^0 = 0$

Update the local model:

1. Initialize: $\mathbf{z} = \mathbf{x}$, $res_1 = y - \beta_0^{n+1}$
2. For $i = 1 : r$
   (a) $\mathbf{u}_i^{n+1} = \lambda \mathbf{u}_i^n + w\mathbf{z}\, res_i$
   (b) $s = \mathbf{z}^T \mathbf{u}_i^{n+1}$
   (c) $SS_i^{n+1} = \lambda SS_i^n + w\, s^2$
   (d) $SR_i^{n+1} = \lambda SR_i^n + w\, s\, res_i$
   (e) $SZ_i^{n+1} = \lambda SZ_i^n + w\mathbf{z}s$
   (f) $\beta_i^{n+1} = SR_i^{n+1}/SS_i^{n+1}$
   (g) $\mathbf{p}_i^{n+1} = SZ_i^{n+1}/SS_i^{n+1}$
   (h) $\mathbf{z} = \mathbf{z} - s\mathbf{p}_i^{n+1}$
   (i) $res_{i+1} = res_i - s\beta_i^{n+1}$
   (j) $MSE_i^{n+1} = \lambda MSE_i^n + w\, res_{i+1}^2$

Predicting with novel data:

Initialize: $y = \beta_0$, $\mathbf{z} = \mathbf{x} - \mathbf{x}_0$
For i=1:k

1. $s = \mathbf{u}_i^T \mathbf{z}$
2. $y = y + \beta_i s$
3. $\mathbf{z} = \mathbf{z} - s\mathbf{p}_i^n$

Table 2. PLS Pseudocode

3.3 Learning the locality

So far, we have described the process of finding projection directions and based on this, the local linear regression in each local area. The validity of this local model and hence, the size and shape of the receptive field is determined by the distance metric $\mathbf{D}$. It is possible to optimize the distance metric $\mathbf{D}$ individually for each receptive field by using an incremental gradient descent based on stochastic leave-one-out cross validation criterion. The update rule can be writ-

LWPR outline

- Initialize the LWPR with no receptive field (RF);
- **For** every new training sample (x,y):
  - **For** k=1 to RF:
    - calculate the activation from eq.(1)
    - update according to psuedocode of incremental PLS & Distance Metric update
  - **end**
  - **If** no linear model was activated by more than w_{gen};
    - create a new RF with $r = 2$, $c = x$, $D = D_{def}$
  - **end**
- **end**

Table 3. LWPR Outline

ten as :

$$\mathbf{D} = M^T M, \text{ where } \mathbf{M} \text{ is upper triangular} \quad (3)$$

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\delta J}{\delta \mathbf{M}} \quad (4)$$

where the cost function to be minimized is:

$$J = \frac{1}{W} \sum_{i=1}^{M} \sum_{k=1}^{r} \frac{w_i res_{k+1,i}^2}{(1 - w_i \frac{s_{k,i}^2}{s_k^T \mathbf{W} s_k})^2} + \gamma \sum_{i,j=1}^{N} D_{ij}^2. \quad (5)$$

The above update rules can be embedded in an incremental learning system that automatically allocates new locally linear models as needed. An outline of the algorithm is shown in Table 3.

In this pseudo-code algorithm, w_{gen} is a threshold that determines when to create a new receptive field, and D_{def} is the initial (usually diagonal) distance metric in eq.(1). The initial number of projections is set to $r = 2$. The algorithm has a simple mechanism of determining whether r should be increased by recursively keeping track of the mean- squared error (MSE) as a function of the number of projections included in a local model, i.e., Step (j) in the incremental PLS psuedocode. If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, i.e.,

$$\frac{MSE_{i+1}}{MSE_i} > \phi, \quad (6)$$

where $\phi \in [0, 1]$, the algorithm will stop adding new projections to the local model. For a diagonal distance metric $\mathbf{D}$ and under the assumption that the r remains small, the computational complexity of the update of all parameters of LWPR is linear in the number of input dimensions.

3.4 Empirical Evaluations

We implemented LWPR similar to the development in (Vijayakumar & Schaal, 1998). In each local model, the projection regressions are performed by (locally weighted) PLS, and the distance metric $\mathbf{D}$ is learned by stochastic incremental cross validation (Schaal & Atkeson, 1998); all learning methods employed second order learning techniques. As a first test, we ran LWPR on 500 noisy training data drawn from the two dimensional function

$$y = \max\{\exp(-10x_1^2), \exp(-50x_2^2, 1.25\exp(-5(x_1^2 + x_2^2)))\} + \mathrm{N}(0, 0.01)$$

shown in Fig.4(a). This kind of function with a spatial mixture of strong non-linearities and significant linear regions is an excellent test of the learning and generalization capability. Models with low complexity find it hard to capture the non-linearities while it is easy to overfit with more complex models, especially in linear regions. A second test added 8 constant dimensions to the inputs and rotated this new input space by a random 10-dimensional rotation matrix. A third test added another 10 input dimensions to the inputs of the second test, each having $N(0, 0.05^2)$ Gaussian noise, thus obtaining a 20-dimensional input space. The learning results with these data sets are illustrated in Fig.4(c). In all three cases, LWPR reduced the normalized mean squared error (thick lines) on a noiseless test set rapidly in 10-20 epochs of training to less than $nMSE = 0.05$, and it converged to the excellent function approximation result of $nMSE = 0.01$ after 100,000 data presentations. Fig.4(b) illustrates the reconstruction of the original function from the 20-dimensional test – an almost perfect approximation. The rising thin lines in Fig.4(c) show the number of local models that LWPR allocated during learning. The very thin lines at the bottom of the graph indicate the average number of projections that the local models allocated: the average remained at the initialization value of two projections, as is appropriate for this originally two dimensional data set.

Previous work (Schaal & Atkeson, 1998) has quantitatively compared the performance of RFWR, a predecessor of LWPR, to baseline techniques like sigmoidal neural networks as well as to more advanced techniques like the mixture of experts systems of Jordan & Jacobs (Jacobs, 1991; Jordan & Jacobs, 1994) and the Cascade correlation algorithms (Fahlman & Lebiere, 1990). These results show that RFWR is very competitive, outperforms most of these techniques and is especially robust to non-static input distributions and interference during learning. One must note that stripping the LWPR algorithm of it's dimensionality reduction preprocessing essentially gives us the RFWR algorithm.

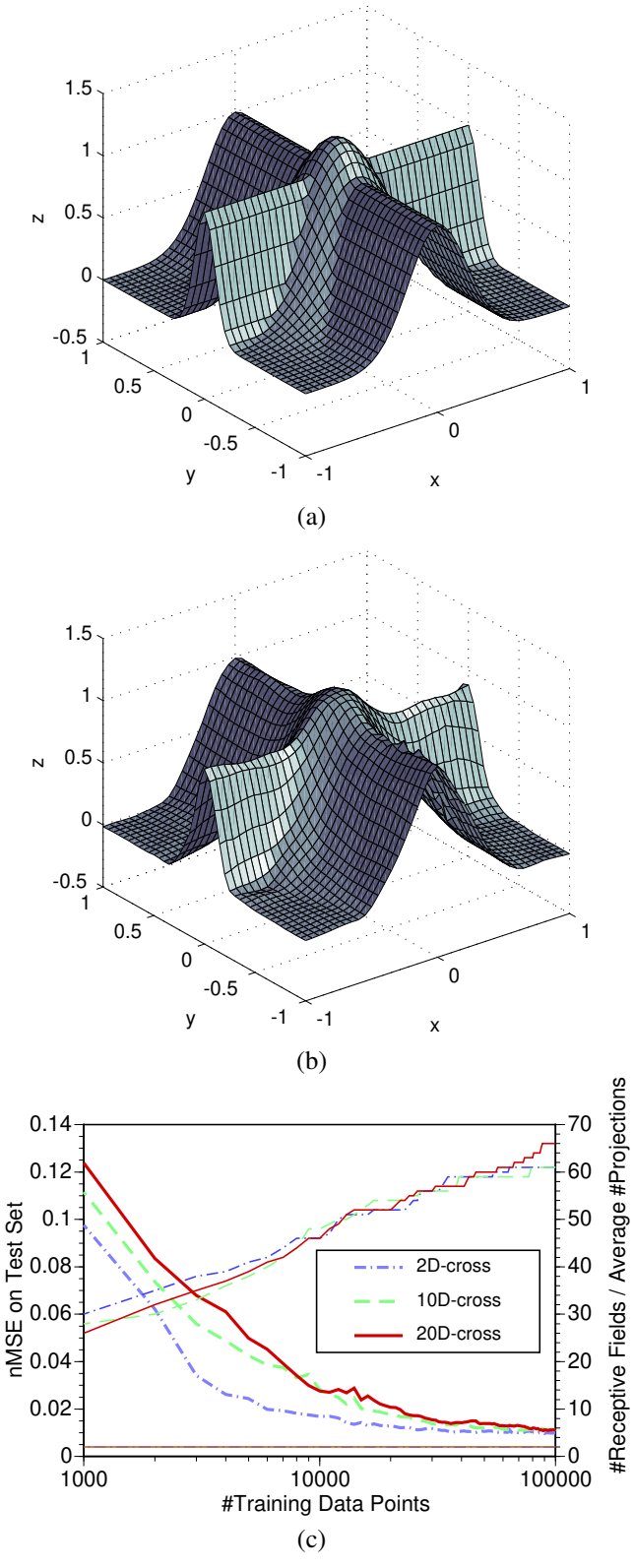


Figure 4. (a) Target and (b) learned nonlinear cross function.(c) Learning curves for 2-D, 10-D and 20-D data

In the second evaluation, we approximated the inverse dynamics model of a 7-degree-of-freedom anthropomorphic robot arm (see Fig.5(a)) from a data set consisting of 45,000 data points, collected at 100Hz from the actual robot performing various rhythmic and discrete movement tasks (this corresponds to 7.5 minutes of data collection). The inverse dynamics model of the robot is strongly nonlinear due to a vast amount of superpositions of sine and cosine functions in robot dynamics. The data consisted of 21 input dimensions: 7 joint positions, velocities, and accelerations. The goal of learning was to approximate the appropriate torque command of one robot motor in response to the input vector. To increase the difficulty of learning, we added 29 irrelevant dimensions to the inputs with $N(0, 0.05^2)$ Gaussian noise. 5,000 data points were excluded from the training data as a test set. Fig.5(b) shows the learning results in comparison to two other state of the art techniques in this field - parameter estimation based on Rigid Body Dynamic models and Levenberg-Marquardt based Backpropogation with sigmoidal neural networks. The parameter estimation technique uses apriori knowledge about the analytical form of the robot dynamics equations and that these equations are linear in the unknown inertial and kinematic parameters of the robot. Linear regression techniques with complex analytical data preprocessing was used to obtain these parameters, thus resulting in a complete analytical model of the robot inverse dynamics. From the very beginning, LWPR outperformed the global parameter estimation technique. Within 250,000 training presentations, LWPR converged to the excellent result of $nMSE = 0.045$. It employed an average of only 3.8 projection dimensions per local model inspite of the input dimensionality of 50. During learning, the number of local models increased by a factor of 10 from about 50 initial models to about 400 models. This increase is due to the adjustment of the distance metric $\mathbf{D}$ in eq.(1), which was initialized to form a very large kernel. Since this large kernel over-smoothes the data, LWPR reduced the kernel size, and in response more kernels needed to be allocated. In comparison, the LM Back-Prop method, which is computationally much more intensive, achieved $nMSE = 0.055$, which is statistically similar. However, as is evident from Fig.5(b), it took much longer to converge to the optimal value compared to LWPR. Once again, the key issue is that none of these compared algorithms are incremental or online. We have not been able to find another incremental, online algorithm in the literature which scales for the input dimensionality and redundancy handled in the tasks here.

4. Discussion

This paper discussed methods of linear projection regression and how to use them in spatially localized nonlinear function approximation for high-dimensional input data

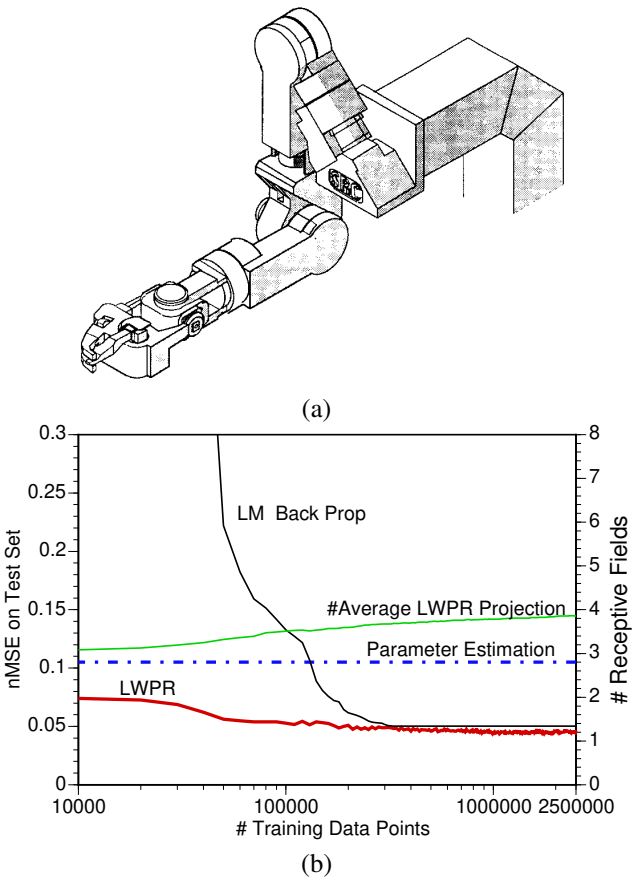


Figure 5. (a) Sketch of the SARCOS dextrous arm (b) Learning curves for 50 dimensional robot dynamics learning

that has redundant and irrelevant components. We derived a family of linear projection regression methods that bridged the gap between principal component regression, a commonly used algorithm with inferior performance, and partial least squares regression, a less known algorithm with, however, superior performance. Each of these algorithms can be used at the core of nonparametric function approximation with spatially localized weighting kernels. As an example, we demonstrated how one nonlinear function approximator derived from this family leads to excellent function approximation results in up to 50 dimensional data sets. Besides showing very fast and robust learning performance due to second order learning methods based on stochastic cross validation, the new algorithm excels by its low computational complexity: updating one projection direction has linear computational cost in the number of inputs, and since the algorithm accomplishes good approximation results with only 3-4 projections irrespective of the number of input dimensions, the overall computational complexity remains linear in the inputs. To our knowledge, this is the first spatially localized incremental learning system that can efficiently work in high dimensional spaces.

References

[1] Atkeson,C., Moore,A. & Schaal,S. Locally weighted learning. *Artificial Intelligence Review*, 11(4):76–113, 1997.

[2] Fahlman,S.E. & Lebiere.C. The cascade correlation learning architecture. *Advances in Neural Information Processing Systems 2*, 1990.

[3] Frank,I.E. &Friedman,J.H. A statistical view of some chemometric regression tools. *Technometrics*, 35(2):109–135, 1993.

[4] Friedman,J.H. & Stutzle,W. Projection pursuit regression. *Journal of the American Stat. Assoc.*, 76:817-823(1981).

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

[6] Hastie,T.J. & Tibshirani,R.J. *Generalized Additive Models*, Chapman & Hall, 1990.

[7] Jacobs,R.A., Jordan,M.I., Nowlan, S.J. & Hinton,G.E. Adaptive mixture of local experts, *Neural Computation*,3:79–87, 1991.

[8] Jordan,M.I. & Jacobs,R.A. Heirarchical mixture of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[9] Massy,W.F. Principal component regression in exploratory statistical research. *Journal of Amer. Stat. Assoc.*,60:234–246, 1965.

[10] Sanger,T.D. Optimal unsupervised learning in a single layer liner feedforward neural network, *Neural Networks*, 2:459–473, 1989.

[11] Scott,D.W. *Multivariate Density Estimation*, Wiley-NY, 1992.

[12] Schaal,S. & Atkeson,C.G. Receptive Field Weighted Regression, *Technical Report TR-H-209, ATR Human Information Processing Labs.*, Kyoto, Japan.

[13] Schaal,S. & Atkeson,C.G. Constructive Incremental Learning from only Local Information. *Neural Computation*, 10(8):2047–2084, 1998.

[14] Schaal,S., Vijayakumar,S. & Atkeson,C.G. Local Dimensionality Reduction. *Advances in Neural Information Processing Systems 10*,633–639, 1998.

[15] Vijayakumar,S. & Schaal,S. Local Adaptive Subspace Regression. *Neural Processing Letters*, 7(3):139–149, 1998.

[16] Wold,H. Soft modeling by latent variables: the nonlinear iterative partial least squares approach. *Perspectives in Probability and Statistics*, 1975.

[17] Ljung,L. & Soderstrom,T. *Theory and practice of recursive identification*, Cambridge MIT Press,1986.

Real Time Learning in Humanoids: A challenge for scalability of Online Algorithms

Sethu Vijayakumar and Stefan Schaal

Computer Science & Neuroscience and Kawato Dynamic Brain Project
University of Southern California, Los Angeles, CA 90089-2520, USA
{*sethu,sschaal*}*@usc.edu*

Abstract. While recent research in neural networks and statistical learning has focused mostly on learning from finite data sets without stringent constraints on computational efficiency, there is an increasing number of learning problems that require real-time performance from an essentially infinite stream of incrementally arriving data. This paper demonstrates how even high-dimensional learning problems of this kind can successfully be dealt with by techniques from nonparametric regression and locally weighted learning. As an example, we describe the application of one of the most advanced of such algorithms, Locally Weighted Projection Regression (LWPR), to the on-line learning of the inverse dynamics model of an actual seven degree-of-freedom anthropomorphic robot arm. LWPR's linear computational complexity in the number of input dimensions, its inherent mechanisms of local dimensionality reduction, and its sound learning rule based on incremental stochastic leave-one-out cross validation allows – to our knowledge for the first time – implementing inverse dynamics learning for such a complex robot with real-time performance. In our sample task, the robot acquires the local inverse dynamics model needed to trace a figure-8 in only 60 seconds of training.

1 Introduction

An increasing number of learning problems involves real-time modeling of complex high-dimensional processes. Typical examples include the on-line modeling of dynamic processes observed by visual surveillance, user modeling for advanced computer interfaces and game playing, and the learning of value functions, policies, and internal models for learning control. Among the characteristics of these learning problems are high dimensional input spaces with potentially redundant and irrelevant dimensions, nonstationary input and output distributions, essentially infinite training data sets with no representative validation sets, and the need for continual learning. Most of these learning tasks fall into the domain of regression problems.

Interestingly, this class of problems has so far not been conquered by the new developments in statistical learning. Bayesian inference [3] is usually computationally too expensive for real-time application as it requires representation of the complete joint probablity densities of the data. The framework of structural risk minimization[9], the most advanced in form of Support Vector Machines, excels in classification and finite batch learning problems, but has yet to show compelling performance in regression and incremental learning. In contrast, techniques from nonparametric regression, in particular the methods of locally weighted learning [2], have recently advanced to meet all the requirements of real-time learning in high-dimensional spaces. In this paper, we will describe how one of the most highly developed algorithms amongst them, Locally Weighted Projection Regression (LWPR), accomplishes learning of a highly nonlinear model for robot control– the inverse dynamics model of a seven degree-of-freedom (DOF) anthropomorophic robot. In the following sections, we will first explain the learning task, then spell out the LWPR algorithm, and finally illustrate learning results from real-time learning on the actual robot. To the best of our knowledge, this is the first time that real-time learning of such a complex model has been accomplished in robot control.

2 Inverse Dynamics Learning

A common strategy in robotic and biological motor control is to convert kinematic trajectory plans into motor commands by means of an inverse dynamics model. The inverse dynamics takes the desired positions, velocities, and accelerations of all DOFs of the robot and outputs the appropriate motor commands. For our robot, a seven DOF anthropomorphic robot arm, the inverse dynamics model receives 21 inputs and outputs 7 torque commands. If derived analytically using a rigid body dynamics assumption [1], the most compact recursive formulation of the inverse dynamics of our robot results in about 15 pages of compact C-code, filled with nested sine and cosine terms.

Table 1. Pseudocode of PLS projection regression

1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\mathbf{y}_{res} = \mathbf{y}$
2. **For** $r = 1$ **to** R (# projections)
   (a) $\mathbf{u}_r = \mathbf{X}_{res}^T \mathbf{y}_{res}$; $\beta_r = \mathbf{s}_r^T \mathbf{y}_{res} / (\mathbf{s}_r^T \mathbf{s}_r)$ where $\mathbf{s}_r = \mathbf{X}_{res} \mathbf{u}_r$.
   (b) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{s}_r \beta_r$; $\mathbf{X}_{res} = \mathbf{X}_{res} - \mathbf{s}_r \mathbf{p}_r{}^T$ where $\mathbf{p}_r = \mathbf{X}_{res}^T \mathbf{s}_r / (\mathbf{s}_r^T \mathbf{s}_r)$.

For on-line learning, motor commands need to be generated from the model at 480Hz in our implementation. Updating the learning system can take place at a lower rate but should remain above 10Hz to capture suffcient data in fast movements.

Learning regression problems in a 21-dimensional input space is a daunting problem from the view of the bias-variance trade-off. In learning control, training data is generated by the learning system itself, and it is impossible to assess a priori what structural complexity that data is going to have. Fortunately, actual movement systems do not fill the data space in a completely random way. Instead, when viewed locally, data distributions are low dimensional, e.g., about 4-6 dimensional for the inverse dynamics [8] of our robot instead of the global 21 input dimensions. This property will be a key element in our approach to learning such models.

3 Locally Weighted Projection Regression

The core concept of our learning approach is to approximate nonlinear functions by means of piecewise linear models [2]. The learning system automatically determines the appropriate number of local models, the parameters of the hyperplane in each model, and also the region of validity, called receptive field (RF), of each of the model, usually formalized as a Gaussian kernel:

$$w_k = exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x} - \mathbf{c}_k)), \tag{1}$$

Given a query point $\mathbf{x}$, each linear model calculates a prediction y_k. The total output of the learning system is the weighted mean of all K linear models:

$$\hat{y} = \sum_{k=1}^{K} w_k y_k / \sum_{k=1}^{K} w_k,$$

also illustrated in Fig.1. Learning in the system involves determining the linear regression parameter β_k and the distance metric $\mathbf{D}_k$. The center $\mathbf{c}_k$ of the RF remains fixed. Local models are created as and when needed as described in Section 3.3.

3.1 Local Dimensionality Reduction

Despite its appealing simplicity, the "piecewise linear modeling" approach becomes numerically brittle and computationally too expensive in high dimensional problems. Given the empirical observation that high dimensional data is often locally low dimensional, it is possible to develop a very efficient approach to exploit this property. Instead of using ordinary linear regression to fit the local hyperplanes, we suggest to employ Partial Least Squares (PLS) [11, 4]. PLS recursively computes orthogonal projections of the input data and performs single variable regressions along these projections on the residuals of the previous iteration step. Table 1 illustrates PLS in pseudocode for a global linear model where the input data is in the rows of the matrix $\mathbf{X}$, and the corresponding one dimensional output data is in the vector $\mathbf{y}$. The key ingredient in PLS is to use the direction of maximal correlation between the residual error and the input data as the projection direction at every regression step. Additionally, PLS regresses the inputs of the previous step against the projected inputs $\mathbf{s}_r$ in order to ensure the orthogonality of all the projections $\mathbf{u}_r$ (Step 2b). Actually, this additional regression could be avoided by replacing $\mathbf{p}_r$ with $\mathbf{u}_r$, similar to techniques used in principal component analysis [5]. However, using this regression step leads to better performance of the algorithm. This effect is due to the fact that PLS chooses the most effective projections if the input data has a spherical distribution: with only one projection, PLS will find the direction of the gradient and achieve optimal regression results. The regression step in 2b modifies the input data $\mathbf{X}_{res}$ such that each resulting data vectors have coefficients of minimal magnitude and, hence, push the distribution of $\mathbf{X}_{res}$ to become more spherical.

An incremental locally weighted version of the PLS algorithm[10] is derived in Table 2. Here, $\lambda \in [0, 1]$ denotes a forgetting factor that determines how quickly older data will be forgotten in the various PLS parameters, similar to

Table 2. Incremental locally weighted PLS for one RF

Initialization:
$\mathbf{x}_0^0 = \mathbf{0}$, $\mathbf{u}^0 = \mathbf{0}$, $\beta_0^0 = 0$, $W^0 = 0$
Given: A training point $(\mathbf{x}, y)$

$$w = exp(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^T \mathbf{D}(\mathbf{x}-\mathbf{c}))$$

Update the means :

$$W^{n+1} = \lambda W^n + w$$
$$\mathbf{x}_0^{n+1} = \frac{\lambda W^n \mathbf{x}_0^n + w\mathbf{x}}{W^{n+1}}$$
$$\beta_0^{n+1} = \frac{\lambda W^n \beta_0^n + wy}{W^{n+1}}$$

Update the local model:
Initialize:
$\mathbf{z} = \mathbf{x} - \mathbf{x}_0^{n+1}$, $res_1 = y - \beta_0^{n+1}$
For $r = 1 : R$ (# projections)
1. $\mathbf{u}_r^{n+1} = \lambda \mathbf{u}_r^n + w\mathbf{z}\, res_r$
2. $s_r = \mathbf{z}^T \mathbf{u}_r^{n+1} / ({\mathbf{u}_r^{n+1}}^T \mathbf{u}_r^{n+1})$
3. $SS_r^{n+1} = \lambda SS_r^n + w\, s_r^2$
4. $SR_r^{n+1} = \lambda SR_r^n + w\, s_r\, res_r$
5. $\beta_r^{n+1} = SR_r^{n+1} / SS_r^{n+1}$
6. $res_{r+1} = res_r - s_r \beta_r^{n+1}$
7. $MSE_r^{n+1} = \lambda MSE_r^n + w\, res_{r+1}^2$
8. $SZ_r^{n+1} = \lambda SZ_r^n + w\mathbf{z}s_r$
9. $\mathbf{p}_r^{n+1} = SZ_r^{n+1} / SS_r^{n+1}$
10. $\mathbf{z} = \mathbf{z} - s_r \mathbf{p}_r^{n+1}$

Predicting with novel data ($\mathbf{x}_q$): Initialize: $y = \beta_0$, $\mathbf{z} = \mathbf{x}_q - \mathbf{x}_0$
Repeat for r=1:R
- $y = y + \beta_r s_r$ where $s_r = \mathbf{u}_r^T \mathbf{z}$
- $\mathbf{z} = \mathbf{z} - s_r \mathbf{p}_r^n$

the recursive system identification techniques [12]. The variables SS_r, SR_r and SZ_r are memory terms that enable us to do the univariate regression in step (7) in a recursive least squares fashion, i.e., a fast Newton-like method.

Since PLS selects the univariate projections very efficiently, it is even possible to run locally weighted PLS with only *one* projection direction (denoted as LWPR-1). The optimal projection is in the direction of the local gradient of the function to be approximated. If the locally weighted input data forms a spherical distribution in a local model, the single PLS projection will suffice to find the optimal direction. Otherwise, the distance metric (and hence, weighting of the data) will need to be adjusted to make the local distribution more spherical. The learning rule of the distance metric can accomplish this adjustment, as explained below. It should be noted that Steps 2(h-j) in Table 2 become unnecessary for the uni-projection case.

3.2 Learning the Distance Metric

The distance metric $\mathbf{D}_k$ and hence, the locality of the receptive fields, can be learned for each local model individually by stochastic gradient descent in a leave-one-out cross validation cost function. Note that this update does *not* require competitive learning – only a completely local learning rule is needed, and leave-one-out cross validation can be performed *without* keeping data in memory [7]. The update rule can be written as :

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \text{ where } \mathbf{D} = \mathbf{M}^T \mathbf{M} \text{ (for positive definiteness)} \tag{2}$$

and the cost function to be minimized is:

$$J = \frac{1}{\sum_{i=1}^M w_i} \sum_{i=1}^M \sum_{r=1}^R \frac{w_i res_{r+1,i}^2}{(1 - w_i \frac{s_{r,i}^2}{\mathbf{s}_r^T \mathbf{W} \mathbf{s}_r})^2} + \frac{\gamma}{N} \sum_{i,j=1}^N D_{ij}^2 = \sum_{r=1}^R (\sum_{i=1}^M J_{1,r}) + J_2 \,. \tag{3}$$

where M denotes the number of training data, and N the number of input dimensions. A stochastic version of the gradient $\frac{\partial J}{\partial \mathbf{M}}$ can be derived from the cost function by keeping track of several "memory terms" as shown in Table 3.

3.3 The Complete LWPR Algorithm

All the ingredients above can be combined in an incremental learning scheme that automatically allocates new locally linear models as needed. The final learning network is illustrated in Fig. 1 and an outline of the algorithm is shown below.

Table 3. Derivatives for distance metric update

$$\frac{\partial J}{\partial \mathbf{M}} \approx \sum_{r=1}^{R}(\sum_{i=1}^{M}\frac{\partial J_{1,r}}{\partial w})\frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W^{n+1}}\frac{\partial J_2}{\partial \mathbf{M}} \text{ (stochastic update)}$$

$$\frac{\partial w}{\partial M_{kl}} = -\frac{1}{2}w(\mathbf{x}-\mathbf{c})^T\frac{\partial \mathbf{D}}{\partial M_{kl}}(\mathbf{x}-\mathbf{c});\;\; \frac{\partial J_2}{\partial M_{kl}} = 2\frac{\gamma}{N}\sum_{i=1,j=1}^{N} D_{ij}\frac{\partial D_{ij}}{\partial M_{kl}}$$

$$\frac{\partial D_{ij}}{\partial M_{kl}} = M_{kj}\delta_{il} + M_{ki}\delta_{jl};\; \text{where } \delta_{ij}=1 \text{ if } i=j \text{ else } \delta_{ij}=0.$$

Compute the following for each projection direction r:

$$\sum_{i=1}^{M}\frac{\partial J_{1,r}}{\partial w} = \frac{e_{cv,r}^2}{W^{n+1}} - 2\frac{(P_r^{n+1}s_r e_r)}{W^{n+1}}H_r^n - 2\frac{(P_r^{n+1}s_r)^2}{W^{n+1}}R_r^n - \frac{E_r^{n+1}}{(W^{n+1})^2}$$
$$+[\mathbf{T}_r^{n+1} - 2R_r^{n+1}P_r^{n+1}\mathbf{C}_r^{n+1}]\frac{(\mathbf{I}-\mathbf{u}_r\mathbf{u}_r^T/(\mathbf{u}_r^T\mathbf{u}_r))\mathbf{z}\; res_r}{W^{n+1}\sqrt{\mathbf{u}_r^T\mathbf{u}_r}}$$

$$\mathbf{C}_r^{n+1} = \lambda\mathbf{C}_r^n + ws_r\mathbf{z}^T,\;\; e_r = res_{r+1},\;\; e_{cv,r} = \frac{e_r}{1-wP_r^{n+1}s_r^2},\;\; P_r^{n+1} = \frac{1}{SS_r^{n+1}}$$

$$H_r^{n+1} = \lambda H_r^n + \frac{w\; e_{cv,r}s_r}{(1-w\; h_r)};\;\; R_r^{n+1} = \lambda R_r^n + \frac{w^2 s_r^2 e_{cv,r}^2}{(1-w\; h_r)} \text{ where } h_r = P_r^{n+1}s_r^2$$

$$E_r^{n+1} = \lambda E_r^n + we_{cv,r}^2;\;\; \mathbf{T}_r^{n+1} = \lambda\mathbf{T}_r^n + \frac{w(2we_{cv,r}^2 s_r P_r^{n+1} - e_{cv,r}\beta_r^{n+1})}{(1-w\; h_r)}\mathbf{z}^T$$

LWPR outline

- Initialize the LWPR with no receptive field (RF);
- **For** every new training sample (x,y):
  - **For** k=1 to K:
    - calculate the activation from eq.(1)
    - update projections & regression (Table 2) and Distance Metric (Table 3)
  - **If** no RF was activated by more than w_{gen};
    - create a new RF with $r=2$, $\mathbf{c}=\mathbf{x}$, $\mathbf{D}=\mathbf{D}_{def}$

In this pseudo-code, w_{gen} is a threshold that determines when to create a new receptive field, and $\mathbf{D}_{def}$ is the initial (usually diagonal) distance metric in eq.(1). The initial number of projections is set to $r=2$. The algorithm has a simple mechanism of determining whether r should be increased by recursively keeping track of the mean-squared error (MSE) as a function of the number of projections included in a local model, i.e., step (g) in the incremental PLS pseudocode. If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, i.e., $\frac{MSE_{i+1}}{MSE_i} > \phi$, where $\phi \in [0,1]$, the algorithm will stop adding new projections locally. For a diagonal distance metric $\mathbf{D}$ and under the assumption that the number of projections R remains small, the computational complexity of the update of all parameters of LWPR is linear in the number of input dimensions. The LWPR-1 variant on the other hand uses just one projection direction.

4 Real-Time Learning of Inverse Dynamics

4.1 Performance Comparison on a Static Data Set

Before demonstrating the applicability of LWPR in real-time, a comparison with alternative learning methods will serve to demonstrate the complexity of the learning task. We collected 50,000 data points from various movement patterns from a 7 DOF anthropomorphic robot (Fig.2a) at 50Hz sampling frequency. 10 percent of this data was excluded as a test set. The training data was approximated by 4 different methods: i) parameter estimation based on an analytical rigid body dynamics model [1], ii) Support Vector Regression[6], iii) LWPR-1, and iv) full LWPR. It

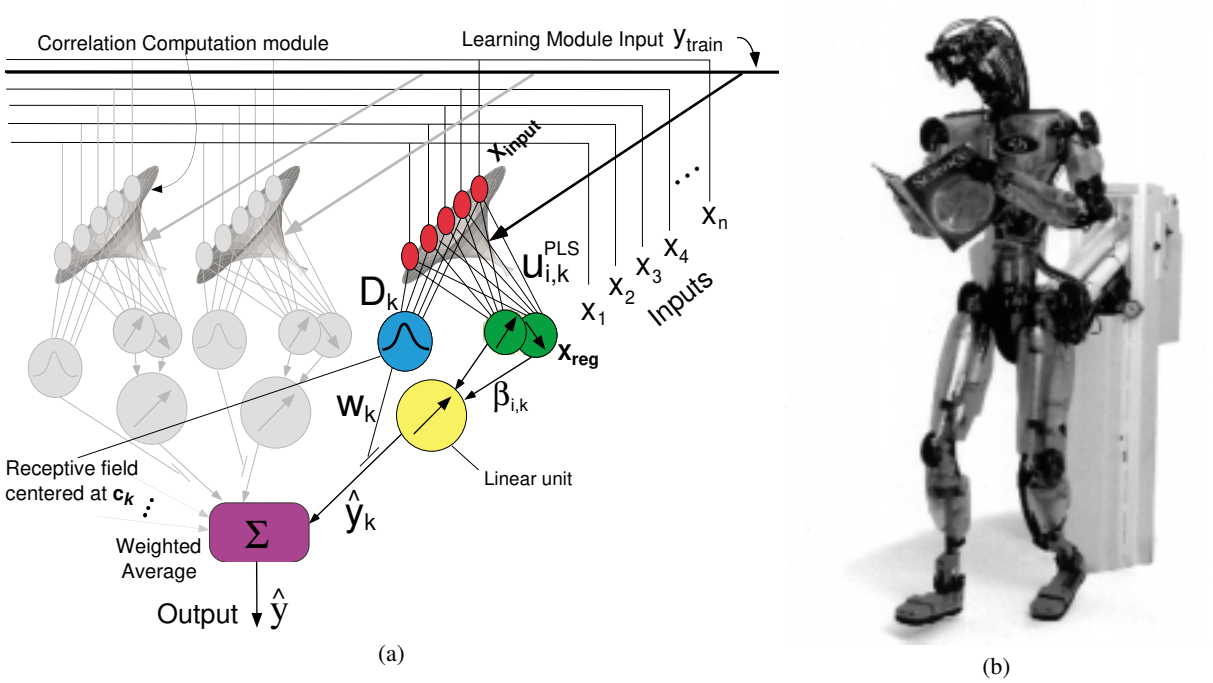


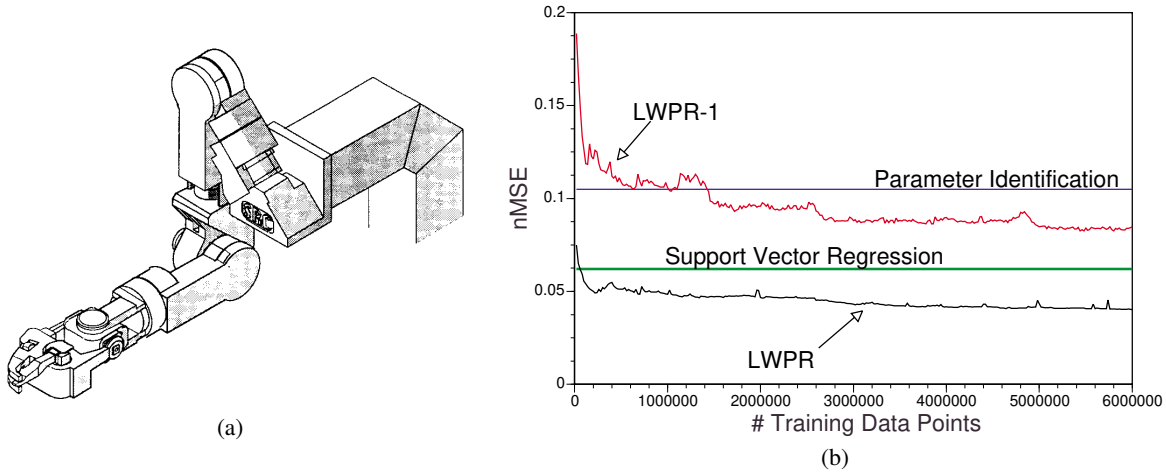
Fig. 1. (a) Information processing unit of LWPR (b) Humanoid Robot used for learning in our laboratory

should be noted that neither i) nor ii) are incremental methods. Using a parametric model as suggested in i) and just approximating its open parameters from data results in a global model of the inverse dynamics and is theroretically the most powerful method. However, given that our robot is actuated hydraulically and rather lightweight and compliant, we know that the rigid body dynamics assumption is not fully justified. In all our evaluations, the inverse dynamics model of each DOF was learned separately, i.e., all models had a univariate output and 21 inputs. LWPR employed a diagonal distance metric.

Fig.2b illustrates the function approximation results for the shoulder motor command graphed over the number of training iterations (one iteration corresponds to the update from one data point). Surprisingly, rigid body parameter estimation achieved the worst results. LWPR-1 outperformed parameter estimation, but fell behind SVM regression. Full LWPR performed the best. The results for all other DOFs were analogous. For the final result, LWPR employed 260 local models, using an average of 3.2 local projections. LWPR-1 did not perform better because we used a diagonal distance metric. The abilities of a diagonal distance metric to "carve out" a locally spherical distribution are too limited to accomplish better results – a full distance metric can remedy this problem, but would make the learning updates quadratic in the number of inputs. These results demonstrate that LWPR is a competitive function approximation technque.

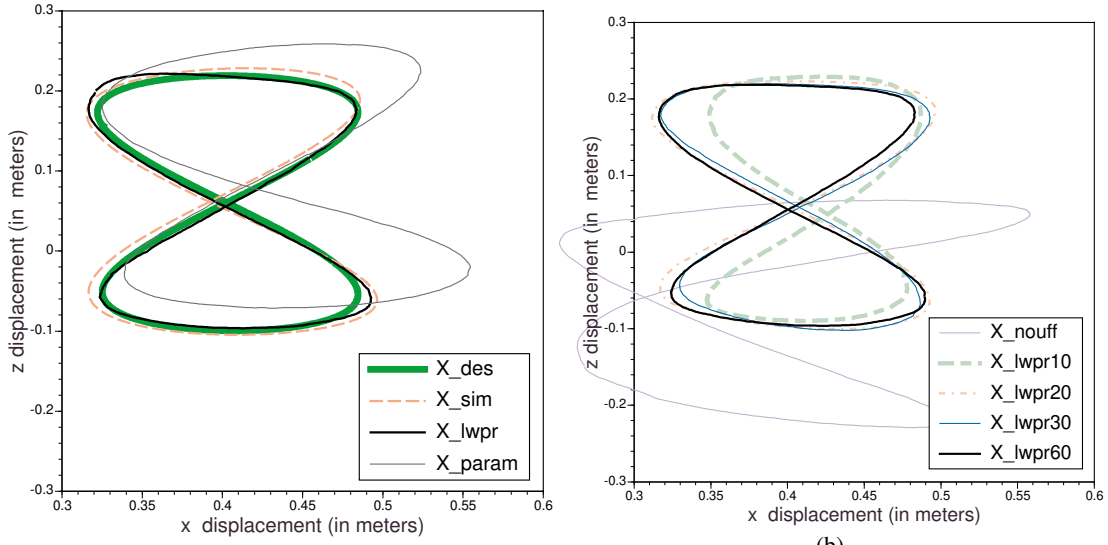
4.2 On-line Learning

We implemented full LWPR on our robotic setup. Out of the four parallel processors of the system, one 366Mhz PowerPC processor was completely devoted to lookup and learning with LWPR. Each DOF had its own LWPR learning system, resulting in 7 parallel learning modules. In order to accelerate lookup and training times, we added a special data structure to LWPR. Each local model maintained a list of all other local models that overlapped sufficiently with it. Sufficient overlap between two local model i and j can be determined from the centers and distance metrics. The point $\mathbf{x}$ in input space that is the closest to both centers in the sense of a Mahalanobis distance is $\mathbf{x} = (\mathbf{D}_i + \mathbf{D}_j)^{-1}(\mathbf{D}_i\mathbf{c}_i + \mathbf{D}_j\mathbf{c}_j)$. Inserting this point into eq.(1) of one of the local models gives the activation w due to this point. The two local models are listed as sufficiently overlapping if $w >= w_{gen}$ (cf. LWPR outline). For diagonal distance metrics, the overlap computation is linear in the number of inputs. Whenever a new data point is added to LWPR, one neighborhood relation is checked for the maximally activated RF. An appropriate counter for each local model ensures that overlap with all other local models is checked exhaustively. Given this "nearest neighbor" data structure and the fact that a movement system generates temporally highly correlated data, lookup and learning can be confined to only few RFs. For every lookup (update), the identification number of the maximally



(a) (b)

Fig. 2. (a) SARCOS dexterous arm (b) Comparison of nMSE traces for different learning schemes



(a) (b)

Fig. 3. (a) Robot end effector motion traces under different control schemes (b) Progress of online learning with LWPR control

activated RF is returned. The next lookup (update) will only consider the neighbors of this RF. It can be proved that this method performs as good as an exhaustive lookup (update) strategy that excludes RFs that are activated below a certain threshold w_{cutoff}.

The LWPR models were trained on-line while the robot performed a pseudo randomly drifting figure-8 pattern in front of its body. Lookup proceeded at 480Hz, while updating the learning model was achieved at about 70Hz. At certain intervals, learning was stopped and the robot attempted to draw a planar figure-8 at 2Hz frequency for the entire pattern. The quality of these drawing patterns is illustrated in Fig.3a,b. In Fig.3a, X_{des} denotes the desired figure-8 pattern, X_{sim} illustrates the figure-8 performed by our robot simulator that uses a perfect inverse dynamics model (but not necessarily a perfect tracking and numerical integration algorithm), X_{param} is the performance of the estimated rigid body dynamics model, and X_{lwpr} shows the results of LWPR. While the rigid body model has the worst performance, LWPR obtained the best results, even better than the simulator. Fig.3b illustrates the speed of LWPR learning. The X_{nouff} trace demonstrates the figure-8 patterns performed without any inverse dynamics model, just using a low gain PD controller. The other traces show how rapidly LWPR learned the figure-8 pattern during training: they denote performance after 10, 20, 30, and 60 *seconds* of training. After 60 seconds, the figure-8 is hardly distinguishable from the desired trace.

5 Conclusions

This paper illustrated an application of Locally Weighted Projection Regression to a complex robot learning task. The *O(n)* update complexity of LWPR, together with its statistically sound dimensionality reduction and learning rules allowed a reliable and successful real-time implementation of the algorithm on an actual anthropomorphic robot. From the viewpoint of learning control, these results demark the first time that complex internal models can be learned autonomously in real-time on sophisticated robotic devices. In an even more challenging application, we will implement LWPR for inverse dynamics learning on a 30 DOF full-body humanoid robot (Fig.1b) in the near future.

References

1. An,C.H., Atkeson,C. & Hollerbach, J. (1988) *Model Based Control of a Robot Manipulator* MIT Press.
2. Atkeson, C., Moore, A. & Schaal, S. (1997). Locally weighted learning. *Artif. Intel. Rev.*, *11*, 76–113.
3. Bishop, C. (1995) *Neural Networks for Pattern Recognition*. Oxford Press.
4. Frank, I. E. & Friedman, J. H. (1993). A stat. view of some chemo. reg. tools. *Technometrics*, *35*, 109–135.
5. Sanger, T. D. (1989). Optimal unsupervised learning in a single layer liner feedforward neural network. *Neural Networks*, *2*, 459–473.
6. Saunders,C., Stitson, M.O., Weston,J., Bottou,L., Schoelkopf,B., Smola,A. (1998) Support Vector Machine - Reference Manual. *TR CSD-TR-98-03*, Dept. of Computer Science, Royal Holloway, Univ. of London.
7. Schaal, S. & Atkeson, C. G. (1998). Const. inc. learning from only local info. *Neural Comp*, *10*, 2047–2084.
8. Schaal, S., Vijayakumar, S. & Atkeson, C. G. (1998). Local dimensionality reduction. *NIPS 10*,633–639.
9. Vapnik, V. (1995) *The Nature of Statistical Learning Theory*. Springer, New York.
10. Vijayakumar, S. & Schaal,S. (2000). Locally Weighted Projection Regression : An *O(n)* algorithm for incremental real time learning in high dimensional space. Proc. ICML 2000 (in press).
11. Wold, H. (1975). Soft modeling by latent variables: the nonlinear iterative partial least squares approach. *Perspectives in Probability and Statistics*.
12. Ljung, L. & Soderstrom, T. (1986) *Theory and practice of recursive identification*. Cambridge MIT Press.

Schaal, S., & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation*, **10,** 8, 2047-2084.

Constructive Incremental Learning From Only Local Information

Stefan Schaal[*]
sschaal@usc.edu
http://www-slab.usc.edu/sschaal

Christopher G. Atkeson[‡]
cga@cc.gatech.edu
http://www.cc.gatech.edu/fac/Chris.Atkeson

[*]Department of Computer Science, Unversity of Southern California, Los Angeles, CA 90089-2520
[*]Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto
[‡]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332
[*‡]ATR Human Information Processing Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto

Abstract

We introduce a constructive, incremental learning system for regression problems that models data by means of spatially localized linear models. In contrast to other approaches, the size and shape of the receptive field of each locally linear model as well as the parameters of the locally linear model itself are learned independently, i.e., without the need for competition or any other kind of communication. Independent learning is accomplished by incrementally minimizing a weighted local cross validation error. As a result, we obtain a learning system that can allocate resources as needed while dealing with the bias-variance dilemma in a principled way. The spatial localization of the linear models increases robustness towards negative interference. Our learning system can be interpreted as a nonparametric adaptive bandwidth smoother, as a mixture of experts where the experts are trained in isolation, and as a learning system which profits from combining independent expert knowledge on the same problem. This paper illustrates the potential learning capabilities of purely local learning and offers an interesting and powerful approach to learning with receptive fields.

1 Introduction

Learning with spatially localized basis functions has become a popular paradigm in machine learning and neurobiological modeling. In the context of radial basis function networks (Moody & Darken, 1988; Poggio & Girosi, 1990), it was demonstrated that such local learning offers an alternative to learning with global basis functions, such as sigmoidal neural networks, and that its theoretical foundation can be solidly grounded in approximation theory (Powell, 1987). In neurophysiological studies, the concept of localized information processing in the form of receptive fields has been known since at least the work of Mountcastle (1957) and Hubel and Wiesel (1959). Since then, a wealth of experimental evidence has been accumulated which suggests that information processing based on local receptive fields is a ubiquitous organizational principle in neurobiology that offers interesting computational opportunities (e.g., Lee, Rohrer, & Sparks, 1988; Georgopoulos, 1991; Field, 1994; Olshausen & Field, 1996; Daugman & Downing, 1995).

In this paper we explore the computational power of local, receptive field-based incremental learning with the goal of approximating unknown functional relationships between incoming streams of input and output data. By incremental learning we do not just mean that the parameters of the learning system are updated incrementally. We want to address a learning scenario in which limited memory is available such that after a new data point is incorporated in the learning system it is discarded and cannot be reused, in which input and output distributions of the data are unknown, and in which these distribution may change over time. This situation resembles the learning of sensory and sensorimotor transformations in biology, and it also applies to a variety of artificial domains, ranging from autonomous robotic systems to process control.

Given these constraints on incremental learning, two major problems need to be addressed. The first is how to allocate the appropriate number of resources, e.g., receptive fields, in order to deal with the tradeoff between overfitting and oversmoothing, called the bias-variance dilemma (e.g., Geman, Bienenstock, & Doursat, 1992). The second problem of incremental learning comes from negative interference, the forgetting of useful knowledge while learning from new data. Methods to prevent negative interference require either validation data sets, memorizing of all training data, or strong prior knowledge about the learning problem. However, none of these alternatives are available in the setting we have described as we want to avoid storing data and do not have much knowledge about the structure of the learning task.

In order to address the problems of incremental learning, we will resort to techniques from nonparametric statistics (e.g., Scott, 1992; Hastie & Tibshirani, 1990). Nearest neighbor algorithms for pattern recognition and Parzen windows for density estimation are among the best known methods out of this field (e.g., Duda & Hart, 1973). It is interesting to note that many nonparametric methods are essentially receptive field-based: predictions are made using data from a restricted local neighborhood around the query point. The size of the neighborhood can be irregular, as typically is the case in nearest neighbor approaches, or it can be a symmetric smooth weighting function as in Parzen windows. Receptive fields in nonparametric regression are mostly built on the fly and are discarded right after the prediction—a paradigm that has been termed lazy learning (Aha, 1997). Necessarily, such nonparametric methods need to store training data. Another characteristic is that predictions are usually based on a single receptive field. This property inspired the field of nonparametric regression to pursue more complex models in a receptive field, for instance, low order polynomials (e.g., Cleveland, 1979; Cleveland & Loader, 1995). In contrast, many neural network algorithms, such as radial basis function systems, focused on combining the activation strengths of many receptive fields to optimize predictions.

In this paper we will demonstrate how a nonparametric regression approach can be used to build a receptive field-based learning system for incremental function approximation without the need to store the training data and without discarding receptive fields after using them. A locally linear model will be fitted incrementally within each receptive field such that local function approximation is accomplished in the spirit of a

Taylor series expansion. A new property of this learning approach is that each receptive field is trained *independently* of all other receptive fields, thereby adjusting the parameters of its locally linear model, the size and shape of its receptive field, as well as the bias on the relevance on its individual input dimensions. New receptive fields are allocated as needed. The resulting algorithm, Receptive Field Weighted Regression (RFWR), achieves robust incremental learning. It also has some interesting relations to previously suggested learning methods. It can be interpreted as a mixture of experts system (Jacobs, Jordan, Nowlan, & Hinton, 1991; Jordan & Jacobs, 1994) where the experts are trained in isolation. It can also be interpreted as system where a set of experts is trained independently on the same problem, and which profits from combining these experts for making predictions (e.g., Perrone & Cooper, 1993). And finally, RFWR can be interpreted as a nonparametric memory-based learner (Atkeson, Moore, & Schaal, 1997) which only stores data that are surprising.

In the next section, we will give some motivation for our approach to incremental learning. Section 3 describes the details of our nonparametric incremental learning system and outlines some of its statistical characteristics. Section 4 discusses a variety of empirical evaluations. Section 5 outlines related work, while Section 6 concludes this paper.

2 Incremental Learning

2.1 Statistical Assumptions

The assumed statistical model of our problems is the standard regression model:

$$\mathbf{y} = f(\mathbf{x}) + \varepsilon \tag{1}$$

where $\mathbf{x} \in \Re^n$ denotes the n-dimensional vector of input variables, $\mathbf{y} \in \Re^m$ the m-dimensional vector of output variables, and $f(\cdot)$ a deterministic vector valued function mapping the input $\mathbf{x}$ to the output $\mathbf{y}$. The additive random noise ε is assumed to be independently distributed, $E\{\varepsilon_i \varepsilon_j\} = 0$ for $i \neq j$, and mean zero, $E\{\varepsilon \mid \mathbf{x}\} = 0$, but otherwise of unknown distribution ($E\{\cdot\}$ denotes the expectation operator). The input data is distributed according to the density $p(\mathbf{x})$.

2.2 Localizing Interference

Interference in learning is a natural side-effect of the ability to generalize, i.e., to interpolate or extrapolate an output for an unseen input from previously learned data. Generalization is accomplished by allowing changes to the parameters of the learning system to have non-local effects. If these effects reduce the overall correctness of predictions to a larger extent than they improve them, interference is called negative or even catastrophic. Incremental learning is particularly endangered by negative interference because there is no direct way to balance the amount of positive interference (i.e., gen-

eralization) with the amount of negative interference: any parameter update is usually greedy; its only concern is with the reduction of the error of the current piece of training data. To see the statistical causes of interference, consider using the mean squared error criterion J to select a model $\hat{f}(\cdot)$ to approximate the true function $f(\cdot)$:

$$J = E\left\{\left\|\mathbf{y} - \hat{f}(\mathbf{x})\right\|^2\right\} = \int_{-\infty}^{+\infty} \left\|\mathbf{y} - \hat{f}(\mathbf{x})\right\|^2 p(\mathbf{x},\mathbf{y})\, d\mathbf{x}\, d\mathbf{y} = \int_{-\infty}^{+\infty} \left\|\mathbf{y} - \hat{f}(\mathbf{x})\right\|^2 p(\mathbf{y}|\mathbf{x}) p(\mathbf{x})\, d\mathbf{x}\, d\mathbf{y} \tag{2}$$

This equation states that, in general, the approximation result for $\hat{f}(\cdot)$ depends on *both* the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$ and the input distribution $p(\mathbf{x})$ of the data (Fan & Gijbels, 1996). Only for an infinite amount of training data, $\hat{f}(\cdot)$ will asymptotically depend solely on $p(\mathbf{y} \mid \mathbf{x})$ (Papoulis, 1991):

$$\hat{f}(\mathbf{x}) = E\{\mathbf{y}|\mathbf{x}\} = \int_{-\infty}^{+\infty} \mathbf{y}\, p(\mathbf{y}|\mathbf{x})\, d\mathbf{y} \tag{3}$$

Thus, for a finite amount of data, a stable model $\hat{f}(\cdot)$ can only be obtained if neither of these distributions changes during learning.

These considerations point towards the two major causes for negative interference. If $p(\mathbf{y} \mid \mathbf{x})$ changes, i.e., the functional relationship between $\mathbf{x}$ and $\mathbf{y}$ is non-stationary, the parameters in a learning system may have to change. Analogously, if the data for learning are not sampled from a fixed input distribution $p(\mathbf{x})$, the parameters of the learning system may also change. It is particularly a change of the input distribution $p(\mathbf{x})$ which is likely to happen in incremental learning. Imagine a robot learning an inverse dynamics model of its arm, a model which maps joint positions, joint velocities, and joint accelerations to corresponding joint torques. Whenever the robot moves, it receives valid data about this functional relationship. However, since the robot is fulfilling different tasks at different times, the sampled data will come from quite different input distributions—for example, consider the difference between movements for cooking and movements for playing tennis.

One of the interesting properties of learning with localized receptive fields lies in their potential robustness towards interference. If learning is spatially localized, i.e., training data at one location have negligible effect on the parameters of distant receptive fields, interference will be spatially localized as well. The example of Figure 1 gives an illustration of this effect. Using a synthetic data set suggested by Fan and Gijbels (1995), we trained a 3-layer sigmoidal feedforward neural network (6 hidden units, using backpropagation with momentum) on 130 noisy data points uniformly distributed in $x \in [-2.0, 0.5]$ ("•" in Figure 1). The excellent function fit obtained is shown by the "predicted y" trace in Figure 1a. Then we continued training the network on 70 new data points ("+" in Figure 1) drawn from the same function but with a changed input distribution $x \in [0.5, 2.0]$. The network learned to accommodate these new data points, but in doing so, it also significantly changed its predictions for the previously learned data, although this data is largely separate from the new training data. This effect is due

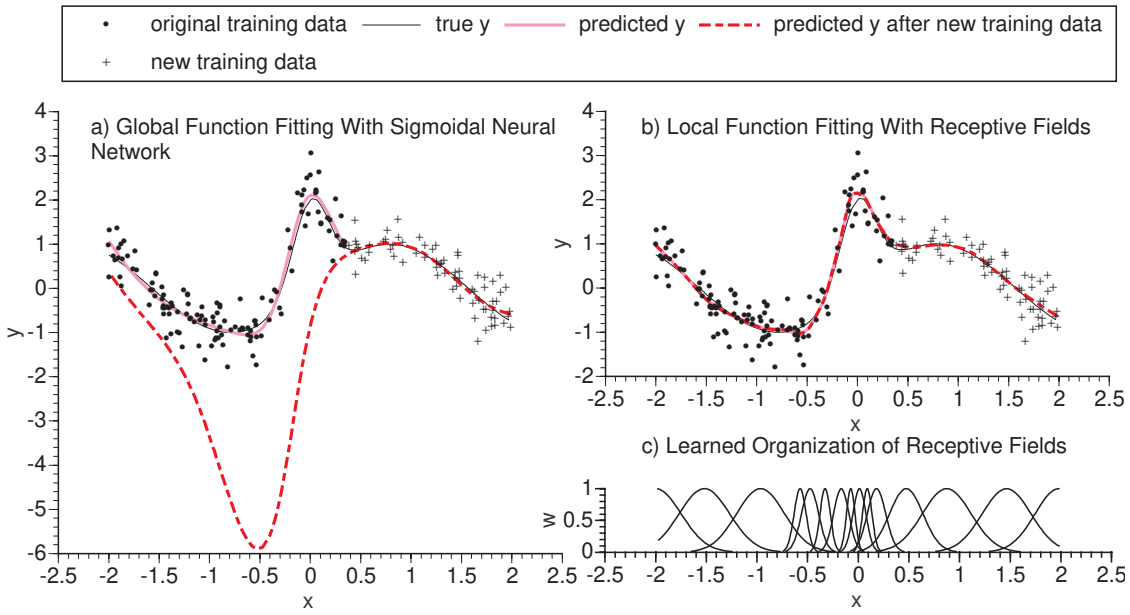


Figure 1: a) Results of function approximation of the function $y=\sin(2x)+2\exp(-16x^2)+N(0,0.16)$ with a sigmoidal neural network, b) results of function approximation by a local receptive field-based algorithm, fitting locally linear models in each receptive field (note that the data traces "true y", "predicted y", and "predicted y after new training data" largely coincide), c) the organization of the (Gaussian) receptive fields of b) after training.

to the non-local nature of sigmoidal basis functions, and is prone to lead to catastrophic interference, as shown in Figure 1a.

We repeated the same experiment with our receptive field-based learning system, RFWR, which generates locally linear models in each receptive field and blends them for predictions (Figure 1b,c). On the original training data, RFWR achieves comparable results to that of the sigmoidal neural network. After training on the new data, however, no interference is apparent. The original fit in the left part of the graph was not visibly altered, in contrast to the neural network. Robustness towards negative interference is accomplished by localizing interference—the best we can do since interference cannot be eliminated for finite data samples.

2.3 Avoiding The Problem of Resource Allocation

Due to the bias-variance tradeoff (Geman et al., 1992), learning algorithms have to include a model selection phase in order to find an appropriate compromise between oversmoothing and overfitting. Usually, this is accomplished by setting certain meta parameters, for instance, the number of hidden units in a neural network, according to some model selection criterion, e.g., cross validation (Stone, 1974). A question frequently asked in model selection (e.g., Bishop, 1996) thus becomes: "How many free parameters should be allocated in order to achieve (in expectation) a good bias-variance

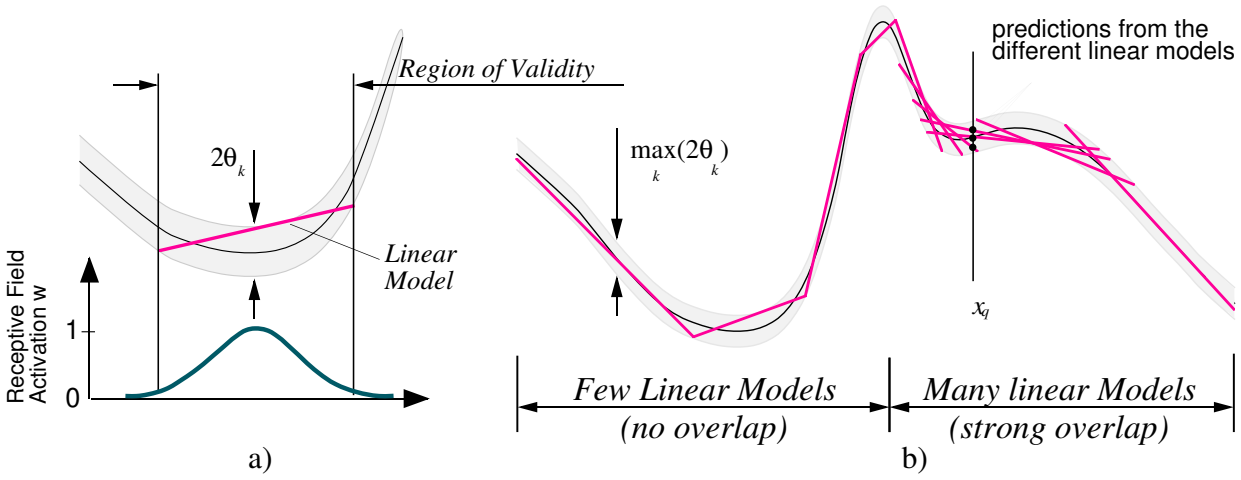


Figure 2: a) Region of validity of a linear model and its approximation bias θ_k; b) function approximation with piecewise linear models.

tradeoff?" However, another approach can be pursued: "Given a *fixed* number of free parameters, how should a given data set be *spatially limited* in order to achieve (in expectation) a good bias-variance tradeoff for the remaining data?"—instead of adapting the complexity of the learning system, one can also adapt the complexity of the region the data is drawn from. For general nonlinear function approximators, it is unclear how to answer this question. For spatially localized function fitting, however, this question translates into: "How should the extent of a receptive field be changed in order to make its associated parametric model fit the data appropriately?" Such an approach transforms the *global* bias-variance problem into a *local* one.

The advantage of having each receptive field deal with the bias-variance tradeoff individually lies in avoiding the resource allocation problem. In the spirit of a Taylor series expansion, let us assume that we know how to adjust the region of validity, i.e., the size and shape of a receptive field, of each locally linear model such that its approximation error at the center—its bias θ_k—is determined by an optimal bias-variance tradeoff (Figure 2a). In order to approximate the entire nonlinear function, we have to cover the input space with sufficiently many locally linear models such that every data point is handled by at least one of them. Importantly, it does *not* matter whether we allocate too many local models: if we restrict extrapolation of the linear models to the θ_k bound (which actually corresponds to a minimal activation strength of a receptive field), an average of the outputs of all k linear models at a query point x_q cannot have a larger error than $\max(\theta_k)$ as illustrated in Figure 2b. Indeed, allocating too many local models has actually a positive effect: due to averaging, more overlapping linear models will tend to improve function estimates in the spirit of and with the same limitations as in ensemble methods (Perrone & Cooper, 1993).

Although deriving an optimal local bias-variance tradeoff remains hard (Friedman, 1984; Fan & Gijbels, 1996), the local nature of the problem allows new ways to find at

least satisfying and computationally affordable solutions. Section 3 will demonstrate how a stochastic approximation of local leave-one-out cross validation in conjunction with a regularization approach can be used to realize the local bias-variance tradeoff, and to even approximately control the expected bias θ_k in each local model.

2.4 Summary

Given the discussion of the last two sections, a promising route to robust incremental learning seems to be a local receptive field-based system that can also adjust the extent of its receptive fields. However, care must be taken how one goes about accomplishing this goal. Learning methods based on competitive learning do usually not achieve the properties described in the previous section. In competitive learning, the size of a receptive field results from a global competition process of all local models to account for the training data. Therefore, changing the number of local models causes a change of the extent of *all* receptive fields such that the number of local models becomes a critical choice for the bias-variance tradeoff—exactly what we would wish to avoid. The next section will explain how an alternative approach based on nonparametric statistics offers a route to achieve our goals without resorting to competitive learning.

3 Receptive Field Weighted Regression

RFWR constructs a system of receptive fields for incremental function approximation. A prediction $\hat{\mathbf{y}}$ for a query point $\mathbf{x}$ is built from the normalized weighted sum of the individual predictions $\hat{\mathbf{y}}_k$ of all receptive fields:

$$\hat{\mathbf{y}} = \frac{\sum_{k=1}^{K} w_k \hat{\mathbf{y}}_k}{\sum_{k=1}^{K} w_k} \tag{4}$$

The weights w_k correspond to the activation strengths of the corresponding receptive fields. They are determined from the size and shape of each receptive field, characterized by a kernel function. A variety of possible kernels have been suggested (e.g., Atkeson et al., 1997). For analytical convenience, we use a Gaussian kernel:

$$w_k = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x}-\mathbf{c}_k)\right), \quad \text{where} \quad \mathbf{D}_k = \mathbf{M}_k^T \mathbf{M}_k \tag{5}$$

which parameterizes the receptive field by its location in input space, $\mathbf{c}_k \in \Re^n$, and a positive definite distance metric $\mathbf{D}_k$, determining the size and shape of the receptive field. For algorithmic reasons, it is convenient to generate $\mathbf{D}_k$ from an upper triangular matrix $\mathbf{M}_k$ in order to ensures that $\mathbf{D}_k$ is positive definite.

Within each receptive field, a simple parametric function models the relationship between input and output data. Local polynomials of low order have found widespread

use in nonparametric statistics (Nadaraya, 1964; Watson, 1964; Wahba & Wold, 1975; Cleveland, 1979; Cleveland & Devlin, 1988). We will focus on locally linear models, as they accomplish a favorable compromise between computational complexity and quality of result (Hastie & Loader, 1993):

$$\hat{\mathbf{y}}_k = \left(\mathbf{x} - \mathbf{c}_k\right)^T \mathbf{b}_\mathbf{k} + b_{0,k} = \tilde{\mathbf{x}}^T \beta_k, \qquad \tilde{\mathbf{x}} = \left(\left(\mathbf{x} - \mathbf{c}_k\right)^T, 1\right)^T \tag{6}$$

where β_k denotes the parameters of the locally linear model and $\tilde{\mathbf{x}}$ a compact form of the center-subtracted, augmented input vector to simplify the notation.

To clarify the elements and parameters of RFWR, Figure 3 gives a network-like illustration for a single output system. The inputs are routed to all receptive fields, each of which consists of a linear and a Gaussian unit. The learning algorithm of RFWR determines the parameters $\mathbf{c}_k$, $\mathbf{M}_k$, and β_k for each receptive field *independently*, i.e., without any information about the other receptive fields, in contrast to competitive learning. RFWR adds and prunes receptive fields as needed, such that the number of receptive fields, K, will automatically adjust to the learning problem at hand. A one dimensional example of function fitting with RFWR was already shown in Figure 1b,c. It should be noted that the size of each receptive field adapted according to the local curvature of the function, that there is a certain amount of overlap between the receptive fields, and that the center locations have not been chosen with respect to any explicit optimization criterion.

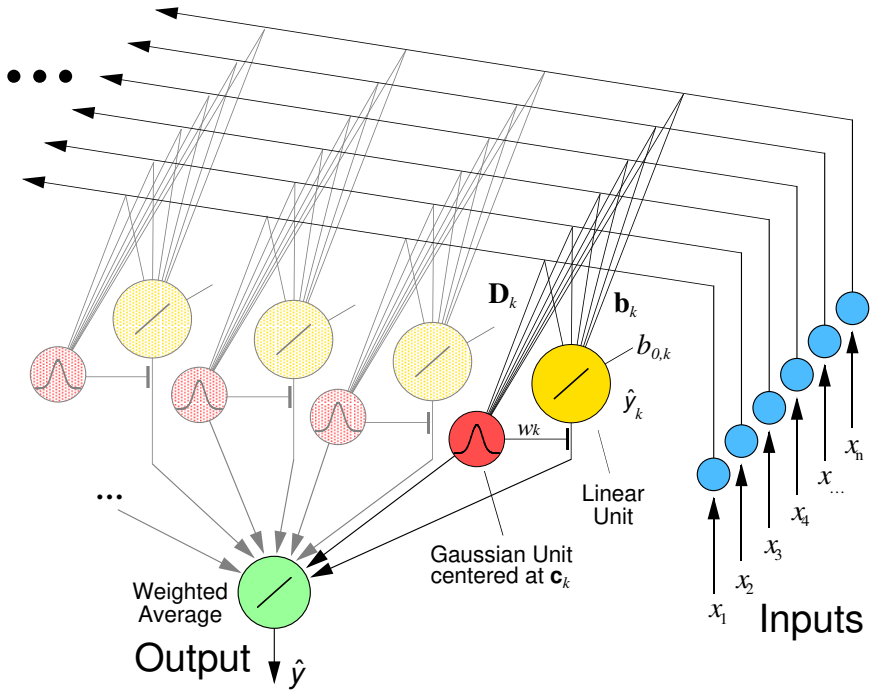


Figure 3: A network illustration of Receptive Field Weighted Regression

3.1 Learning With RFWR

Three ingredients of the algorithm need to be discussed: the update of the linear model parameters β_k, the decomposed distance metric $\mathbf{M}_k$, and when and where to add and prune receptive fields. The centers $\mathbf{c}_k$ are not changed after they are allocated. For the sake of clarity, we will drop the subscript k whenever we deal with one receptive field at a time from now on since each receptive field is updated in the same way.

3.1.1 Learning the Linear Model

Learning of β is straightforward since the problem is linear in β. It will be useful to leave the incremental learning framework for a moment and think in terms of a batch update. If we summarize the input part of all p training data points in the rows of the matrix $\mathbf{X} = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_p)^T$, the corresponding output part in the rows of the matrix $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_p)^T$, and the corresponding weights in the diagonal matrix $\mathbf{W} = \mathrm{diag}\left(\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_\mathrm{p}\right)$, the parameter vector β can be calculated from a weighted regression:

$$\beta = \left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{W} \mathbf{Y} = \mathbf{P} \mathbf{X}^T \mathbf{W} \mathbf{Y} \tag{7}$$

This kind of locally weighted regression has found extensive application in non-parametric statistics (Cleveland, 1979; Cleveland & Loader, 1995), in time series prediction (Farmer & Sidorowich, 1987, 1988), and in regression learning problems (Atkeson, 1989; Moore, 1991; Schaal & Atkeson, 1994; Atkeson et al., 1997). The result for β in Equation (7) is *exactly* the same when β is calculated by recursive least squares from one sequential sweep through the training data (Ljung & Söderström, 1986). Given a training point $(\mathbf{x}, \mathbf{y})$, the incremental update of β yields:

$$\beta^{n+1} = \beta^n + w \mathbf{P}^{n+1} \tilde{\mathbf{x}} \mathbf{e}_{cv}^T \tag{8}$$

$$\text{where} \quad \mathbf{P}^{n+1} = \frac{1}{\lambda}\left(\mathbf{P}^n - \frac{\mathbf{P}^n \tilde{\mathbf{x}} \tilde{\mathbf{x}}^T \mathbf{P}^n}{\frac{\lambda}{w} + \tilde{\mathbf{x}}^T \mathbf{P}^n \tilde{\mathbf{x}}}\right) \quad \text{and} \quad \mathbf{e}_{cv} = \left(\mathbf{y} - \beta^{n^T} \tilde{\mathbf{x}}\right)$$

This update is employed by RFWR. It is useful to note that recursive least squares corresponds to a Newton training method with guaranteed convergence to the global minimum of, in our case, a weighted squared error criterion (Atkeson et al., 1997). Furthermore, the recursive update avoids an explicit matrix inversion. Differing from the batch update in Equation (7), Equation (8) also includes a forgetting factor λ. Changes to the decomposed distance metric $\mathbf{M}$ during learning (see below) will change the weights w. For this reason, it is necessary to include λ in (8) in order to gradually cancel the contributions from previous data points where $\mathbf{M}$ was not yet learned properly (Ljung & Söderström, 1986).

3.1.2 Learning the Shape and Size of the Receptive Field

Adjusting the shape and size of the receptive field is accomplished by adjusting the decomposed distance metric **M**. At the first glance, one might hope that this can be done by gradient descent in the weighted mean squared error criterion:

$$J = \frac{1}{W}\sum_{i=1}^{p} w_i \left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|^2 \quad \text{where} \quad W = \sum_{i=1}^{p} w_i \tag{9}$$

which is the basis of the solution of locally weighted regression in Equation (7) (Atkeson et al, 1997). Unfortunately, minimizing (9) may result in a quite inappropriate solution. If for each training point one receptive field is centered right on this point, and the corresponding **M** is chosen such that the receptive field is so narrow that it is only activated by this data point, the corresponding linear model can fit this one data point with zero error. The function approximation result would strongly tend towards overfitting. It is this property that has made learning algorithms resort to competitive learning with a fixed number of local receptive fields: the global competitive process will prevent receptive fields from modeling just one data point (assuming there are more data points than receptive fields) (e.g., Moody & Darken, 1988; Jordan & Jacobs, 1994). But allowing for such a global competitive process takes away the property of being a local learner, even if the receptive fields are actually spatially localized.

An alternative way to address this overfitting effect is to use leave-one-out cross validation. The cost function to be minimized changes from Equation (9) to

$$J = \frac{1}{W}\sum_{i=1}^{p} w_i \left\| \mathbf{y}_i - \hat{\mathbf{y}}_{i,-i} \right\|^2 \tag{10}$$

The notation $\hat{\mathbf{y}}_{i,-i}$ denotes that the prediction of the *i*-th data point is calculated from training the learning system with the *i*-th data point excluded from the training set. Thus, it becomes inappropriate for a receptive field to just focus on one training point since the error measure is calculated from data which did not exist in the training set. Leave-one-out cross validation is usually computationally very expensive since a *p*-fold training of the learning system is required, for *p* data points in the training set. Furthermore, for example for a sigmoidal neural network, it might be unclear how to combine the resultant *p* different learned parameters into a single solution. However, for linear regression problems, there is a result rendering these concerns irrelevant. Due to the Sherman-Morrison-Woodbury Theorem (e.g., Belsley, Kuh, & Welsh, 1980), Equation (10) can be re-written as:

$$J = \frac{1}{W}\sum_{i=1}^{p} w_i \left\| \mathbf{y}_i - \hat{\mathbf{y}}_{i,-i} \right\|^2 = \frac{1}{W}\sum_{i=1}^{p} \frac{w_i \left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|^2}{\left(1 - w_i\, \tilde{\mathbf{x}}_i^T \mathbf{P}\, \tilde{\mathbf{x}}_i\right)^2} \tag{11}$$

This equation states that the leave-one-out cross validation error can be obtained without *p*-fold training of the learning system, but instead by an adjustment of the weighted mean squared error with the help of the inverted covariance matrix **P** (cf. (7)). Equation

(11) corresponds to a weighted version of the PRESS residual error in standard linear regression techniques (Myers, 1990). Neglecting for a moment how this cost function can be minimized incrementally, we have obtained a criterion which can be used to adjust $\mathbf{M}$ (Schaal & Atkeson, 1994).

Unfortunately, there is still a point of concern with Equation (11). Minimizing the locally weighted leave-one-out cross validation error results in a consistent learning system, i.e., with an increasing number of training data, the receptive fields will shrink to a very small size. The advantage of this behavior is that function approximation becomes asymptotically unbiased, i.e., consistent, but as a disadvantage, an ever increasing number of receptive fields will be required to represent the approximated function. This property can be avoided by introducing a penalty term in (11):

$$J = \frac{1}{W}\sum_{i=1}^{p}\frac{w_i\left\|\mathbf{y}_i - \hat{\mathbf{y}}_i\right\|^2}{\left(1 - w_i\,\tilde{\mathbf{x}}_i^T\mathbf{P}\tilde{\mathbf{x}}_i\right)^2} + \gamma\sum_{i,j=1}^{n} D_{ij}^2 \tag{12}$$

where the scalar γ determines the strength of the penalty. By penalizing the sum of squared coefficients of the distance metric $\mathbf{D}$, we are essentially penalizing the second derivatives of the function at the site of a receptive field. This is similar to approaches taken in spline fitting (deBoor, 1978; Wahba, 1990) and acts like a low-pass filter: the higher the second derivatives, the more smoothing (and thus bias) will be introduced locally. Another positive effect of the penalty term is that the introduction of bias reduces the variance of the function estimate, a problem usually associated with local function fitting methods (Friedman, 1984). Section 3.3 will outline the properties of (12) in more detail.

What remains is how to minimize (12) incrementally by adjusting $\mathbf{M}$ by gradient descent with learning rate α:

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha\frac{\partial J}{\partial \mathbf{M}} \tag{13}$$

Applying the chain rule, the derivative of (13) can be written as

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{M}} &= \frac{\partial}{\partial \mathbf{M}}\left(\sum_{i=1}^{p}\frac{w_i\left\|\mathbf{y}_i - \hat{\mathbf{y}}_i\right\|^2}{W\left(1 - w_i\,\tilde{\mathbf{x}}_i^T\mathbf{P}\tilde{\mathbf{x}}_i\right)^2} + \gamma\sum_{i,j=1}^{n} D_{ij}^2\right) \\ &= \frac{\partial}{\partial \mathbf{M}}\left(\sum_{i=1}^{p} J_{1,i} + J_2\right) = \sum_{i=1}^{p}\sum_{j=1}^{p}\frac{\partial J_{1,i}}{\partial w_j}\frac{\partial w_j}{\partial \mathbf{M}} + \frac{\partial J_2}{\partial \mathbf{M}}\end{aligned} \tag{14}$$

Without storing data in incremental learning, we cannot use cross validation and, thus, cannot obtain the true gradient in (14). The usual approach to deriving a stochastic gradient would be to drop the two sums in (14). However, this approximate gradient would be quite inaccurate since the first term of (14) would always be positive: shrinking the receptive field reduces the weight of a data point and thus its contribution to the weighted error. It turns out that we are able to derive a much better stochastic approxi-

mation. Given one training point $(\mathbf{x},\mathbf{y})$ and its associated weight w from (5), the derivative for this point can be approximated as:

$$\frac{\partial J}{\partial \mathbf{M}} \approx \sum_{i=1}^{p} \frac{\partial J_{1,i}}{\partial w}\frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W}\frac{\partial J_2}{\partial \mathbf{M}} = \frac{\partial w}{\partial \mathbf{M}}\sum_{i=1}^{p}\frac{\partial J_{1,i}}{\partial w} + \frac{w}{W}\frac{\partial J_2}{\partial \mathbf{M}} \tag{15}$$

Summing (15) over all data points and recalling that W stands for the sum of weights (cf. Equation (9)), Equation (15) can be verified to result in Equation (14). Despite the term $J_{1,i}$, it is now possible to obtain an incremental version of the stochastic derivative in (15) by introducing the "memory traces" W, E, $\mathbf{H}$, and $\mathbf{R}$ (cf. notation in (8)):

$$\begin{aligned} W^{n+1} &= \lambda W^n + w \\ E^{n+1} &= \lambda E^n + w\,\mathbf{e}_{cv}^T\mathbf{e}_{cv} \\ \mathbf{H}^{n+1} &= \lambda\mathbf{H}^n + \frac{w\,\tilde{\mathbf{x}}\,\mathbf{e}_{cv}^T}{1-h}, \quad \text{where} \quad h = w\,\tilde{\mathbf{x}}^T\mathbf{P}^{n+1}\,\tilde{\mathbf{x}} \\ \mathbf{R}^{n+1} &= \lambda\mathbf{R}^n + \frac{w^2\,\mathbf{e}_{cv}^T\mathbf{e}_{cv}\tilde{\mathbf{x}}\,\tilde{\mathbf{x}}^T}{1-h} \end{aligned} \tag{16}$$

The resulting incremental version of the derivative (15) becomes:

$$\frac{\partial J}{\partial \mathbf{M}} \approx \frac{\partial w}{\partial \mathbf{M}}\sum_{i=1}^{p}\frac{\partial J_{1,i}}{\partial w} + \frac{w}{W^{n+1}}\frac{\partial J_2}{\partial \mathbf{M}} \tag{17}$$

where :

$$\frac{\partial w}{\partial M_{rl}} = -\frac{1}{2}w(\mathbf{x}-\mathbf{c})^T\frac{\partial \mathbf{D}}{\partial M_{rl}}(\mathbf{x}-\mathbf{c}), \quad \frac{\partial J_2}{\partial M_{rl}} = 2\gamma\sum_{i,j=1}^{n} D_{ij}\frac{\partial D_{ij}}{\partial M_{rl}}$$

$$\frac{\partial D_{ij}}{\partial M_{rl}} = \delta_{lj}M_{ri} + \delta_{il}M_{rj} \quad (\delta \text{ is the Kronecker operator})$$

$$\sum_{i=1}^{p}\frac{\partial J_{1,i}}{\partial w} \approx -\frac{E^{n+1}}{\left(W^{n+1}\right)^2} + \frac{1}{W^{n+1}}\left(\mathbf{e}_{cv}^T\mathbf{e}_{cv} - \left(2\,\mathbf{P}^{n+1}\,\tilde{\mathbf{x}}\left(\mathbf{y}-\tilde{\mathbf{x}}^T\beta^{n+1}\right)^T\right)\otimes\mathbf{H}^n - \left(2\,\mathbf{P}^{n+1}\,\tilde{\mathbf{x}}\,\tilde{\mathbf{x}}^T\mathbf{P}^{n+1}\right)\otimes\mathbf{R}^n\right)$$

Deriving this derivative is possible due to the fact that an application of the Sherman-Morrison-Woodbury theorem allows us to take derivatives through the inverted covariance matrix $\mathbf{P}$ (Belsley et al., 1980; Atkeson & Schaal, 1995), and that a sum of the form $\Sigma\mathbf{v}_i^T\mathbf{Q}\,\mathbf{v}_i$ can be written as $\Sigma\mathbf{v}_i^T\mathbf{Q}\,\mathbf{v}_i = \mathbf{Q}\otimes\Sigma\,\mathbf{v}_i\mathbf{v}_i^T$, where the operator $\otimes$ denotes an element-wise multiplication of two homomorphic matrices or vectors with a subsequent summation of all coefficients, $\mathbf{Q}\otimes\mathbf{V} = \Sigma Q_{ij}V_{ij}$. It is interesting to note that the stochastic derivative (17) is not just concerned with reducing the error of the current training point as in many other learning algorithms, but rather that it takes into account the previously encountered training data, too, through the memory traces (16). Thus, both the β and $\mathbf{M}$

update in RFWR are not greedy with respect to the current training sample, a characteristic which will contribute favorably to speed and robustness of incremental learning.

3.1.3 Adding Receptive Fields and Automatic Bias Adjustment

A new receptive field is created if a training sample $(\mathbf{x}, \mathbf{y})$ does not activate any of the existing receptive field by more than a threshold w_{gen}. The center of the new receptive field becomes $\mathbf{c} = \mathbf{x}$, $\mathbf{M}$ is set to a manually chosen default value, $\mathbf{M}_{def}$, and all other parameters are initialized to zero, except the matrix $\mathbf{P}$. $\mathbf{P}$ corresponds to an inverted covariance matrix of the weighted inputs (treating the constant input "1" as the *(n+1)*-th input). A suitable initialization of $\mathbf{P}$ is as a diagonal matrix, the diagonal elements set to $P_{ii} = 1/r_i^2$, where the coefficients r_i are usually small quantities, e.g., 0.001 (Ljung & Söderström, 1986). We summarize all r_i in the $(n+1)$-dimensional vector $\mathbf{r} = \left(r_1, r_2, \ldots, r_{n+1}\right)^T$.

The parameters $\mathbf{r}$ have an interesting statistical interpretation: they introduce bias in the regression coefficients β, and correspond to one of the common forms of biased regression, ridge regression (Belsley et al., 1980). From a probabilistic point of view, they are Bayesian priors that the coefficients of β are zero. From an algorithmic perspective, they are fake data points of the form $[\mathbf{x}_r = (0,\ldots,r_i^2,0,\ldots)^T, \mathbf{y}_r = 0]$ (Atkeson et al., 1997). Under normal circumstances, the sizes of the coefficients of $\mathbf{r}$ are too small to introduce noticeable bias. However, ridge regression parameters are important if the input data is locally rank deficient, i.e., the matrix inversion in (7) is close to singular. For high dimensional input spaces, it is quite common to have locally rank deficient input data. Although RFWR does not explicitly require matrix inversions, the rank deficiency affects the incremental update in (8) by generating estimates of β with very large variances, causing unreliable predictions. Non-zero ridge regression parameters reduce this variance, however at the cost of introducing bias. An appropriate compromise can be found by including the ridge parameters as adjustable terms in RFWR using gradient descent in the cost (12):

$$\mathbf{r}^{n+1} = \mathbf{r}^n - \alpha_r \frac{\partial J}{\partial \mathbf{r}} \tag{18}$$

After each update of $\mathbf{P}$, the change in $\mathbf{r}$ is added to $\mathbf{P}$. Additionally, it is necessary to add back the fraction of $\mathbf{r}$ which was lost due to the forgetting factor λ—bias should not be forgotten over time. These two computations can be performed together and are surprisingly simple. Appendix 8.1 details this update and the stochastic approximation of $\partial J / \partial \mathbf{r}$, which is analogous to the derivation of (17).

3.1.4 Pruning Receptive Fields

The last element in RFWR is a pruning facility. A receptive field is pruned if it overlaps too much with another receptive field. This effect is detected by a training sample acti-

vating two receptive fields simultaneously more than w_{prune}. The receptive field with the larger determinant of the distance metric **D** is pruned. For computational convenience, det(**D**) can be approximated by ΣD_{ii}^2 (Deco & Obradovic, 1996). It should be noted that pruning due to overlap aims primarily at computational efficiency, since, as discussed in Section 2.3, overlap does not degrade the approximation quality. The second cause for pruning is if the bias-adjusted weighted mean squared error

$$wMSE = \frac{E^n}{W^n} - \gamma \sum_{i,j=1}^{n} D_{ij}^2 \tag{19}$$

of the linear model of a unit is excessively large in comparison to other units—the bias adjustment term can be derived from the asymptotic behavior of RFWR, outlined in Section 3.3 and detailed in Schaal and Atkeson (1997). Empirically, there are usually two ways to adjust **M** in order to minimize (12). The one we normally want to avoid is **M**=**0**, i.e., the zero matrix. It indicates that the receptive field performs global regression instead of locally weighted regression. Global linear regression for a nonlinear function has a large *wMSE*. A simple outlier detection test among the *wMSE* of all receptive fields suffices to deal with such behavior. The receptive field is then reinitialized with randomized values. Normally, pruning takes place rarely, and if it happens, it is mostly due to an inappropriate initialization of RFWR.

3.1.5 Summary of RFWR

Initialize the RFWR with no receptive field (RF);
For every new training sample (**x**,**y**):

a) **For** k=1 to #RF:
   - calculate the activation from (5)
   - update the receptive field parameters according to (13), and (18)

   end;

b) **If** no subnet was activated by more than w_{gen};
   - create a new RF with **c**=**x**, **M**=$\mathbf{M}_{def}$

   end;

c) **If** two RFs are activated more than w_{prune}:
   - erase the RF with the larger det(**D**)

   end;

d) calculate the $m=E\{wMSE\}$ and $std=E\{(wMSE-m)^2\}^{0.5}$ of all RFs;

e) **For** k=1 to #RF:
   If $|wMSE-m| > \varphi\, std$,
   - reinitialize receptive field with $\mathbf{M} = \varepsilon\ \mathbf{M}_{def}$

   end;

end;

In summary, each receptive field in RFWR has three sets of adjustable parameters: β for the locally linear model, **M** for the size and shape of the receptive field, and **r** for the bias. The linear model parameters are updated by a Newton method, while the

other parameters are updated by gradient descent. A compact pseudo-code overview of RFWR is shown above.

The scalar φ is a (positive) outlier removal threshold, e.g., φ=2.576 or φ=3.291 (corresponding to a 99% or 99.9% confidence value with respect to a normal distribution), and the scalar ε is a random value ε=1+|N(0,1)|. This choice of ε ensures that the new distance metric will result in a smaller receptive field which is less likely to converge to a **M**=**0** solution. It useful to note that the parameters w_{prune}, w_{gen}, and φ can be chosen independently of a particular learning problem and should thus be considered more like constants of the algorithm and not open parameters.

3.2 Second Order Gradient Descent

With little extra computation, it is possible to replace the gradient descent update of **M** in (13) by second order gradient descent to gain learning speed. For this purpose, we adopted Sutton's (1992a,b) Incremental Delta-Bar-Delta (IDBD) algorithm. The derivation of the algorithm remains as demonstrated in Sutton (1992a,b), only that his standard least squares criterion is replaced by our cost function (12), and that we apply IDBD to updating a distance metric. Appendix 8.2 provides the details of the algorithm. It is also possible to apply second order learning to the ridge regression update (18). Empirically, however, we did not find any significant improvements of doing so and, hence, only incorporated second order updates for the distance metric in RFWR.

3.3 Asymptotic Properties of RFWR

In Schaal and Atkeson (1996,1997) we derived the asymptotic properties of RFWR's cost function (12). Here we will just mention some of these results that are directly relevant to this paper. Assuming i) that the number of training data points p goes to infinity, ii) that within the range of a receptive fields a second order Taylor series expansion fits the training function sufficiently accurate, iii) that the variance of the noise σ^2 is locally constant, and iv) that the input distribution is locally uniform, the following statements can be made:

- The penalty term in the cost (12) introduces non vanishing bias like a low pass filter: the higher the second derivatives (Hessian) of the function, the more bias is incurred.
- The estimated locally linear model **b** is asymptotically unbiased.
- The distance metric **D** will be a scaled approximation of the Hessian.
- An appropriate penalty term γ for a learning problem can be computed from an estimate of the maximal eigenvalues of the Hessian—this corresponds to a smoothness bias.

— A bias adjusted weighted mean squared error, *wMSE*, can be formulated in order to compare the approximation quality of receptive fields. This measure was in employed in Equation (19).

These asymptotic results confirm that the penalty term in the cost function (12) has the desired characteristics as mentioned in Section 2.3 and Section 3.1.2: receptive fields cannot shrink to zero size, and a controlled amount of bias was introduced. It is interesting that the estimated locally linear model **b** tends to become unbiased (under the assumption that $O(2)$ errors of the Taylor series are negligible). This implies that applications requiring a gradient estimate from the function approximator can expect reliable results. The calculation of the gradient estimate is a natural by-product of every lookup in RFWR.

4 Simulation Results

4.1 Basic Function Approximation with RFWR

First, we will establish that RFWR is capable of competing with state-of-the-art supervised learning techniques on a fixed training set. A sufficiently complex learning task that can still be illustrated nicely is to approximate the function

$$z = \max\left\{e^{-10x^2}, e^{-50y^2}, 1.25e^{-5\left(x^2+y^2\right)}\right\} + N(0, 0.01) \tag{20}$$

from a sample of 500 points, drawn uniformly from the unit square. This function consists of a narrow and a wide ridge which are perpendicular to each other, and a Gaussian bump at the origin (Figure 4a). Training data is drawn uniformly from the training set without replacement; training time is measured in epochs, i.e., multiples of 500 training samples. The test set consists of 1681 data points corresponding to the vertices of a 41x41 grid over the unit square; the corresponding output values are the exact function values. The approximation error is measured as a normalized mean squared error, *nMSE*, i.e., the *MSE* on the test set normalized by the variance of the outputs of the test set. RFWR's initial parameters are set to $\mathbf{M}_{def} = 5\,\mathbf{I}$ ($\mathbf{I}$ is the identity matrix), $\gamma = 10^{-7}$, $w_{gen} = 0.1$, and $w_{prune} = 0.9$. The pruning and generation thresholds are of minor importance, just determining the overlap of the receptive fields. The choice for the penalty term was calculated to tolerate a maximal bias of 0.1 (Schaal & Atkeson, 1997). The default value for the decomposed distance metric was determined manually such that an initial receptive field covered a significant portion of the input space. Ridge regression parameters did not play any role in this example and were omitted.

A first qualitative evaluation of Figure 4 confirms that RFWR fulfills our expectations. The initially large receptive fields (Figure 4c) adjust during learning according to the local curvature of the function: they become narrow and elongated in the region of the ridges, and they remain large in the flat parts of the function ((Figure 4d). The num-

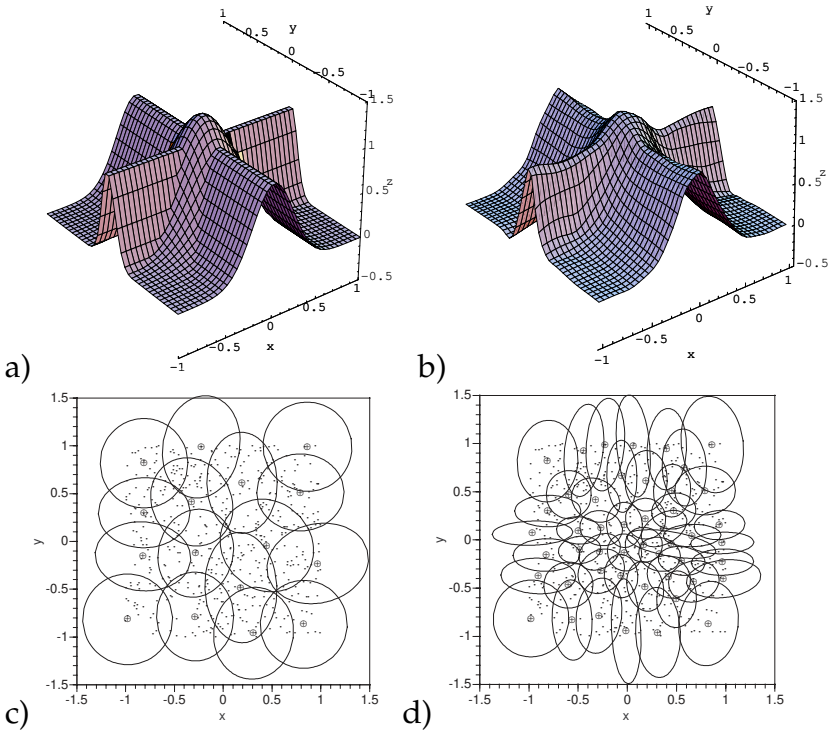


Figure 4: a) target function to be approximated; b) approximated function after 50 epochs of training; c) receptive fields in input space after 1 epoch, given by contour lines of 0.1 isoactivation and a ⊕ mark for the centers (the training data is displayed by small dots); d) receptive fields after 50 epochs of training.

ber of the receptive fields increased from 16 after one training epoch to 48, and the final approximation result was *nMSE*=0.02.

We compared the learning results of RFWR with 3 other algorithms: standard global linear regression and a sigmoidal 3-layer backpropagation neural network as baseline comparisons, and the mixture of experts algorithm as a state-of-the-art comparison (Jacobs et. al, 1991; Jordan & Jacobs, 1994; Xu, Jordan, & Hinton, 1995). Standard linear regression cannot accomplish a better result than *nMSE*=1.0 on this example—the function has no linear trend in the chosen region of input space. The sigmoidal network was trained by backpropagation with momentum in a variety of configurations using 20 to 100 units in the hidden layer (the output layer had one linear unit). These networks did not accomplish results better than *nMSE*=0.1 within 20000 training epochs. Doubling the number of training samples and reducing the noise level to N(0,0.0001) finally resulted in *nMSE*=0.02 for a 100 hidden unit net after about 15000 epochs. By using the Cascade Correlation algorithm (Fahlman & Lebiere, 1990) to fit our original 500 data point training set we confirmed that the function (20) seems to be a difficult learning task for sigmoidal networks: Cascade Correlation did not converge when confined to using only sigmoidal hidden units, while it achieved good function fitting (*nMSE*=0.02) when it was allowed to use Gaussian hidden units.

A more natural and interesting comparison is with the mixture of experts (ME) system, particularly as suggested in Xu et al. (1995). In Xu et al. (1995), in contrast to the softmax gating network of Jordan and Jacobs (1994), the experts use a mixture of Gaussians as the gating network, and both the gating net and the locally linear models in each leaf of the gating net can be updated by an analytical version of the Expectation–Maximation (EM) algorithm (Dempster, Laird, & Rubin, 1977). Thus, the basic elements of this form of ME are the same as in RFWR—locally linear models and Gaussian receptive fields—while the training methods of the two systems differ significantly—competitive parametric likelihood maximization vs. local nonparametric learning. As ME does not add resources, the performance determining parameters are how many experts are allocated and how the system is initialized. The algorithm was tested with 25, 50, 75, and 100 experts. Initially, the experts were uniformly distributed in the input space with an initial covariance matrix of the Gaussians comparable to the initialization of RFWR's distance metric. We conducted a similar test with RFWR, setting its determining parameter, the penalty γ, to 10^{-6}, 10^{-7}, 10^{-8}, and 10^{-10}.

Figure 5 summarizes the results. Each learning curve is the average of 10 learning trials for each condition of the corresponding algorithm; the training data was randomly generated for each trial. Both algorithms achieve a *nMSE*=0.12 after only one training epoch—a typical signature of the fast recursive least squares updating of the linear models employed by both algorithms—which is about what the sigmoidal neural network had achieved after 10000 to 20000 epochs. Both algorithms converge after about 100 epochs. By adding more experts, the mixture of experts improves its performance to a best average value of *nMSE*=0.04 with a slight trend to overfitting for 75 experts. RFWR accomplishes consistently a result of *nMSE*=0.02 for all but the $\gamma=10^{-6}$ runs, with a slight tendency to overfitting for $\gamma=10^{-10}$. One standard deviation error bars are indicated by the black bars at the beginning and end of each learning curve.

It was surprising that ME did not achieve the same ultimate fit accuracy as RFWR. This behavior was due to a) the relative small training set, b) the relatively low signal to noise ratio of the training data, and c) the way the gating network assigns training samples to each expert. By significantly increasing the amount of training data and/or lowering the noise, the results of both algorithms become indistinguishable. It seems to be the method of credit assignment which makes a significant difference. The expectation step in ME uses normalized weights (i.e., posterior probabilities) to assign training data to the experts. Normalized weights create much sharper decision boundaries between the experts than unnormalized weights as in RFWR. Thus, in the case of noise and not too much training data, the ME algorithm tends to establish too sharp decision boundaries between the experts and starts fitting noise. Given the underlying assumption of ME that the world was generated by a mixture of linear models, this behavior may be expected. Since in our test cases, the world is actually a continuous function and not a mixture of linear models, the assumptions of ME are only an approximation, which explains why the algorithm does not perform entirely appropriately.

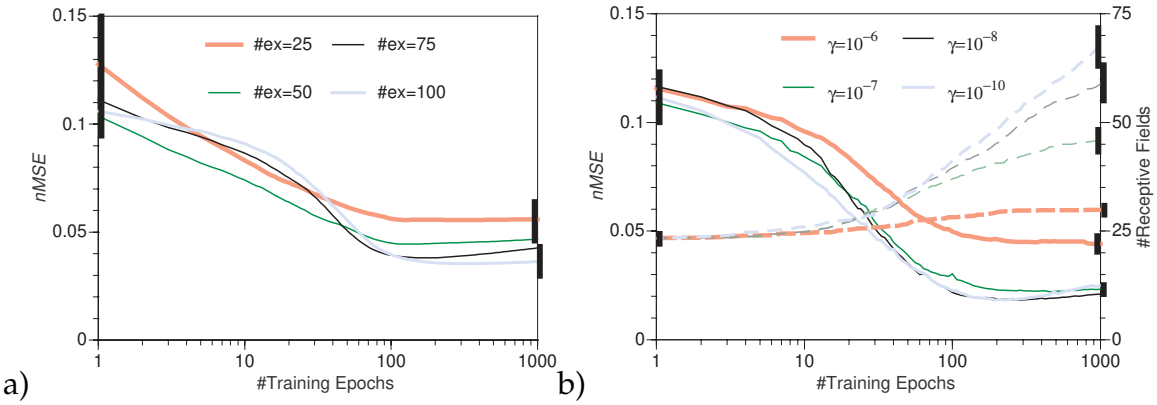


Figure 5: Average learning curves (solid lines) for a) ME, and b) RFWR. The black bars indicate one standard deviation error bars at the beginning and at the end of learning; for overlapping traces having approximately the same standard deviation, only one bar is shown. For RFWR (b), the increase of the number of receptive fields over time (dashed lines) is indicated as well.

The assumptions of RFWR are quite different: each receptive field tries to find a region of validity which allows it to approximate the tangent plane in this region with some remaining bias. In the spirit of a low order Taylor series expansion, this is a reasonable way to proceed. Thus, RFWR achieves consistent results with low variance (Figure 5b). It is also interesting to see how the number of receptive fields of RFWR grows as a function of the penalty factor (Figure 5b). As expected from the derivation of the cost function (12), a very small penalty parameter causes the receptive fields to keep on shrinking and entails a continuous growth of the number of receptive fields. Nevertheless, the tendency towards overfitting remained low, as can be seen in the $\gamma = 10^{-10}$ traces in Figure 5b. When continuing learning until 10000 epochs, the *nMSE* saturated close to the current values for all penalty factors. The local cross validation term in (12) is responsible for this desirable behavior—when cross validation was not used, overfitting was significantly more pronounced and the *nMSE* continued increasing for very small penalty factors.

4.2 Dealing With Irrelevant Inputs

In order to establish the usefulness of the ridge regression parameters, we conducted a further comparison with the Mixture of Experts. In sensorimotor control, it is unlikely that all variables given to the learning system are equally relevant to the task. Possible kinds of extraneous inputs include: a) constant inputs, b) changing inputs which are meaningless, and c) copies and linear combinations of other inputs. Ideally, one would like an autonomous learning system to be robust towards such signals. To explore the behavior of ME and RFWR in such cases, three additional inputs were added to the function (20): a) one almost constant input of N(0.1,0.001), b) one input with a Brownian walk in the interval [-0.1,0.1], and c) one input which was a copy of x with added Gaussian noise N(0,0.0025). Otherwise, training data was generated uniformly by the func-

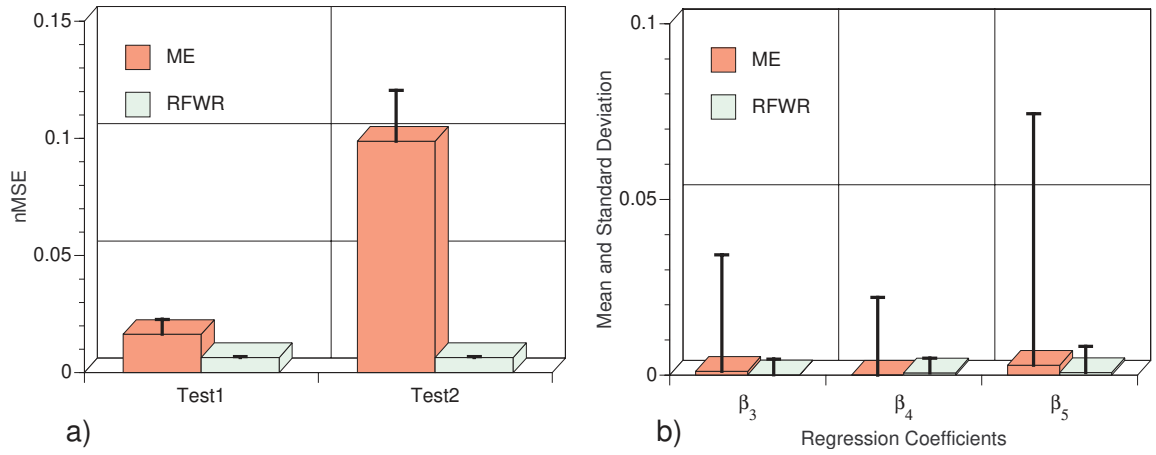
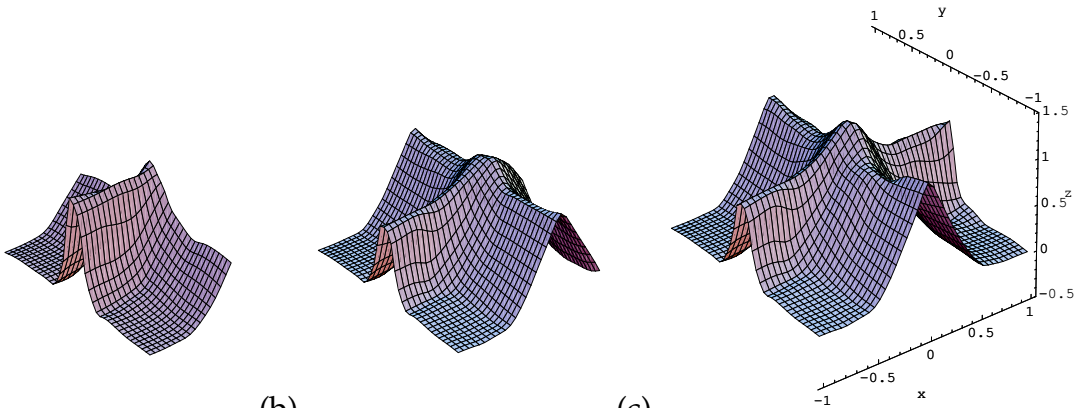


Figure 6: a) average nMSE of ME and RFWR after 1000 training epochs (see text for further explanations); b) mean and standard deviation of the regression coefficients of the irrelevant inputs.

tion (20), but with reduced additive noise of N(0,0.0025) to improve the signal to noise ratio. For these tests, the ridge regression coefficients were initialized to 0.25 for each input.

Figure 6 summarizes the average results of 10 trials for each algorithm. In Figure 6a, we show the mean *nMSE* and its standard deviation on two test sets. In Test1, the predictions were generated by using only the regression coefficients of the relevant inputs, i.e., $\beta_0, \beta_1, \beta_2$, on the same 1681 point test set as in the experiment of Section 4.1. This was to establish whether these coefficients adjusted correctly to model the target function. Both algorithms achieved good learning results on this test (Figure 6a). In Test2, we probed the robustness of the learned model towards the irrelevant inputs: we added the noisy constant, the Brownian, and the noisy x-copy input to the test set, but we also added an offset of 0.1 to each of these signals. If the algorithm learned that these inputs were irrelevant, this change should not matter. However, if the irrelevant inputs were mistakenly employed as signal to improve the *nMSE* on the training data, the predictions should deteriorate. Figure 6a demonstrates that the results of RFWR remained virtually unaltered by this test, while those of ME became significantly worse. This outcome can be explained by looking at the standard deviations of the regression coefficients of all the locally linear models (Figure 6b). In contrast to ME, RFWR set the regression coefficients of the irrelevant inputs ($\beta_3, \beta_4, \beta_5$) very close to zero, thus achieving the desired robustness. Such behavior was due to an adjustment of the corresponding ridge regression parameters: they increased for the irrelevant inputs and decreased to zero for the relevant inputs. We should point out that ME was not designed to deal with learning problems with irrelevant inputs, and that there are ways to improve its performance in such cases. Nevertheless, this experiment clearly illustrates that it is necessary to deal with the problem of irrelevant inputs, and that local bias adjustment by means of ridge regression is one possible way to do so.



(a) (b) (c)

Figure 7: RFWR reconstructed function after training on (a) T_1, (b) then T_2, (c) and finally T_3.

4.3 Shifting Input Distributions

As mentioned in the Introduction, it is easy to conceive of learning tasks where the input distribution of the training data changes over time. To test RFWR's performance on such problems, we designed the following experiment. In three sequential episodes training data for learning (20) was uniformly drawn from three slightly overlapping input regions in the unit cube: $T_1 = \{(x,y,z) \mid -1.0 < x < -0.2\}$, $T_2 = \{(x,y,z) \mid -0.4 < x < 0.4\}$, and $T_3 = \{(x,y,z) \mid 0.2 < x < 1.0\}$. First the algorithm was trained on T_1 for 50,000 points and tested on T_1, then trained on T_2 for 50,000 points and tested on T_1 and T_2, and finally trained on T_3 for 50,000 points and tested on test data from all regions. Figure 7 gives an example of how learning proceeded. This test probes how much of the previously learned competence is forgotten when the input distribution shifts. All parameters of RFWR were chosen as in 4.1.

As the ME algorithm is not constructive and thus not well suited for learning with strongly shifting input distributions, we chose the Resource Allocating Network (RAN) of Platt (1991) for a comparison, a learning algorithm which is constructive, which has no competitive learning component, and which has inspired a variety of other algorithms. RAN is a radial basis function (RBF) network that adds RBFs at the site of a training sample according to two criteria: a) when the approximation of the training sample error is too large, and b) when no RBF is activated by the training sample more than a threshold ξ value. Both criteria have to be fulfilled simultaneously to create a new RBF. The spherical width of the RBF is chosen according to its distance to the nearest neighboring RBF. By using gradient descent with momentum, the RBF centers are adjusted to reduce the mean squared approximation error, as are the weights of the linear regression network in the second layer of the RBF net. The strategy of RAN is to start initially with very wide RBFs and to increase the threshold ξ over time until a prechosen upper limit is reached, causing the creation of ever smaller RBFs at sites with

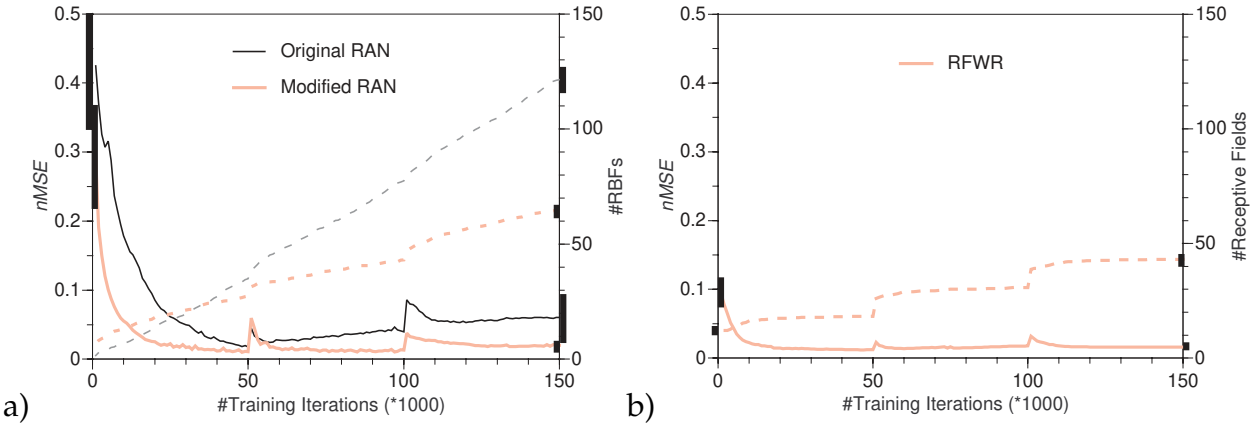


Figure 8: Average learning curves (solid lines) and average number of receptive field/radial basis functions (dashed lines) for a) RAN, and b) RFWR. The black bars give the one standard deviation at the beginning and the end of learning.

large error. As in RFWR, we used Gaussians (Equation (5)) as the parametric structure for the RBF.

Figure 8 summarizes the average of 10 learning trials for each algorithm. RFWR shows large robustness towards the shift of input distribution: there is only a minor increase of *nMSE* due to interference in the overlapping parts of the training data (Figure 7). In contrast, as can be seen in the "original RAN" trace of Figure 8a, RAN significantly increases the *nMSE* during the second and third training episode. Since RAN starts out with initial RBFs which cover the entire input space interference is not properly localized, which explains the observed behavior. Note that we already have excluded the constant term in the linear regression layer of RAN (Platt, 1991), a term that is globally active and would decrease the performance in Figure 8 significantly.

From the experience with RFWR, three possible improvements of RAN come to mind. First, instead of starting with very large RBFs initially, we can limit the maximal initial size as in RFWR to $\mathbf{M}_{def}$. Second, we can employ the hyper radial basis function technique of Poggio and Girosi (1990) to also adjust the shape $\mathbf{M}$ of the RBFs by gradient descent as in RFWR (Furlanello, Giuliani, & Trentin, 1995). And third, instead of having the time varying threshold ξ a global variable, we can define it as an individual variable for each RBF, thus removing the explicit dependency on global training time. By initializing RAN with $\mathbf{M}_{def} = 5\,\mathbf{I}$ as in RFWR, these modification resulted in a significant improvement of robustness of RAN as shown in Figure 8a. Note that this version of RAN requires only half as many RBFs, converges more quickly, and achieves very low final approximation errors. As in RFWR, localizing the learning parameters leads to a clear improvement of robustness of incremental learning.

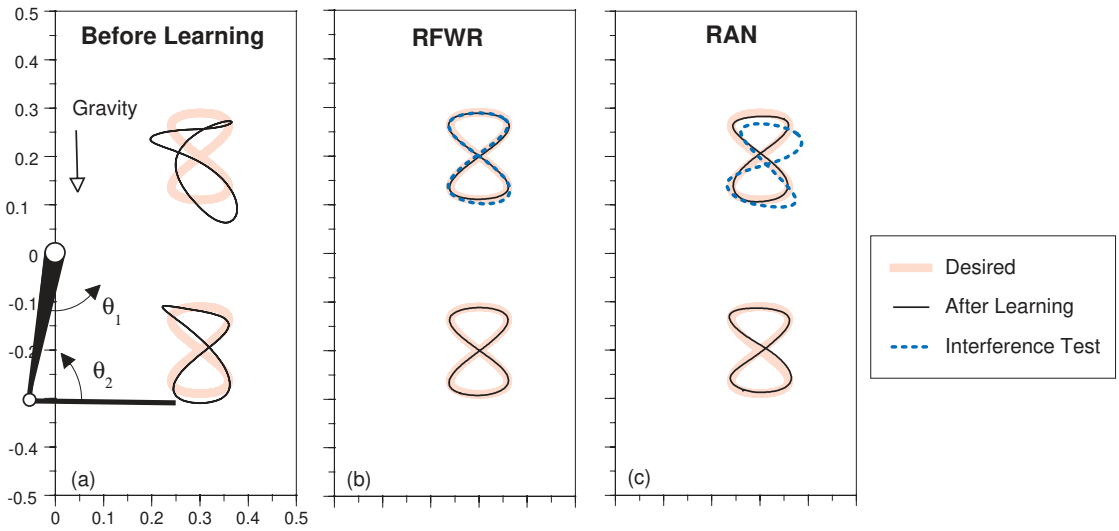


Figure 9: a) Initial performance of the two joint arm when drawing the figure "8" without feedforward control signals; b) performance of RFWR after learning; c) performance of RAN after learning.

4.4 Sensorimotor Learning

As a last evaluation, we use a traditional example of sensorimotor learning, the approximation of the inverse dynamics of a two-joint arm (Atkeson, 1989). The configuration of the arm is given by two joint angles, θ_1 and θ_2 (Figure 9a). The inverse dynamics model is the map from the two joint angles, two joint velocities, and two joint accelerations to the corresponding torques (we assume that the arm controller makes use of a low gain feedback PID controller whose performance is enhanced by feedforward commands from the learned inverse dynamics (An, Atkeson, & Hollerbach, 1988)). The torques for the shoulder and elbow joint are learned by separate networks as there is no reason to believe that a receptive field for the elbow torque should have the same shape as for the shoulder torque—for RFWR this would mean that both outputs have the same Hessian which is definitely not the case. The task goal is to draw a figure "8" in two parts of the work space. Figure 9a shows the desired and the initial performance without the learned commands. Training proceeded in two steps: first, the arm performed sinusoidal movements with varying frequency content in the area of the upper "8". A total of 45,000 training points, sampled at 100Hz, was used for training—each training sample was only used once in the sequential order it was generated. The learning results are shown in the top part of Figure 9b for RFWR, and Figure 9c for the modified RAN. Both algorithms were able to track the figure "8" properly.

Next, the algorithms were trained in an analogous fashion on 45,000 samples around the lower figure "8". The bottom parts of Figure 9b,c show the corresponding good learning results. However, when returning to performing the upper figure "8", RAN showed significant interference (dashed line in Figure 9c), although both algorithms

were initiated with the same $\mathbf{M}_{def} = 6.0\mathbf{I}$ (note that position, velocity, and acceleration inputs were normalized prior to learning to compensate for the differences in units). This interference effect highlights the difference between the learning strategy of RBF networks in comparison to the nonparametric statistics approach to modeling with locally linear models. RBF networks need a sufficient overlap of the radial basis functions to achieve good learning results—one RBF by itself has only limited function approximation capabilities, an effect discussed in the context of hyperacuity (e.g., Churchland & Sejnowski, 1992). Gradient descent on the shape parameter $\mathbf{M}$ of the Gaussian RBFs quickly decreased $\mathbf{M}$ in our example to achieve an appropriately large overlap. This overlap, however, encourages negative interference, as is evident in Figure 9c. The 6-dimensional input space of this example emphasized the need for large overlap, while the 2-dimensional example of the previous section did not. Experiments which used a fixed $\mathbf{M}$ as in the original RAN algorithm did not achieve better learning results within a reasonable training time. To avoid interference, there is always the unattractive solution of adding thousands of quite narrow overlapping RBFs. In the results of Figure 9, both algorithms allocated less than 100 receptive fields.

5 Related Work

The field which contributes the most to the development of RFWR is nonparametric statistics. Cleveland (1979) introduced the idea of employing locally linear models for memory-based function approximation, called locally weighted regression (LWR). In a series of subsequent papers, he and his colleagues extended the statistical framework of LWR to include multi-dimensional function approximation and local approximation techniques with higher order polynomials (e.g., Cleveland, Devlin, & Gross, 1988; Clevland & Devlin, 1988). Cleveland and Loader (1995) suggested local C_p-tests and local PRESS for choosing the degree of local mixing of different order polynomials as well as local bandwidth adjustment and reviewed a large body of literature on the history of LWR. Hastie and Tibshirani (1990, 1994) give related overviews of nonparametric regression methods. Hastie and Loader (1993) discuss the usefulness of local polynomial regression and show that locally linear and locally quadratic function fitting have appealing properties in terms of the bias/variance trade-off. Friedman (1984) proposed a variable bandwidth smoother for one dimensional regression problems. Using different statistical techniques, Fan and Gijbels (1992, 1995) suggested several adaptive bandwidth smoothers for LWR and provided detailed analyses of the asymptotic properties of their algorithms.

For the purpose of time series prediction, LWR was first used by Farmer and Siderowich (1987, 1988). Atkeson (1989) introduced the LWR framework for supervised learning in robot control. Moore (1991) employed LWR for learning control based on learning forward models. In the context of learning complex manipulation tasks with a robot, Schaal and Atkeson (1994a,b) demonstrated how LWR can be extended to allow for local bandwidth adaptation by employing local cross validation and local confidence criteria. Schaal and Atkeson (1996) introduced the first non memory-based ver-

sion of LWR. Schaal (1997) applied RFWR for value function approximation in reinforcement learning. Locally weighted learning for classification problems can be found, e.g., in Lowe (1995). Aha (1997) compiled a series of papers on nonparametric local classification and regression learning, among which Atkeson, Moore, and Schaal (a,b, 1997) give an extended survey on locally weighted learning and locally weighted learning applied to control.

Besides nonparametric statistics, RFWR is related to work on constructive learning algorithms, local function approximation based on radial basis functions (RBF), and Kohonen-like self-organizing maps (SOM). A RBF function approximator with a locally linear model in each RBF was suggested by Millington (1991) for reinforcement learning. Platt (1991) suggested a constructive RBF-based learning system. Furlanello et al. (1995b) and Furlanello and Giuliani (1995a) extended Platt's method by using Poggio and Girosi's (1990) hyper radial basis functions and local principal component analysis. For learning control, Cannon and Slotine (1995) derived a constructive radial basis function network which used wavelet-like RBFs to adapt to spatial frequency; this is similar to local bandwidth adaptation in nonparametric statistics and the adjustable receptive fields in RFWR. Orr (1995) discussed recursive least squares methods and ridge regression for learning with radial basis function networks. He also suggests several other methods, e.g., generalized cross validation, for regularizing ill-conditioned regression.

One of the most established constructive learning systems is Cascade Correlation (Fahlman & Lebiere, 1990), a system sharing ideas with projection pursuit regression (Friedman, 1981). Related to this line of research is the Upstart algorithm of Frean (1990), the SOM based cascading system of Littman and Ritter (1993), and the work of Jutton and Chentouf (1995). The first usage of locally linear models for regression problems in the context of SOMs was by Ritter and Schulten (1986) who extended Kohonen maps to fit locally linear models (LLM) within each of the units of the SOM. Related to this work is Smagt and Groen's (1995) algorithm which extended LLM to a hierarchical approximation in which each Kohonen unit itself can contain another LLM network. Fritzke (1994, 1995) demonstrated how SOMs can constructively add units, both in the context of RBF and LLM regression problems. Bruske and Sommer (1995) combined Fritzke's ideas with Martinetz and Schulten's (1994) Neural Gas algorithm to accomplish a more flexible topographic representation as in the original SOM work. A large body of literature on constructive learning stems from fitting high order global polynomials to data, for instance, as given in Sanger (1991), Sanger, Sutton, and Matheus (1992), and Shin and Ghosh (1995). Due to the global character of these learning methods, the danger of negative interference is quite large. Additional references on constructive learning for regression can be found in the survey by Kwok and Yeung (1995).

The idea of the mixture of experts in Jacobs et al. (1991) and hierarchical mixtures of experts in Jordan and Jacobs (1994) is related to RFWR as the mixture of experts approach looks for similar partitions of the input space, particularly in the version of Xu et al. (1995). Ormeneit and Tresp (1995) suggested methods to improve the generalization

of mixture models when fit with the EM algorithm (Dempster et al, 1977) by introducing Bayesian priors. Closely related to the hierarchical mixture of experts are nonparametric decision-tree techniques, in which the seminal work of Breiman, Friedman, Olshen, and Stone introduced classification and regression trees (CART), and Friedman (1991) proposed the MARS algorithm, a CART derivative particularly targeted at smooth function approximation for regression problems.

Finally, adaptive receptive fields and the way receptive fields are created in RFWR resemble in part the classification algorithms of Reilly, Cooper, and Elbaum (1982) and Carpenter and Grossberg (1987).

6 Discussion

This paper emphasizes two major points. First, truly local learning—i.e., learning without competition, without gating nets, without global regression on top of the local receptive fields—is a feasible approach to learning, and, moreover, it can compete with state-of-the-art learning systems. Second, truly incremental learning—i.e., learning without knowledge about the input and conditional distributions, learning that must cope with continuously incoming data with many partially redundant and/or partially irrelevant inputs—needs to have a variety of mechanisms to make sure that incremental learning is robust. A carefully designed local learning system can accomplish this robustness.

RFWR borrowed in particular from work in nonparametric statistics. Following the definition of Hájek (1969), the term "nonparametric" indicates that the function to be modeled potentially consists of very large families of distributions which cannot be indexed by a finite-dimensional parameter vector in a natural way. This view summarizes the basic assumptions of our learning system, with the addition of prior knowledge about smoothness incorporated in a penalty term. It should be stressed that, if more prior knowledge is available for a particular problem, it should be incorporated in the learning system. It is unlikely that a nonparametric learner outperforms problem-tailored parametric learning—e.g., fitting sinusoidal data with a sinusoid is the best one can do. The examples given throughout this paper are to highlight when local nonparametric learning can be advantageous, but there is no claim that it is generally superior over other learning systems. On the other hand, when it comes to learning without having strong prior knowledge about the problem, nonparametric methods can be quite beneficial. For instance, Quartz and Sejnowski (in press) claim that constructive nonparametric learning might be one of the key issues in understanding the development of the organization of brains.

RFWR makes use of several new algorithmic features. We introduced a stochastic approximation to leave-one-out local cross validation, i.e., cross validation which does not need a validation set anymore. This technique can potentially be useful for many other domains as it only requires that the (local) parameters to be estimated are linear in the inputs. By employing a novel penalized local cross validation criterion, we were

able to derive locally adaptive multidimensional distance metrics. These distance metrics can be interpreted as local approximations of the Hessians of the function to be modeled. In order to speed up learning of the distance metric, we derived a second order gradient descent method. Finally, the penalized local cross validation criterion could also be employed to achieve automatic local bias adjustment of the relevance of input dimensions, obtained by local ridge regression. Using all these features, the constructive process of RFWR only needs to monitor the activation strength of all receptive fields in order to decide when to create a new receptive field—most constructive learning system need to monitor an approximation error criterion as well, which can easily lead to an unfavorable bias-variance tradeoff.

Several issues have not been addressed in this paper and are left to future research. RFWR makes use of gradient-based learning which requires a proper choice of learning rates. Even though we incorporated second order learning derived from Sutton (1992a,b), it may still be useful to do some experimentation with the choice of the learning rates in order to achieve close to optimal learning speed without entering unstable domains. It is also necessary to choose a roughly appropriate initial distance metric $\mathbf{D}$ (cf. Equation (5)), characterizing the initial size of a receptive field. A way too large initial receptive field has the danger that the receptive field grows to span the entire input domain: the initial receptive field has to be such that structure in the data cannot be mistaken for high variance noise. As a positive side-effect of local learning, however, these open parameters can be explored by allowing just a small number of receptive fields on an initial data set and monitoring their learning behavior—each receptive field learns independently and there is no need to do parameter exploration with a large number of receptive fields.

A last algorithmic point concerns computational complexity. Recursive least squares is an $O(n^2)$ process, i.e., quadratic in the number of inputs, and the update of a full distance metric is worse than $O(n^2)$. If the dimensionality of the inputs goes beyond about 10, a learning task with many receptive fields will run fairly slowly on a serial computer. Fitting only diagonal distance metrics alleviates this effect and might be necessary anyway since the number of open parameters in the learning system might become too large compared to the number of training data points.

This discussion naturally leads to the long standing question of how local learning methods can deal with high dimensional input spaces at all. As nicely described in Scott (1992), the curse of dimensionality has adverse effects on all systems which make use of neighboring points in the Euclidean sense, since the concept of "neighborhood" becomes gradually more counterintuitive when growing beyond 10 input dimensions, and it pretty much vanishes beyond 20 dimensions: every point is about the same distance from every other point. In such domains, the parametric model chosen for learning—be it local or global—becomes the key to success, essentially meaning that any learning system requires strong bias in high-dimensional worlds. However, it remains unclear whether high dimensional input spaces have *locally* high dimensional distributions. Our experience in sensorimotor learning is that this may not be true for many in-

teresting problems, as physical systems do not realize arbitrary distributions. For instance, a seven degree-of-freedom anthropomorphic robot arm, whose inverse dynamics model requires learning in a 21-dimensional input space, seems to realize locally not more than 4-8 dimensional input distributions. In Vijayakumar and Schaal (1997) we we incorporated local dimensionality reduction as a preprocessing step in every receptive field, allowing us to approximate high dimensional data successfully ().

As a last point, one might wonder in how far a local learning system like RFWR could have any parallels with neurobiological information processing. Particularly inspired by work on the visual cortex, one of the mainstream assumptions about receptive field-based learning in the brain is that receptive fields are broadly tuned and widely overlapping, and that the size of the receptive fields does not seem to be a free parameter in normal learning (as opposed to developmental and reorganizational processes after lesions, e.g., Merzenich, Kaas, Nelson, Sur, & Felleman, 1983). This view emphasizes that accuracy of encoding must be achieved by subsequent postprocessing steps. In contrast, RFWR suggest overlapping but much more finely tuned receptive fields, such that accuracy can be achieved directly by one or several overlapping units. Fine tuning can be achieved not only by a change of the size of the receptive field, but also by "plug-in" approaches where several receptive fields tuned for different spatial frequencies contribute to learning (Cannon & Slotine, 1995). To distinguish between those two principles, experiments that test for interference and generalization during learning can provide valuable insights into the macroscopic organization of learning. In motor control, the work by Shadmehr and Mussa-Ivalidi (1994), Imamizu, Uno, and Kawato (1995), and Shadmehr, Brashers-Krug, and Mussa-Ivaldi (1995) are examples of such investigations.

Whether the learning principles of RFWR are biologically relevant or not remains speculative. What we have demonstrated, however, is that there are alternative and powerful methods to accomplish incremental constructive learning based on local receptive fields, and it might be interesting to look out for cases where such learning systems might be applied. Receptive field-based local learning is an interesting research topic for neural computation, and truly local learning methods are just starting to demonstrate their potential.

7 Acknowledgments

Sethu Vijayakumar's helpful comments contributed significantly to improve the manuscript. This research was partly funded by the ATR Human Information Processing Research Laboratories. Additional support for S. Schaal was provided by the German Research Association, the German Scholarship Foundation, and the Alexander von Humboldt Foundation. 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.

8 Appendix

8.1 Ridge Regression Derivatives

Each ridge regression parameter can be conceived of as a weighted data point of the form $[\mathbf{x}_r = r_i^2(0,...,1,0,...)^T, \mathbf{y}_r = 0]$ which was incorporated in the regression by the recursive least squares update (8). Thus, the derivative of the cost function (12) is a simplified version of the derivative (17):

$$\frac{\partial J}{\partial r_i} = \frac{2r_i}{W^{n+1}}\left(-\left(2\,\mathbf{P}^{n+1}\,\mathbf{x}_r\left(\mathbf{y}_r - \mathbf{x}_r^T\beta^{n+1}\right)^T\right)\otimes\mathbf{H}^{n+1} - \left(2\,\mathbf{P}^{n+1}\,\mathbf{x}_r\mathbf{x}_r^T\mathbf{P}^{n+1}\right)\otimes\mathbf{R}^{n+1}\right) \tag{21}$$

By taking advantage of the many zero elements of the ridge "data points", the actual computation of this derivative is greatly sped up.

There are several ways to incorporate the update of the ridge regression parameters in the matrix **P**, and it should be noted that we also need to add back the fraction of the ridge parameters which was forgotten due to the forgetting factor λ in each update of **P** (Equation (8)). It turns out, that there is a quite efficient way to perform this update. At every update of a receptive field, the forgetting factor effectively reduces the contribution of each ridge parameter by:

$$\Delta_{\lambda,i} = (1-\lambda)\,r_i^2 \tag{22}$$

The update due to gradient descent is:

$$\Delta_{grad,i} = \left(r_i + \Delta r_i\right)^2 - r_i^2 \tag{23}$$

and the total increment becomes:

$$\Delta_i = \Delta_{\lambda,i} + \Delta_{grad,i} = (1-\lambda)\,r_i^2 + \left(r_i + \Delta r_i\right)^2 - r_i^2 = -\lambda\,r_i^2 + \left(r_i + \Delta r_i\right)^2 \tag{24}$$

Due to the fact that the ridge vectors are all unit vectors, it is possible to update **P** by just executing a recursive least squares update for the *increment*, i.e., to add a ridge data point of the form $[\mathbf{x}_r = \Delta_i(0,...,1,0,...)^T, \mathbf{y}_r = 0]$ for every ridge parameter by using Equation (8). This update can be accelerated by taking into account the zeros in the ridge points. An additional speed up can be obtained by not updating **P** every iteration but rather by accumulating the increments until they exceed a manually chosen threshold.

8.2 Second Order Learning of the Distance Metric

The idea of the Incremental Delta-Bar-Delta (IDBD) algorithm (Sutton, 1992a,b) is to replace the learning rate α in the gradient descent update (13) by an individual learning rate for each coefficient of **M** of the following form:

$$M_{ij}^{n+1} = M_{ij}^n - \alpha_{ij}^{n+1}\frac{\partial J}{\partial M_{ij}}, \quad \text{where} \quad \alpha_{ij}^{n+1} = \exp\left(\beta_{ij}^{n+1}\right) \quad \text{and} \quad \beta_{ij}^{n+1} = \beta_{ij}^n - \theta\frac{\partial J}{\partial M_{ij}}h_{ij}^n \tag{25}$$

Thus, the learning rates α_{ij} are changed in geometric steps by gradient descent in the meta parameter β_{ij} with meta learning rate θ. The term h_{ij} is updated as

$$h_{ij}^{n+1} = h_{ij}^{n}\left[1-\alpha_{ij}^{n+1}\frac{\partial^2 J}{\partial M_{ij}^2}\right]^{+} - \alpha_{ij}^{n+1}\frac{\partial J}{M_{ij}}, \quad \text{where we define} \quad [z]^{+} = \begin{cases} z & \text{if} \quad z>0 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

h_{ij} is initialized to zero when a receptive field is created. It corresponds to a memory term which stores a decaying trace of the cumulative sum of recent changes to M_{ij}. For more details see Sutton (1992a,b). In order to apply this second order update, it is necessary to store the parameters α_{ij}, β_{ij}, and h_{ij}, and to compute the second derivative in (26). This second derivative of the cost function (12) with respect to the coefficients of the decomposed distance metric becomes:

$$\frac{\partial J}{\partial \mathbf{M}} \approx \frac{\partial w}{\partial \mathbf{M}}\sum_{i=1}^{p}\frac{\partial J_{1,i}}{\partial w} + \frac{w}{W^{n+1}}\frac{\partial J_2}{\partial \mathbf{M}} \tag{27}$$

$$\frac{\partial^2 J}{\partial \mathbf{M}^2} \approx \frac{\partial^2 w}{\partial \mathbf{M}^2}\sum_{i=1}^{p}\frac{\partial J_{1,i}}{\partial w} + \frac{\partial w}{\partial \mathbf{M}}\sum_{i=1}^{p}\frac{\partial^2 J_{1,i}}{\partial w^2}\frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W^{n+1}}\frac{\partial^2 J_2}{\partial \mathbf{M}^2}$$

where :

$$\frac{\partial^2 w}{\partial M_{rl}^2} = \frac{1}{w}\left(\frac{\partial w}{\partial \mathbf{M}}\right)^2 - w\left(x_l - c_l\right)^2, \quad \frac{\partial^2 J_2}{\partial M_{rl}^2} = 2\gamma\left(2D_{ll} + \sum_{i,j=1}^{n}\left(\frac{\partial D_{ij}}{\partial M_{rl}}\right)^2\right)$$

$$\begin{aligned}\sum_{i=1}^{p}\frac{\partial^2 J_{1,i}}{\partial w^2} \approx & -\frac{\mathbf{e}_{cv}^T\mathbf{e}_{cv}}{\left(W^{n+1}\right)^2} \\ & -\frac{2}{W^{n+1}}\left(\left(-\frac{\mathbf{I}}{W^{n+1}} - 2\,\mathbf{P}^{n+1}\,\tilde{\mathbf{x}}\,\tilde{\mathbf{x}}^T\right)\mathbf{P}^{n+1}\,\tilde{\mathbf{x}}\left(\mathbf{y}-\tilde{\mathbf{x}}^T\beta^{n+1}\right)^T\right)\otimes\mathbf{H}^n \\ & +\frac{2}{W^{n+1}}\frac{\left(\mathbf{y}-\tilde{\mathbf{x}}^T\beta^{n+1}\right)^T\left(\mathbf{y}-\tilde{\mathbf{x}}^T\beta^{n+1}\right)h}{w} \\ & -\frac{1}{\left(W^{n+1}\right)^2}\left(\mathbf{e}_{cv}^T\mathbf{e}_{cv} - 2\left(\mathbf{P}^{n+1}\,\tilde{\mathbf{x}}\left(\mathbf{y}-\tilde{\mathbf{x}}^T\beta^{n+1}\right)^T\right)\otimes\mathbf{H}^n\right) + 2\frac{E^{n+1}}{\left(W^{n+1}\right)^3}\end{aligned}$$

Equation (27) makes use of notation and results derived in (16) and (17).

9 References

Aha, D. (1997). "Lazy Learning." *Artificial Intelligence Review*, **11**, 1-5.

An, C. H., Atkeson, C. G., & Hollerbach, J. M. (1988). *Model-based control of a robot manipulator*. Cambridge, MA: MIT Press.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). "Locally weighted learning." *Artificial Intelligence Review*, **11**, 1-5, pp.11-73.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). "Locally weighted learning for control." *Artificial Intelligence Review*, **11**, 1-5, pp.75-113.

Atkeson, C. G. (1989a). "Learning arm kinematics and dynamics." *Annual Review Neuroscience*, **12**, pp.157-83.

Atkeson, C. G. (1989b). "Using local models to control movement." In: Touretzky, D. (Ed.), *Advances in Neural Information Processing Systems 1*, pp.79-86 San Mateo, CA: Morgan Kaufmann.

Atkeson, C. G., & Schaal, S. (1995). "Memory-based neural networks for robot learning." *Neurocomputing*, **9**, pp.243-269.

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression diagnostics: Identifying influential data and sources of collinearity*. New York: Wiley.

Bishop, C., M. (1996). *Neural networks for pattern recognition*. Oxford University Press.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth International Group.

Bruske, J., & Sommer, G. (1995). "Dynamic cell structure learns perfectly topology preserving map." *Neural Computation*, **7**, pp.845-865.

Cannon, M., & Slotine, J. E. (1995). "Space-frequency localized basis function networks for nonlinear system estimation and control." *Neurocomputing*, **9**, 3, pp.293-342.

Carpenter, G. A., & Grossberg, S. (1987b). "A massively parallel architecture for a self-organizing neural pattern recognition machine." *Computer Vision, Graphics, and Image Processing*, **37**, pp.54-115.

Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain*. Boston, MA: MIT Press.

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

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

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

Cleveland, W. S., & Loader, C. (1995a). "Smoothing by local regression: Principles and methods." Technical Report, AT&T Bell Laboratories, Murray Hill, NY.

Daugman, J., & Downing, C. (1995). "Gabor wavelets for statistical pattern recognition." In: Arbib, M. A. (Ed.), *The Handbook of Brain Theory and Neural Networks*, pp.414-420. Cambridge, MA: MIT Press.

de Boor, C. (1978). *A practical guide to splines*. New York: Springer.

Deco, G., & Obradovic, D. (1996). *An information-theoretic approach to neural computation*. New York: Springer.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society B*, **39**, pp.1-38.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

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

Fan, J., & Gijbels, I. (1995). "Data-driven bandwidth selection in local polynomial fitting: Variable bandwidth and spatial adaptation." *Journal of the Royal Statistical Society B*, **57**, pp.371-395.

Fan, J., & Gijbels, I. (1996). *Local polynomical modelling and its applications*. London: Chapman & Hall.

Farmer, J. D., & Sidorowich (1987). "Predicting chaotic time series." *Phys. Rev. Lett.*, **59 (8)**, pp.845-848.

Farmer, J. D., & Sidorowich (1988b). "Exploiting chaos to predict the future and reduce noise." In: Lee, Y. C. (Ed.), *Evolution, Learning, and Cognition*, p.27. Singapore: World Scientific.

Field, D. J. (1994). "What is the goal of sensory coding?." *Neural Computation*, **6**, pp.559-601.

Frean, M. (1990). "The upstart algorithm: A method for constructing and training feedforward neural networks." *Neural Computation*, **2**, pp.198-209.

Friedman, J. H., & StŸtzle, W. (1981b). "Projection pursuit regression." *Journal of the American Statistical Association, Theory and Models*, **76**, 376, pp.817-823.

Friedman, J.H. (1984). "A variable span smoother." Technical Report No.5, Department of Statistics, Stanford University.

Friedman, J. H. (1991). "Multivariate adaptive regression splines." *The Annals of Statistics*, **19**, pp.1-141.

Fritzke, B. (1994b). "Growing cell structures _ A self-organizing network of unsupervised and supervised learning." *Neural Networks*, **7**, 9, pp.1441-1460.

Fritzke, B. (1995). "Incremental learning of locally linear mappings." In: *Proceedings of the International Conference on Artificial Neural Networks*, Paris, France, Oct.9-13.

Furlanello, C., & Giuliani, D. (1995a). "Combining local PCA and radial basis function networks for speaker normalization." In: Girosi, F., Makhoul, J., Manolakas, E., & Wilson, E. (Eds.), *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V*, pp.233-242. New York: IEEE.

Furlanello, C., Giuliani, D., & Trentin, E. (1995b). "Connectionist speaker normalization with generalized resource allocating networks." In: Tesauro, D., Touretzky, D. S., & Leen, T. K. (Eds.), *Advances in Neural Information Processing Systems 7*, pp.867-874. Cambridge, MA: MIT Press.

Geman, S., Bienenstock, E., & Doursat, R. (1992). "Neural networks and the bias/variance dilemma." *Neural Computation*, **4**, pp.1-58.

Georgopoulos, A. P. (1991). "Higher order motor control." *Annual Review of Neuroscience*, **14**, pp.361-377.

Hájek, J. (1969). *A course in nonparametric statistics*. San Francisco, CA: Holden-Day.

Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized additive models*. London: Chapman and Hall.

Hastie, T., & Loader, C. (1993). "Local regression: Automatic kernel carpentry." *Statistical Science*, **8**, pp.120-143.

Hastie, T. J., & Tibshirani, R. J. (1994c). "Nonparametric regression and classification: Part I: Nonparametric regression." In: Cherkassky, V., Friedman, J. H., & Wechsler, H. (Eds.), *From Statistics to Neural Networks: Theory and Pattern Recognition Applications. ASI Proceedings, subseries F, Computer and Systems Sciences*. Springer.

Hubel, D. H., & Wiesel, T. N. (1959). "Receptive fields of of single neurons in the cat's striate cortex." *Journal of Neurophysiology*, **148**, 574-591.

Imamizu, H., Uno, Y., & Kawato, M. (1995). "Internal representations of the motor apparatus: Implications from generalization in visuomotor learning." *Journal of Experimental Psychology*, **21**, 5, pp.1174-1198.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). "Adaptive mixtures of local experts." *Neural Computation*, **3**, pp.79-87.

Jordan, M. I., & Jacobs, R. (1994). "Hierarchical mixtures of experts and the EM algorithm." *Neural Computation*, **6**, 2, pp.181-214.

Jutten, C., & Chentouf, R. (1995). "A new scheme for incremental learning." *Neural Processing Letters*, **2**, 1, pp.1-4.

Kwok, T.-Y., & Yeung, D.-Y. (1995). "Constructive feedforward neural networks for regression problems: A survey." Technical Report HKUST-CS95-43, Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.

Lee, C., Rohrer, W. R., & Sparks, D. L. (1988). "Population coding of saccadic eye movement by neurons in the superior colliculus." *Nature*, **332**, pp.357-360.

Littmann, E., & Ritter, H. (1993). "Generalization abilities of cascade network architectures." In: Hanson, S. J., Cowan, J. , & Giles, C. L (Eds.), *Advances in Neural Information Processing Systems 5*, pp.188-195. Morgan Kaufmann.

Ljung, L., & Söderström, T. (1986). *Theory and practice of recursive identification*. Cambridge, MIT Press.

Lowe, D. G. (1995). "Similarity metric learning for a variable-kernel classifier." Neural Computation, .

Martinetz, T., & Schulten, K. (1994). "Topology representing networks." *Neural Networks*, **7**, 3, pp.507-522.

Merzenich, M. M., Kaas, J. H., Nelson, R. J., Sur, M., & Felleman, D. (1983). "Topographic reorganization of somatosensory cortical areas 3b and 1 in adult monkeys following restricted deafferentation." *Neuroscience*, **8**, pp.33-55.

Millington, P. J. (1991). "Associative reinforcement learning for optimal control." Master Thesis CSDL-T-1070, Massachusetts Institute of Technology, Cambridge, MA.

Moody, J., & Darken, C. (1988). "Learning with localized receptive fields." In: Touretzky, D., Hinton, G., & Sejnowski, T. (Eds.), *Proceedings of the 1988 Connectionist Summer School*, pp.133-143. San Mateo, CA: Morgan Kaufmann.

Moore, A. (1991a). "Fast, robust adaptive control by learning only forward models." In: Moody, J. E., Hanson, S. J., & and Lippmann, R. P. (Eds.), *Advances in Neural Information Processing Systems 4*. San Mateo, CA: Morgan Kaufmann.

Mountcastle, V. B. (1957). "Modality and topographic properties of single neurons of cat's somatic sensory cortex." *Journal of Neurophysiology*, **20**, pp.408-434.

Myers, R. H. (1990). *Classical and modern regression with applications*. Boston, MA: PWS-KENT.

Nadaraya, E. A. (1964). "On estimating regression." *Theor. Prob. Appl.*, **9**, pp.141-142.

Olshausen, B. A., & Field, D. J. (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images." *Nature*, **381**, pp.607-609.

Ormoneit, D., & Tresp, V. (1995). "Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging." Technical Report FKI-205-95, Theoretical Computer Science and Foundations of Artificial Intelligence, Technische Universität München, Munich, Germnay.

Orr, M. J. L. (1995). "Regularization in the selection of radial basis function centers." *Neural Computation*, **7**, pp.606-623.

Papoulis, A. (1991). *Probability, random variables, and stochastic processes*. New York: McGraw-Hill.

Perrone, M. P., & Cooper, L. N. (1993). "When networks disagree: Ensemble methods for hybrid neural networks." In: Mammone, R. J. (Ed.), *Neural Networks for Speech and Image processing*. Chapman-Hall.

Platt, J. (1991). "A resource-allocating network for function interpolation." *Neural Computation*, **3**, pp.213-225.

Poggio, R., & Girosi, F. (1990). "Regularization algorithms for learning that are equivalent to multilayer networks." *Science*, **247**, 4945, pp.978-982.

Powell, M. J. D. (1987). "Radial basis functions for multivariate interpolation: A review." In: Mason, J. C., & Cox. M. G. (Eds.), *Algorithms for Approximation*, pp.143-167. Oxford: Clarendon Press.

Quartz, S. R., & Sejnowski, T. J. (in press). "The neural basis of cognitive development: A constructivist manifesto." *Journal of Brain and Behavioral Sciences.*

Reilly, D. L., Cooper, L. N., & Elbaum, C. (1982). "A neural model for category learning." *Biological Cybernetics*, **45**, pp.35-41.

Ritter, H., & Schulten, K. (1986). "Topology conserving mappings for learning motor tasks." In: Denker, J. S. (Ed.), *Neural Networks for Computing*, **151**, pp.376-380. AIP Conference Proceedings, Snowbird, Utah.

Sanger, T. D. (1991). "A tree-structured adaptive network for function approximation in high-dimensional spaces." *IEEE Transactions on Neural Networks*, **2**, 2, pp.285-293.

Sanger, T. D., Sutton, R. S., & Matheus, C. J. (1992). "Iterative construction of sparse polynomial approximations." In: Hanson, S. J., Moody, J. E., & Lippmann, R. P. (Eds.), *Advances in Neural Information Processing Systems 4*, pp.1064-1071. San Mateo, CA: Morgan-Kaufmann.

Schaal, S. (1997). "Learning from demonstration." In: *Advances in Neural Information Processing Systems 9*, pp.1040-1046, Cambridge, MA: MIT Press.

Schaal, S., & Atkeson, C. G. (1994a). "Robot juggling: An implementation of memory-based learning." *Control Systems Magazine*, **14**, 1, pp.57-71.

Schaal, S., & Atkeson, C. G. (1994b). "Assessing the quality of learned local models." In: Cowan, J. , Tesauro, G., & Alspector, J. (Eds.), *Advances in Neural Information Processing Systems 6*. San Mateo, CA: Morgan Kaufmann.

Schaal, S., & Atkeson, C. G. (1996). "From isolation to cooperation: An alternative of a system of experts." In: Touretzky, D. S., Mozer, M. C., & Hasselmo, M. E. (Eds.), *Advances in Neural Information Processing Systems 8*, pp.605-611. Cambridge, MA: MIT Press.

Schaal, S., & Atkeson, C. G. (1997b). "Receptive field weighted regression." Technical Report TR-H-209, ATR Human Information Processing Laboratories, 2-2Seika-cho, Soraku-gun, Kyoto 619-02, Japan.

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

Shadmehr, R., Brashers-Krug, T., & Mussa-Ivaldi, F. A. (1995). "Interference in learning internal models of inverse dynamics in humans." In: Tesauro, G., Touretzky, D. S., & Leen, K. T. (Eds.), *Advances in Neural Information Processing Systems 7*. .

Shadmehr, R., & Mussa-Ivaldi, F. A. (1994b). "Adaptive representation of dynamics during learning of a motor task." *Journal of Neuroscience*, **14**, 5, pp.3208-3224.

Shin, Y., & Ghosh, J. (1995). "Ridge polynomial networks." *IEEE Transactions on Neural Networks*, **6**, 2. pp.610-622.

Stone, M (1974). "Cross-validatory choice and assessment of statistical predictors." *Journal of the Royal Statistical Society*, **B36**, 111-147.

Sutton, R. S. (1992a). "Gain adaptation beats least squares." In: *Proceedings of Seventh Yale Workshop on Adaptive and Learning Systems*, pp.161-166, New Haven, CT.

Sutton, R. S. (1992b). "Adapting bias by gradient descent: An incremental version of Delta-Bar-Delta." In: *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp.171-176, Cambridge, MA: MIT Press.

van der Smagt, P., & Groen, F. (1995). "Approximation with neural networks: Between local and global approximation." In: *Proceedings of the 1995 International Conference on Neural Networks*, **II**, pp.1060-1064, Perth, Austrialia.

Vijayakumar, S., & Schaal, S. (1997). "Local dimensionality reduction for locally weighted learning." IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp.220-225, Monterey, CA, July 10-11.

Wahba, G., & Wold, S. (1975). "A completely automatic french curve: Fitting spline functions by cross-validation." *Communications in Statistics*, **4 (1)**, .

Wahba, G. (1990). *Spline models for observational data*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Watson, G. S. (1964). "Smooth regression analysis." *Sankhaya: The Indian Journal of Statistics A*, **26**, pp.359-372.

Xu, L., Jordan, M. I., & Hinton, G. E. (1995). "An alternative model for mixture of experts." In: Tesauro, G., Touretzky, D. S., & Leen, T. K. (Eds.), , pp.633-640. Cambridge, MA: MIT Press.