Proc. of Seventeenth International Conference on Machine Learning (ICML2000) , pp.1079-1086 (2000)

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:

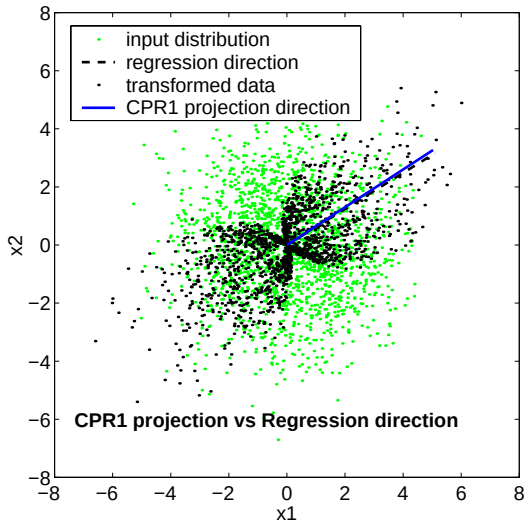
Weighting scheme CPR1: $T_{ii}^{cpr1} = \frac{1}{\sigma_{\mathbf{x}_{res}}} \left(\frac{|y_{res,i}|}{\|\mathbf{x}_{res,i}\|} \right)^p$. See Fig.1(a).

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).

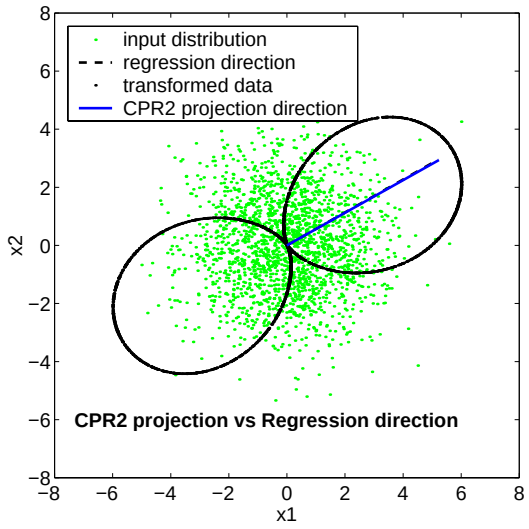
CPR1 spheres the input data and then weights it by the "slope" of each data point, taken to the power p for increasing the impact of the input/output relationships. CPR2 is a variant that, although a bit idiosyncratic, had the best average performance in practice: CPR2 first maps the input data onto 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 $1 - r^2$.

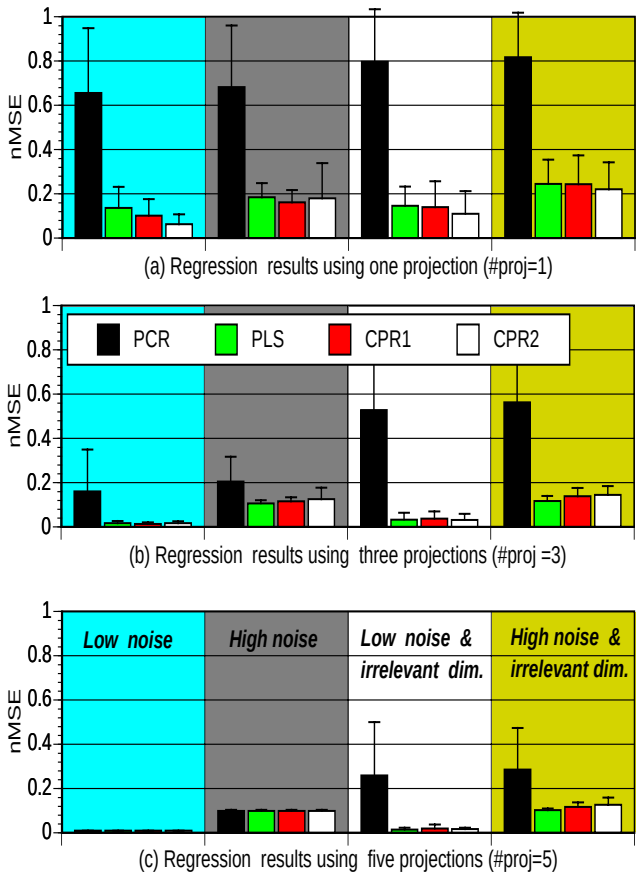


Figure 2. Simulation Results for PCR/PLS/CPR1/CPR2. The subplots show results for projection dimensions 1, 3 and 5. Each of the subplots have four additional conditions made of permutations of : (i) low and high noise (ii) With and without irrelevant(non constant) inputs.

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

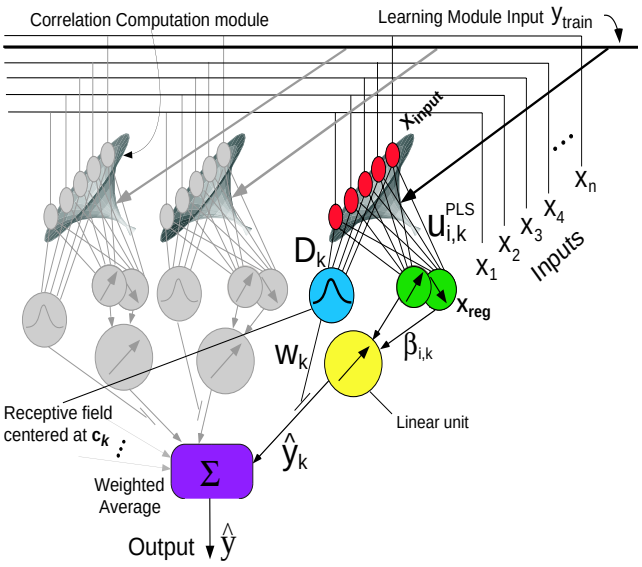


Figure 3. Information processing unit of LWPR

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.

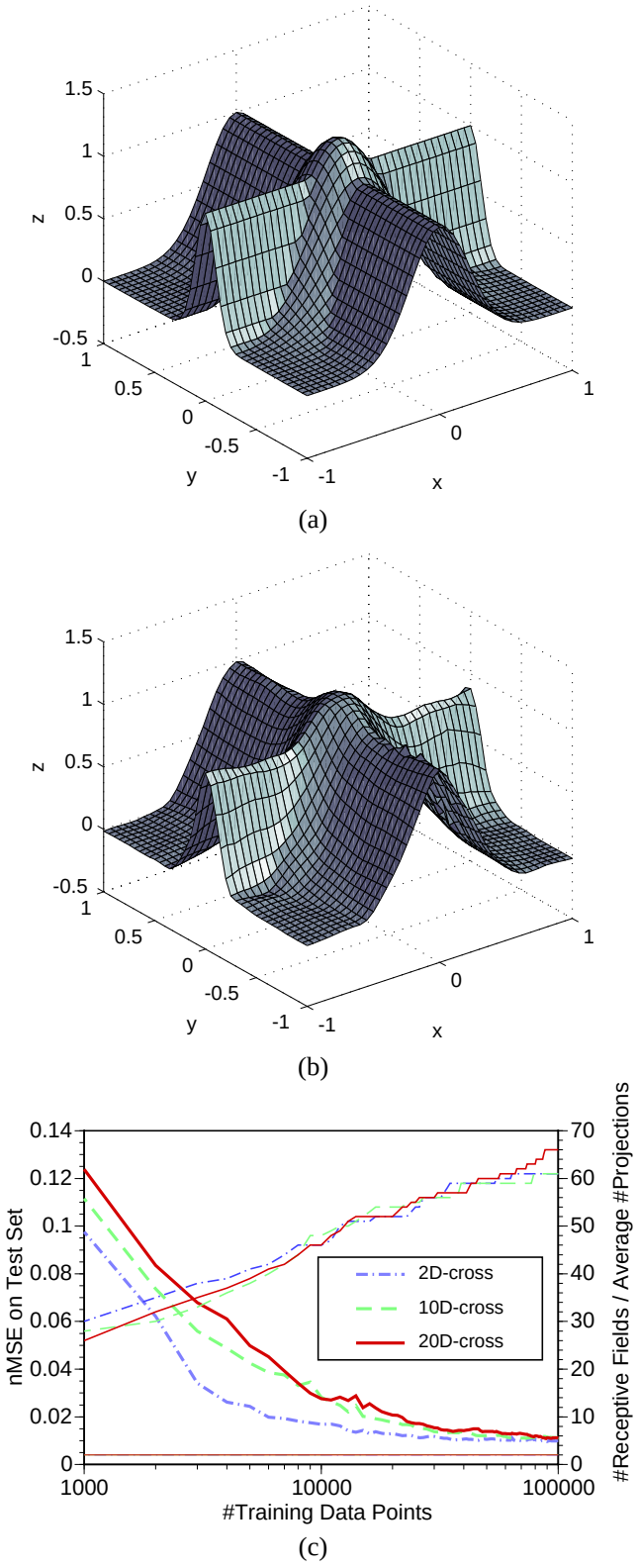


Figure 4. (a) Target and (b) learned nonlinear cross function.(c) Learning curves for 2-D, 10-D and 20-D data

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

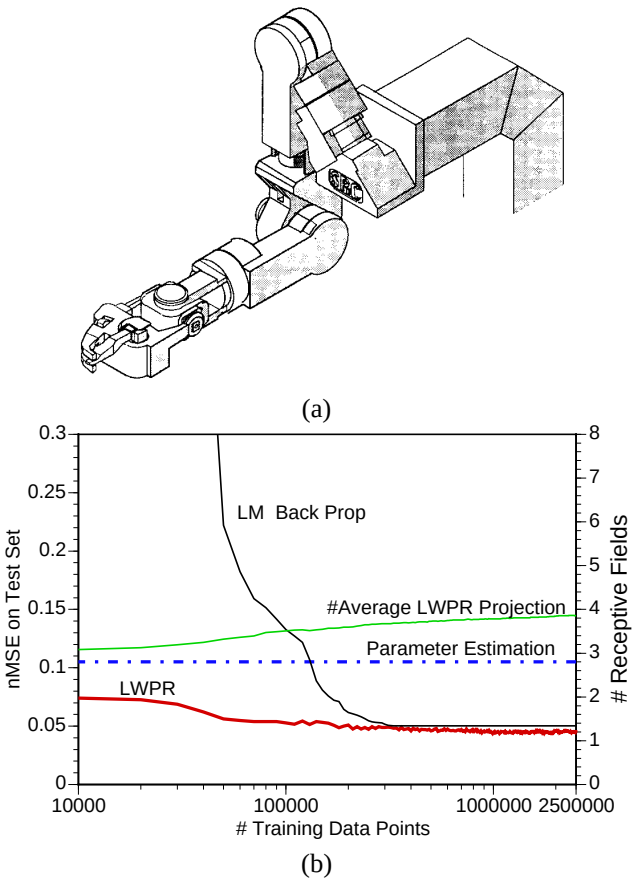


Figure 5. (a) Sketch of the SARCOS dextrous arm (b) Learning curves for 50 dimensional robot dynamics learning

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.