Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'97), July 10-11,1997,Monterey, California, pp.220-225 (1997)

Local Dimensionality Reduction For Locally Weighted Learning

Sethu Vijayakumar[†‡] **and Stefan Schaal**[§‡]

[†]Dept. of Computer Science, Tokyo Institute of Technology, Meguro-ku, Tokyo-152.

sethu@cs.titech.ac.jp, http://ogawa-www.cs.titech.ac.jp/∼sethu.

[§]College of Computing, Georgia Institute of Technology, Atlanta GA 30332-0280

sschaal@cc.gatech.edu, http://www.cc.gatech.edu/fac/Stefan.Schaal.

[‡]ATR Human Information Processing Research Laboratories, Kyoto, Japan 619-02.

February 19, 1997

Abstract

Incremental learning of sensorimotor transformations in high dimensional spaces is one of the basic prerequisites for the success of autonomous robot devices as well as biological movement systems. So far, due to sparsity of data in high dimensional spaces, learning in such settings requires a significant amount of prior knowledge about the learning task, usually provided by a human expert. In this paper we suggest a partial revision of the view. Based on empirical studies, it can been observed that, despite being *globally* high dimensional and sparse, data distributions from physical movement systems are *locally* low dimensional and dense. Under this assumption, we derive a learning algorithm, Locally Adaptive Subspace Regression, that exploits this property by combining a local dimensionality reduction as a preprocessing step with a nonparametric learning technique, locally weighted regression. The usefulness of the algorithm and the validity of its assumptions are illustrated for a synthetic data set and data of the inverse dynamics of an actual 7 degree-of-freedom anthropomorphic robot arm.

1 Introduction

One of the outstanding characteristics of biological systems is their ability to learn, in particular, to learn incrementally in real-time from a multitude of sensory inputs. Despite progress in artificial neural network learning, statistical learning, and machine learning, we are still far away from equipping an artificial system of even moderate complexity with a "black-box" learning system that can perform as autonomously and robustly as the biological counterpart. Among the most basic ingredients that are missing in most learning approaches are three critical components. First, a learning system should possess the ability to learn continually from incrementally arriving data without the danger of forgetting useful knowledge from previously incorporated data, an effect called catastrophic interference. Second, the system has to automatically allocate the appropriate number of resources, e.g., hidden units in a neural network, to represent the learning problem at hand without the undesirable effects of overfitting or oversmoothing. And third, the learning system must be able to deal with a large number of inputs that are possibly redundant or irrelevant.

In this paper we will address these goals in the context of learning sensorimotor transformations, as needed, for instance, in the control of biological or robotic movement systems. From a statistical point of view, this involves approximating a functional relationship $f : R^N \rightarrow R^M$ from N inputs to M outputs. A typical example is to learn the inverse dynamics model of a robot, a highly nonlinear map that relates joint positions, velocities, and accelerations to appropriate joint torques. Such function approximation can be carried out in essentially three different ways. A classical control theoretic approach would model the robot dynamics as accurately as possible based on the system's equations of motion and estimate remaining open parameters in these equation from data collected from the machine, possibly using linear or nonlinear regression techniques [1]. If the equations of motion represent the real physics of the system accurately, this method is going to provide an optimal solution to the learning problem. Assuming no knowledge about the often quite complex structure of the equations of motion, an alternative approach would employ a general nonlinear function approximator. Basis function networks [20] have become a common tool for this purpose, particularly in the form of sigmoidal neural networks and radial basis function networks [10, 12]. However, it remains an open research issue how the

size and structure of these networks should be chosen a priori such that the function approximation capability matches the learning problem, an issue discussed as the bias-variance tradeoff [7]. The third approach, the one that is pursued in this paper, is grounded in a statistical learning framework based on nonparametric statistics. In essence, it assumes that after collecting sufficiently many data points, the prediction of the output for a lookup point (query point) can be accomplished by interpolating over data points that are close to this query point. In the simplest form, this approach becomes a nearest neighbor method where the predicted output equals the output of the nearest data point seen so far [6]. More sophisticated methods use several neighbors to fit a simple parametric model to smoothly interpolate predictions [3]. Function approximation with these nonparametric methods is in the spirit of a Taylor series expansion at the query point. An advantage is that no commitment needs to be made as to how "large" the learning system has to be – the local parametric model is calculated for every query point from scratch using data stored in memory. However, the critical dependency of these methods on an appropriate set of neighbors must be addressed.

In previous work, we developed a learning method that can automatically determine the size and shape of the neighborhood for a nonparametric learning method, Receptive Field Weighted Regression (RFWR), that uses locally weighted linear regression to interpolate the neighboring data [15, 16]. RFWR removed the need to store all the training data in memory by just retaining a sufficient number of locally linear models. It could be shown that this learning system has favorable properties for incremental learning in the spirit of the issues mentioned at the beginning of this section, and it was successfully applied in real-time learning tasks with an anthropomorphic robot arm [17]. However, the computational complexity of RFWR increases more than quadratic with the number of inputs to the system, a fact which confined the application of RFWR to problems of low dimensionality. In addition, nonparametric learning methods that depend on the notion of "neighborhood" generally scale unfavorably to high dimensions. The reason for this behavior comes from the non-intuitive effect that in high dimensional spaces, e.g., 20-dimensional, all data points are approximately the same distance away from each other [18], thus destroying the discriminative power of neighborhood relations.

Given this "curse of dimensionality", nonparametric learning systems – and actually all other general nonlinear learning systems – seem to have limited merits for sensorimotor control. However, when one examines data distributions of high dimensional data sets generated from real physical systems, one often notices that *locally* such data is not high dimensional at all and rarely exceed 5-8 dimensions. It is this observation which motivates the approach suggested in this paper. Despite the fact that our previously developed learning techniques are theoretically able to profit from such low dimensional distributions, they, as mentioned above already, quickly become computationally infeasible and also tend to be numerically less robust. This effect is due to only exploiting the low dimensional distributions *implicitly* by a regularization technique called Ridge Regression [3]. However, if we can exploit the low dimensional distributions explicitly by performing a *local* dimensionality reduction of the data before we apply our nonparametric learning techniques, we should be able to extend nonparametric learning and its favorable incremental learning properties to high dimensional spaces within acceptable computational costs.

To pursue this line of thought, this paper is structured as follows. Section 2.1 will briefly review locally weighted regression (LWR) [4, 2], an algorithm which is the backbone of our function approximation techniques. Section 2.2 discusses local dimensionality reduction and introduces locally weighted principal component analysis (LWPCA). Section 2.3 combines LWR and LWPCA to form our new learning algorithm, Locally Adaptive Sub-Space Regression (LASS). Finally, Section 3 demonstrates the properties of LASS using synthetic data and learning the inverse dynamics model of an actual 7-degree-of-freedom anthropomorphic robot arm.

2 Locally Adaptive Subspace Regression

The assumed underlying statistical model of our problems is the standard regression model $y = f(\mathbf{x}) + \epsilon$, where $\mathbf{x}$ denotes the N dimensional input vector, y, for the sake of clarity, a scalar output, and ϵ the additive mean-zero noise term. LASS consists of an automatically adjusting number of elements, each of which is processing data in the same way. The different stages and additional notation of the information flow of one element in the LASS system are shown in Figure 1. In essence, the input is transformed into a predicted output $\hat{y_k}$ by two linear transformations, $\mathbf{W}_k$ and $\mathbf{b}_k$. $\mathbf{W}_k$ performs the dimensionality reduction and $\mathbf{b}_k$ fits a (hyper)plane to the reduced data. Additionally, a weight w_k is calculated with the help of the connections $\mathbf{D}_k$. The weight, w_k, indicates how much this LASS element should contribute to the total prediction of the entire system. The total prediction $\hat{y}$ results from the weighted average of the individual predictions $\hat{y}_k$ of all the K elements of the LASS system:

$$\hat{y} = (\sum_{k=1}^{K} w_k \hat{y}_k)/(\sum_{k=1}^{K} w_k). \quad (1)$$

In the following section, we will introduce each of the processing stages and explain their notation and the learning rules. We will also drop the subscript k since every LASS elements learns independently of every other one and is updated by the same formulae.

2.1 Locally Weighted Regression

In this section, we focus on how to estimate the regression parameters $\mathbf{b}$ and b_0 which combined form a linear mapping

$$\hat{y} = \mathbf{x}_{reg}^T \mathbf{b} + b_0 = \tilde{\mathbf{x}}^T \beta, \text{ where } \tilde{\mathbf{x}} = (\mathbf{x}_{reg}^T, 1)^T.$$

We will leave the determination of the map, $\mathbf{x}$ to $\mathbf{x}_{reg}$ in Figure 1, to the next section.

Every training data point $(\mathbf{x}, y)$ has a corresponding weight from a Gaussian activation function

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

whose center $\mathbf{c}$ is assigned at the time of creation of the LASS unit (see below). The matrix $\mathbf{D}$ is called a distance metric and determines the size and shape of the "receptive field" created by (2). $\mathbf{D}$ can be learned as shown in [16]. In this version of LASS, however, we will assume that $\mathbf{D}$ has been chosen appropriately beforehand.

A fast and efficient calculation of β can be performed by locally weighted regression [4, 3]. The objective function of locally weighted regression is the minimization of the weighted mean squared error criterion

$$J_1 = \frac{1}{\sum_{i=1}^{p} w_i} \sum_{i=1}^{p} w_i \, \| y_i - \hat{y}_i \|^2. \qquad (3)$$

It will be useful to leave the incremental learning framework for a moment and think in terms of a batch update of β. If we summarize the input part of all p training points in the rows of the matrix $\mathbf{X} = (\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_p)^T$, the corresponding output part in the rows of the vector $\mathbf{y} = (y_1, ..., y_p)^T$, and the corresponding weights in the diagonal matrix $\mathbf{W} = diag(w_1, ..., w_p)$, the regression parameters β can be calculated from a weighted regression:

$$\beta = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y} = \mathbf{P} \mathbf{X}^T \mathbf{W} \mathbf{y} \qquad (4)$$

An incremental update for β is obtained by recursive least squares [9, 16]. It results in *exactly* the same estimation for β as (4) after one sequential sweep through the training data and constitutes a very fast Newton training method with guaranteed convergence to the global minimum of (3). The equations for the incremental update will be given in Section 2.3.

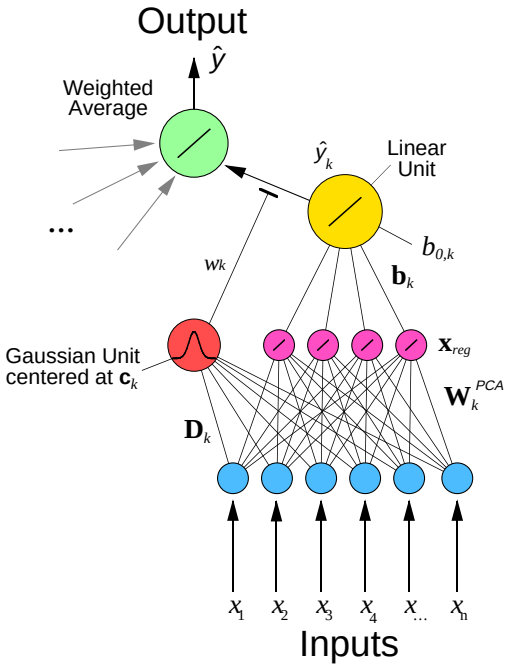


Figure 1: Illustration of the information processing stages of LASS.

2.2 Locally Weighted Principal Component Analysis

The new element to be introduced in the context of local nonparametric learning is an incremental local dimensionality reduction. Various candidate techniques of dimensionality reduction can be considered for this purpose, including principal component analysis (PCA), independent component analysis, and factor analysis [11, 5, 21]. At the time being, PCA seems to offer the best compromise for LASS in terms of computational feasibility and its statistical assumptions.

As the computational complexity of the update of β by recursive least squares is quadratic in the number of inputs and, furthermore, linear regression is vulnerable to redundant input data, the goal of a PCA preprocessing stage is to transform the original N dimensional input $\mathbf{x}$ by means of the matrix $\mathbf{W}^{PCA}$ to as low dimensional a representation as possible:

$$\mathbf{x}_{reg} = \mathbf{W}^{PCA} \mathbf{x}_{mz}, \qquad (5)$$

where $\mathbf{x}_{mz} = \mathbf{x} - \bar{\mathbf{x}}$ denotes the mean subtracted input data. In other words, we want to locally project $\mathbf{x}$ into a L dimensional subspace – a subspace which accounts for the maximal local variance of the input data up to a user defined threshold θ_{PCA}. This processing stage can be accomplished by minimizing a weighted cost criterion in the spirit of Minimum Description Length (MDL) [13]:

$$J_2 = \frac{1}{2} \sum_{i=1}^{p} w_i \| \mathbf{x}_{reconst,i} - \mathbf{x}_{mz,i} \|^2, \qquad (6)$$

The variable $\mathbf{x}_{reconst}$ is calculated from the inverse transformation $\mathbf{x}_{reconst} = \mathbf{W}^{PCA^{-1}}\mathbf{x}_{reg} = \mathbf{W}^{PCA^T}\mathbf{x}_{reg}$, since (7), the update for $\mathbf{W}^{PCA}$, enforces that $\mathbf{W}^{PCA^T}$ is an orthogonal transformation and each of it's row has unit length. The minimization of (6) is achieved by gradient descent with learning rate η

$$W_{ij}^{PCA^{n+1}} = W_{ij}^{PCA^n} + \eta \frac{\delta J_2}{\delta W_{ij}^{PCA}}. \quad (7)$$

where

$$\frac{\delta J_2}{\delta W_{ij}^{PCA}} = w x_{reg,i}(\sum_{r=1}^{i} x_{reg,r} W_{ij}^{PCA^n} - x_{mz,j}).$$

This corresponds to a weighted version of the incremental PCA algorithm of Oja[11] and Sanger[14]. In order to speed up learning, we can also derive a second order gradient descent minimization of (6) based on [19] by assigning an adjustable η_{ij} to each coefficient W_{ij}^{PCA}. η_{ij} is updated according to

$$\eta_{ij}^{n+1} = exp(\tau_{ij}^{n+1}), \quad (8)$$

where τ_{ij}^{n+1} is defined as

$$\tau_{ij}^{n+1} = \tau_{ij}^{n} - \theta \frac{\delta J_2}{\delta W_{ij}^{PCA}} h_{ij}^n.$$

Here, h_{ij}^{n+1} is updated incrementally as

$$h_{ij}^{n+1} = h_{ij}^n [1 - \eta_{ij}^{n+1} \frac{\delta^2 J_2}{\delta W_{ij}^{PCA^2}}]^+ - \eta_{ij}^{n+1} \frac{\delta J_2}{\delta W_{ij}^{PCA}},$$

where

$$[z]^+ = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Computing this second order update adds only little computation. It adjusts the learning rates η_{ij} in geometric steps by gradient descent in the meta parameter τ_{ij} with meta learning rate θ. More details are given in [19]. For the calculation of $\mathbf{x}_{mz}$, it is also necessary to incrementally obtain the weighted mean of $\mathbf{x}$ by

$$\bar{\mathbf{x}}^{n+1} = (W^n \bar{\mathbf{x}}^n + w\mathbf{x})/(W^n + w) \quad (9)$$

where $W^{n+1} = W^n + w$.

2.3 The LASS Algorithm

Combining LWR and LWPCA, we can obtain the LASS algorithm. It should be noted that the LWPCA pre-processing does not only yield a computational advantage in terms of providing a significantly reduced input to the LWR step; LWPCA also decorrelates the dimensions of $\mathbf{x}_{reg}$ such that the matrix $\mathbf{P}$ in (4) becomes diagonal. Thus, the regression step can be decomposed into $L+1$ one dimensional regressions [8] and the incremental update of β becomes linear in the computational complexity with respect to the number of regression inputs:

$$\beta_i^{n+1} = s_{xy,i}^{n+1} / s_{xx,i}^{n+1} \quad (10)$$

where

$$\begin{aligned} s_{xy,i}^{n+1} &= \lambda s_{xy,i}^n + w\tilde{x}_i y \\ s_{xx,i}^{n+1} &= \lambda s_{xx,i}^n + w\tilde{x}_i{}^2 \end{aligned}$$

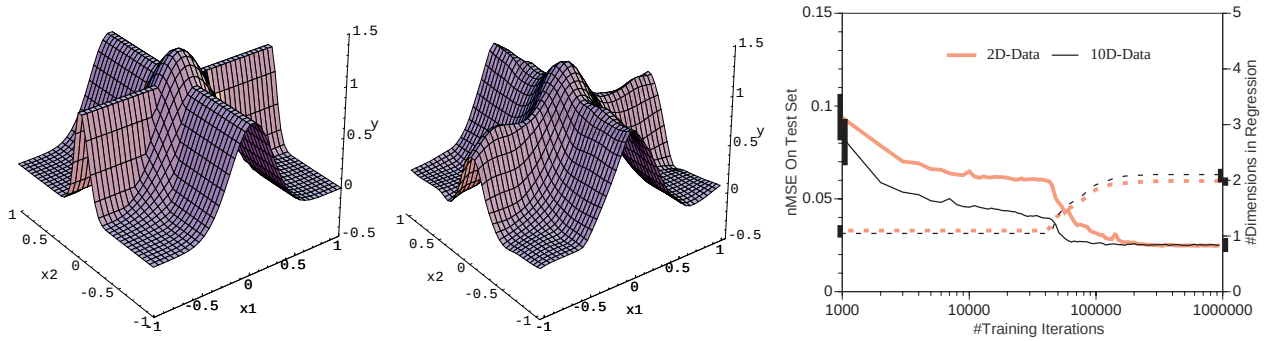
The variable λ denotes a forgetting factor such that initial input to the regression, stemming from a LWPCA which has not properly converged yet, will not negatively influence the regression result in the long run. Applying a forgetting factor is a standard technique in recursive estimation techniques [9].

The LASS algorithm then proceeds as follows. The entire system is initialized with no processing element. Every piece of training data $(\mathbf{x}, y)$ is used to update all the existing elements. If no element is activated (cf. Equation 2) more than a threshold w_{gen}, a new LASS element is created with its receptive field center $\mathbf{c}$ in (2) initialized to $\mathbf{c} = \mathbf{x}$. All other parameters of the new LASS element are initialized to zero, except for $\mathbf{D}$ and $\mathbf{W}^{PCA}$. The distance metric $\mathbf{D}$ is set to a user supplied value $\mathbf{D}_{def}$, while the coefficients W_{ii}^{PCA} are set to "1", and all other coefficients to zero. The initial dimensionality of the LWPCA starts out with $L = 2$, although the regression will only use $L - 1$ inputs. Having one more LWPCA output than used in the regression allows the algorithm to monitor when the dimensionality of the regression stage should be increased, as explained below.

The dimensionality of the regression stage is augmented based on a variance threshold criterion. For this purpose, each LASS unit keeps an incremental record of the variances of its L LWPCA outputs:

$$\mathbf{v}_{PCA}^{n+1} = (\lambda W^n \mathbf{v}_{PCA}^n + w\mathbf{x}_{reg}^2)/(W^n + w) \quad (11)$$

If the condition $\mathbf{v}_{PCA,L} / \sum_{i=1}^{L} \mathbf{v}_{PCA,i} > \theta_{PCA}$ is true, the dimensionality L is incremented by one and appropriate coefficients are added in $\mathbf{W}^{PCA}$ and $\mathbf{b}$. The learning rule (7) guarantees that the variances $\mathbf{v}_{PCA}$ are in descending order[14]. Thus, if the last element, i.e, the L-th element, contains a large percentage of the total variance, it should be added to the regression analysis. To avoid premature adding of dimensions, it is useful to monitor the rate of change of the variances and add dimensions only if the rate of change is close to zero. It is important to note that adding dimensions does not disturb the results obtained by the previously trained LASS parameters. A new dimension in the LWPCA adds a row to $\mathbf{W}^{PCA}$,



(a) (b) (c)

Figure 2: (a) Surface plot of function to be approximated; (b) LASS approximation results for 10-dimensional input data set; (c) $nMSE$(solid lines) and dimensionality of regression(dashed lines) as function of training iterations,averaged over 10 learning trials. One training iteration corresponds to one incremental presentation of a training data. The black vertical bars denote one standard deviation at the beginning and end of learning.

but the learning rule (7) ensures that updates of coefficients of $\mathbf{W}^{PCA}$ are not affected by coefficients whose row index is larger. Thus, the new row is trained entirely independently. Similarly, adding a coefficient to $\mathbf{b}$ just adds a new element to an additive regression. As shown in (10), the regression updates for each dimension are independent of each other due to the decorrelation of $\mathbf{x}_{reg}$ in the LWPCA.

In sum, LASS is a constructive learning algorithm in two different ways. First, LASS elements are added whenever a training point in input space does not sufficiently activate any existent LASS element. This process will guarantee that the entire input distribution of the training data is quickly covered by LASS. Second, within each LASS element the dimensionality of the regression stage can grow until the LWPCA models a user specified fraction of the local variance of the inputs. This feature ensures the quality of the regression result. The adjustable parameters in each LASS element are the LWPCA weights $\mathbf{W}^{PCA}$ and the regression parameters β, both of which are trained with second order learning techniques.

3 Empirical Results

In the first example we will use a synthetic data set that allows to illustrate function fitting results with LASS graphically. The task is to approximate

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

from noisy data drawn incrementally and 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, as shown in Figure 2a. The test data 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. The initial parameters of LASS are set to $\mathbf{D}_{def} = 100\mathbf{I}$ ($\mathbf{I}$ is the identity matrix), $w_{gen} = 0.2$, $\theta_{PCA} = 0.05$. The learning rates were $\theta = 10$, $\eta = 1$, and the forgetting factor was set to $\lambda = 0.9995$.

The initial test consisted of just fitting this function of two variables. In the second test we augmented the input space by 8 additional dimensions whose values were zero, and transformed this input space by a 10 dimensional randomly chosen rotation matrix. Thus, the task of LASS was to recover this low dimensional function now embedded in a high dimensional space. Figure 2b shows a typical example of the reconstruction of the function for the 10 dimensional test after rotating the results back into the original low dimensional space; the approximation results for the 2 dimensional test are very similar to this plot and are not shown separately. Figure 2c illustrates the course of learning for both tests. It takes about 50,000 iterations until the LWPCA converges initially. At this point the dimensionality of the regression starts to increase and saturates at about 2. Simultaneously, the $nMSE$ drops quickly and converges at about $nMSE = 0.025$. Both the low and high dimensional learning show the same qualitative and quantitative behavior. LASS properly detects that in the high dimensional test the actual dimension of the input data is not more than 2. It should also be noted that the $nMSE$ starts at a fairly low value after only 1,000 iterations, despite the system only employing one dimensional regressions at this point. This fast approximation speed is due to the second order learning of LWR. As a baseline compar-

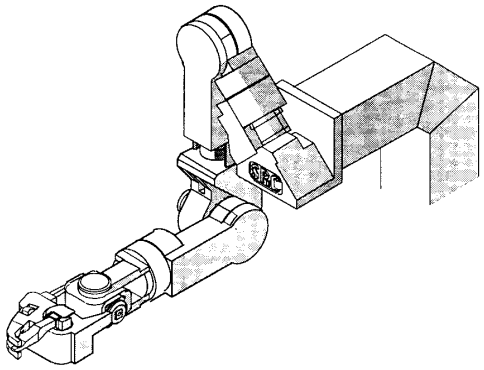


Figure 3: Sketch of Sarcos Dextrous Arm

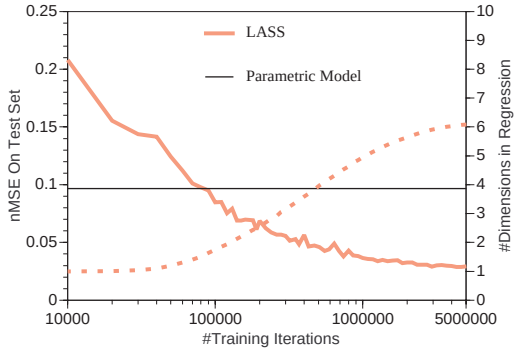


Figure 4: $nMSE$(solid lines) and dimensionality of regression(dashed lines) as function of training iterations.

ison, a sigmoidal neural network requires about 100 hidden units and about 20,000,000 iterations to accomplish a similar function fitting result. In the second example, we approximated the inverse dynamics model of 7-degree-of-freedom anthropomorphic robot arm (Sarcos Dextrous Arm, Figure 3). The input to this model are 7 joint positions, 7 joint velocities, and 7 joint accelerations. The outputs are the corresponding 7 joint torques. The data set consisting of 45,000 data points, collected at 100Hz from the actual robot performing various rhythmic and discrete movement tasks, was randomly split into half to obtain a training set and testing set. Figure 4 shows the learning results in comparison to a parametric estimation technique based on the equations of motion [1]. After about 100,000 iterations (roughly 2 passes through the training set or 15 minutes of real-time data), LASS already accomplished a $nMSE$ (averaged over all 7 output dimensions) as good as the parametric model, despite, on average, only 2 dimensions contributing to the locally weighted regression. The system converges at $nMSE = 0.03$, about 3 times better than the parametric model. An average of 6 dimensions were employed locally to accomplish this, a result that confirms our hypothesis that physical systems have locally low dimensional data distributions. However, care has to be taken in interpreting the better $nMSE$ results of LASS in comparison to the parametric inverse dynamics estimation. Good performance on the $nMSE$ does not necessarily imply good tracking performance, the ultimate test to judge the quality of estimated inverse dynamics models. Such evaluations have yet to be performed.

4 Discussion

The goal of this paper is to emphasize one major point, i.e, learning in high dimensional spaces may not be as complicated and as previously thought. The rationale for this statement is based on the assumption that data distributions, despite being *globally* high dimensional, are *locally* often of only low dimensional structure. Based on this assumption, we developed a nonparametric learning algorithm which is targeted to make use of such locally low dimensional distributions. Our learning system, Locally Adaptive SubSpace regression (LASS), preprocesses data by a local principle component analysis (LWPCA) before handing it to a locally weighted regression analysis (LWR). Learning is thus accomplished by fitting low dimensional hyperplanes to the data. The entire function is finally approximated in a piecewise linear fashion. For a synthetic and an actual data set, we illustrated that LASS achieved the expected performance: in both cases, locally low dimensional data distributions were detected and exploited appropriately. In contrast to our previously developed learning methods whose computational complexity is more than quadratic in the number of input dimensions, LASS scales linearly with the number of inputs.

In future work, we will address a missing component in the learning algorithm, the automatic adjustment of the local region in which a locally linear model is valid. Indeed, the way LASS has been designed already provides all the statistical values to perform such computations. Besides numerical stability, one of the major reasons to break up the information flow of the LASS elements (cf. Figure 1) into two linear transformation was based on the necessity to derive higher order statistics as shown in Equation (11). Another open point of research concerns how the regression analysis could influence the LWPCA. At the current stage of LASS, LWPCA proceeds independently of LWR, which, from a statistical point of view, is not satisfying as the quality of the regression depends on the distribution of the input data ([16]). Empirical evaluations will provide insight into how much this shortcoming affects the quality of learning. Performing LWPCA in joint data space could offer a possible way out. To explore all these issues, LASS will be further developed and explored in learning various sensorimotor transformations on our anthropomorphic robot.

Acknowledgements

Support for S.Vijayakumar and S.Schaal was partially provided by the ATR Human Information Processing Research Laboratories. This work was funded in parts by grants to S.Vijayakumar provided by the Japanese Ministry of Education, Science and Culture (Monbusho) and to S.Schaal provided by the German Research Association, the Alexander von Humboldt Foundation and the German Scholarship Foundation.

References

[1] An,C.H., Atkeson.C.G. & Hollerbach.J.M., *Model based control of a robot manipulator*, MIT Press,Cambridge,MA,1988.

[2] Atkeson.C.G., "Using local models to control movement", In:Touretzky,D.(Eds.), *Advances in Neural Information Processing Systems 1,* San Mateo,CA:Morgan Kauffman (1989).

[3] Atkeson,C.G., Moore,A.W. & Schaal,S.(in press), "Locally weighted learning", *Artificial Intelligence Review.*

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

[5] Deco,G. & Obradovic,D., *An Information Theoretic Approach to Neural Computing*, Springer-Verlag New York, (1996).

[6] Duda,R.O. & Hart,P.E., *Pattern Classification and Scene Analysis*, New York:Wiley (1973).

[7] Geman,S., Bienenstock,E. & Doursat,R., "Neural networks and the bias-variance dilemma", *Neural Computation*, No.4, pp.1-58 (1992).

[8] Hastie,T.J. & Tibshirani,R.J., *Generalized additive models*, London:Chapman-Hall (1990).

[9] Ljung,L. & Soederstroem,T., *Theory and Practice of Recursive Identification*, Cambridge,MIT Press (1986). .

[10] Moody,J. & Darken,C., "Learning with localized receptive fields", In: Touretzsky,D., Hinton,G.,& Sejnowsk,T.(Eds.) *Proceedings of the 1988 Connectionist Summer School*, pp.133-143 (1988).

[11] Oja,E., "A simplified neuron model as a principal component analyser", *Journal of Mathematical Biology* Vol.15, pp.267-273 (1982)

[12] Poggio,R. & Girosi,F, "Regularization algorithms for learning that are equivalent to multilayer networks" *Science* Vol.247 (1990).

[13] Rissanen,J., *Stochastic complexity in statistical enquiry*, Singapore:World Scientific (1989).

[14] Sanger,T.D., "Optimal unsupervised learning in a single layer linear feedforward neural network", *Neural Networks*, Vol.2, pp.459-473 (1989).

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

[16] Schaal,S. & Atkeson,C.G., "Receptive field weighted regression" *Technical Report TR-H-209, ATR Human Information Processing Labs., Kyoto 619-02, Japan* (1997)

[17] Schaal,S., "Learning from demonstration", *Advances in Neural Information Processing Systems 9* (in press).

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

[19] Sutton,R.S., "Adapting bias by gradient descent: An incremental version of Delta-Bar-Delta" *Proc. Tenth National Conf. Artificial Intelligence* pp.171-176 (1992).

[20] Vijayakumar,S. & Ogawa,H., "A functional analytic approach to incremental learning in optimally generalizing neural networks", *Proc. IEEE Intl. Conf. Neural Networks, Perth, Australia*, pp.777-782 (1995)

[21] Witten,I.H., Neal, R.M. & Cleary,J.G., "Arithmetic coding for data compression", *Communications of the ACM*, Vol.30, pp.520-540 (1997).