Vijayakumar, S, Schaal, S (2000). Fast and efficient incremental learning for high-dimensional movement systems, International Conference on Robotics and Automation (ICRA2000). San Francisco, April 2000.

Fast and Efficient Incremental Learning for High-dimensional Movement Systems

Sethu Vijayakumar *⊕

sethu@brain.riken.go.jp
http://www.islab.brain.riken.go.jp/~sethu

Stefan Schaal ‡⊕

sschaal@usc.edu
http://www-slab.usc.edu/sschaal

*Laboratory for Information Synthesis, Riken Brain Science Research Institute, Wako, Saitama, Japan
‡Computer Science and Neuroscience, HNB-103, Univ. of Southern California, Los Angeles, CA 90089-2520
⊕Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan

***Abstract**: We introduce a new algorithm, Locally Weighted Projection Regression (LWPR), for incremental real-time learning of nonlinear functions, as particularly useful for problems of autonomous real-time robot control that requires internal models of dynamics, kinematics, or other functions. At its core, LWPR uses locally linear models, spanned by a small number of univariate regressions in selected directions in input space, to achieve piecewise linear function approximation. The most outstanding properties of LWPR are that it i) learns rapidly with second order learning methods based on incremental training, ii) uses statistically sound stochastic cross validation to learn iii) adjusts its local weighting kernels based on only local information to avoid interference problems, iv) has a computational complexity that is linear in the number of inputs, and v) can deal with a large number of—possibly redundant and/or irrelevant—inputs, as shown in evaluations with up to 50 dimensional data sets for learning the inverse dynamics of an anthropomorphic robot arm. To our knowledge, this is the first incremental neural network learning method to combine all these properties and that is well suited for complex on-line learning problems in robotics.*

1 Introduction

Motor control of complex movement systems requires knowledge of a variety of continuous valued functions, for instance coordinate transformations of the manipulator kinematics and models of the forward or inverse dynamics. Whenever analytical methods are not available to derive these functions, e.g., as frequently the case in lightweighted and complex (humanoid) dexterous robots (e.g., Figure 1), learning approaches need to be employed to find approximate solutions. However, function approximation for high dimensional nonlinear motor systems 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 less than quadratic in the number of input dimensions, and, of course, achieve accurate function approximation and generalization. In this paper, we suggest to accomplish these goals with techniques of projection regression. The key idea of projection regression is to cope with the complexities of high dimensional function approximation by decomposing the regression into a sequence of one-dimensional localized regressions along particular directions in input space. The major difficulty of projection regression becomes how to select efficient projections, i.e., to achieve the best fitting result with as few as possible one-dimensional regression.

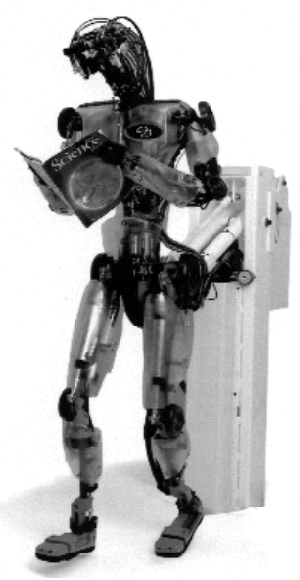

Figure 1: Humanoid robot in our laboratory

Previous work in the learning literature has focussed on finding good global projections for fitting nonlinear one-dimensional functions. Among the best known algorithms is projection pursuit regression ([1]), and its generalization in form of Generalized Additive Models ([2]). 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 [3]). Here we suggest an alternative method of projection regression, focussing on finding efficient *local* projections. Local projections can be used to accomplish local function approximation in the neighborhood of a query point. Such methods allow fitting 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 ([4], [5]). Counterintuitive to the curse of dimensionality ([6]), local regression methods can work successfully in high dimensional spaces ([7]), as we will empirically demonstrate below.

In the next section, we will first motivate why function approximation in high dimensions is complicated, and why there is hope that the "curse of dimensionality" is not really a problem for high-dimensional movement systems. Second, we will introduce our new learning algorithm, Locally Weighted Projection Regression (LWPR) that can efficiently deal with high-dimensional learning problems. In the last section, we will show learning results on synthetic data and data from a Sarcos Dexterous Robot, an anthropomorphic robot arm, whose inverse dynamics model was learned by our algorithm, even when contaminated with irrelevant and redundant inputs.

2 The Curse of Dimensionality

Both in research of biological and robotic motor control, the need for internal models has been emphasized in order to achieve accurate control of fast movements ([8]; [9]). For models with only few input dimensions, learning approaches have been quite successful (e.g, [10]; [11]; [12]). However, for higher dimensional learning tasks, it has been unclear whether learning approaches can succeed. For instance, we are interested in learning the inverse dynamics model of a humanoid robot (Figure 1). The robot has 30 degrees-of-freedom (DOFs), amounting to a 90-dimensional input space to the inverse dynamics function (30 position, 30 velocity, and 30 acceleration states). A rigid body dynamics model performs poorly for this hydraulically actuated light-weighted system.

Depending on the nonlinear activation function employed in the network units, two categories of neural networks are available for such a task. The traditional sigmoidal activation function—or any other function with unbounded support—tries to find global projections in input space that lead to a good approximation of the data. It is well known that these networks learn rather slowly in high-dimensional spaces, and that the network structure, i.e., the number of hidden units, needs to be chosen carefully to achieve good learning results. Moreover, neural networks with unbounded activation functions are very vulnerable to inference problems, i.e., the unlearning of relevant knowledge when trained on new data points ([13]). Since an autonomous robot in a dynamic environment will encounter new data all the time, complex off-line re-training procedures would need to be devised to cope with the interference problem, and a substantial amount of data would have to be stored for the re-training. From a practical point, this approach is hardly useful.

Alternatively, networks types can be selected that have bounded support in their activation functions. Radial basis function networks with Gaussian kernels are among the best known in this category ([14]; [15]). The selection of the number of hidden units is usually easier with this network type, and training proceeds rather fast and has reduced interference problems. However, in a 90-dimensional input space, it is possible that a huge number of radial basis functions would be needed to cover the input space: even if each input dimension is only covered by 2 Gaussians, an astronomical number of 2^{90} Gaussians would be required for our humanoid.[1]

Therefore, the only hope for learning approaches is that the data generated by a movement system actually lie on low dimensional distributions. We tested this hypothesis empirically by collecting data from human and robot (Sarcos Dexterous Arm, Figure 4) seven degree-of-freedom unconstrained arm movements by recording about 100 Mb of data of the joint angular trajectories during various movement tasks. For the human data, we assumed a biomechanically reasonable mass distribution and computed the torque trajectories of every DOF; for the robot data, load sensors measured the torques directly. Thus, we obtained appropriate data for training a supervised neural network. The network we chose employed Gaussian kernels as nonlinear activation function that included a principle component dimensionality reduction (PCA) in each kernel ([7]). Importantly, the PCA automatically determined the number of local dimensions needed in each kernel to accomplish good function approximation. Results of incremental learning with this system are shown in Figure 2. Besides that the neural network achieved very good approximation results, characterized by a low normalized mean squared error (c.f. [16]), it most noteworthy that the network only required on average 4 to 6 dimensions locally for good function fitting (cf. the dashed lines in Figure 2).

[1] This example is only to highlight the problem of function fitting in high-dimensional spaces—it will never be possible to collect enough data with a real robot to fill such big spaces, even when running the robot for hundreds of years.

These analyses confirmed our assumption that movement data in high-dimensional spaces actually lie on locally low dimensional distributions. Appropriate learning algorithms that can discover the local distributions should thus be able to approximate really complex internal models.

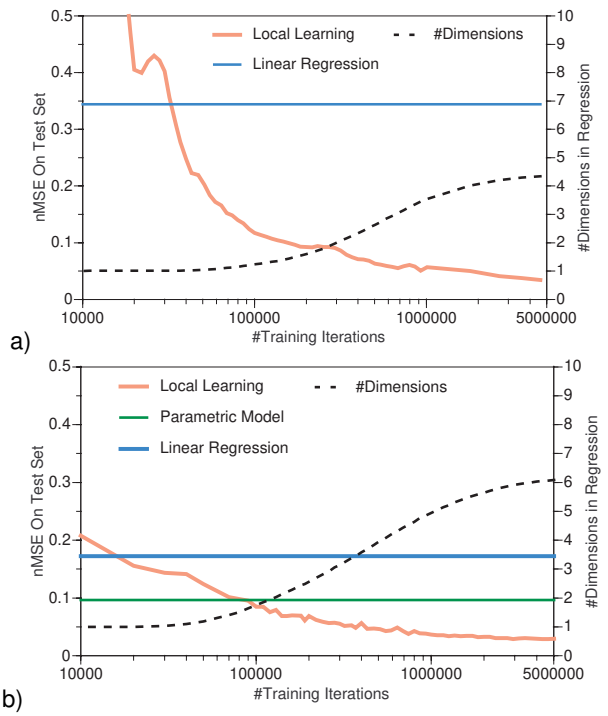

Figure 2: Learning inverse dynamics models with a nonparametric neural network that uses Gaussian activation functions with principal component-based dimensionality reduction in each Gaussian. The "local learning" curves show the reduction of the mean squared error as a function of the training iterations. The "linear regression" straight line is the result of a global linear regression fit of the training data, and the "Parametric Model" line demonstrates the results from using a rigid-body based parameter estimation method to fit the inverse dynamics model ([10])—the latter method was only applicable for the robot data. a) Learning from human behavioral data, b) learning from robot data.

A drawback of the PCA-based local dimensionality reduction of ([7]), however, is that PCA only reduces the dimensionality by looking at the *input* data. In this way, PCA can mistake noise with relevant input signals, and irrelevant dimensions, as they may occur in some learning problems, will influence the results of learning. These considerations lead to a new learning network, as described in the next section.

3 Locally Weighted Projection Regression

In the following, we assume that the data generating model for our regression problem 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 the output is one-dimensional. The key concept of our regression network is to approximate nonlinear functions by means of piecewise linear models. The region of validity, called a *receptive field*, of each linear model is computed from a Gaussian function:

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

where $\mathbf{c}_k$ is the center of the k^{th} linear model, and $\mathbf{D}_k$ corresponds to a distance metric that determines the size and shape of region of validity of the linear model. 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:

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

Previous work ([13]) 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 $O(n^2)$ computations which became infeasible for about more than 10 dimensional input spaces. Here we suggest reducing the computational burden in each local linear model by applying a sequence of one-dimensional regressions along selected projections $\mathbf{u}_r$ in input space (note that we dropped the index k from now on unless it is necessary to distinguish explicitly between different linear models):

$$\begin{aligned} &\text{Initialize: } y = \beta_0, \mathbf{z} = \mathbf{x} - \mathbf{x}_0 \\ &\quad \text{For i = 1:r} \\ &\quad\quad s = \mathbf{u}_i^T \mathbf{z} \\ &\quad\quad y = y + \beta_i s \\ &\quad\quad \mathbf{z} \leftarrow \mathbf{z} - \mathbf{p}_i s \end{aligned} \quad (3)$$

The projections $\mathbf{u}_i$, the univariate regression parameters β_i, the mean $\mathbf{x}_0$, and the number of projections r are determined by the learning algorithm. Additionally, the learning algorithm also finds a projection vector $\mathbf{p}_i$ that reduces the input space for the next univariate regression. As will be explained below, this step allows to find more efficient projections $\mathbf{u}_i$ at subsequent univariate regression steps.

In order to determine the open parameters in Equation (3), the technique of partial least squares (PLS) regression can be adapted from the statistics literature ([17]). The important ingredient of PLS is to choose projections according to the correlation of the input data with the output data. The following algorithm, Locally Weighted Projection Regression (LWPR), uses an incremental locally weighted version of PLS to determine the linear model parameters:

***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: $\mathbf{z} = \mathbf{x}, res = y - \beta_0^{n+1}$

For i = 1: r,

a) $\mathbf{u}_i^{n+1} = \lambda\,\mathbf{u}_i^n + w\,\mathbf{z}\,res$ (4)

b) $s = \mathbf{z}^T \mathbf{u}_i^{n+1}$

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

d) $SR_i^{n+1} = \lambda\,SR_i^n + w\,s\,res$

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) $\mathbf{z} \leftarrow \mathbf{z} - s\mathbf{p}_i^{n+1}$

i) $res \leftarrow res - s\beta_i^{n+1}$

j) $MSE_i^{n+1} = \lambda\,MSE_i^n + w\,res^2$

In the above equations, $\lambda \in [0,1]$ is a forgetting factor that determines how much older data in the regression parameters will be forgotten, similar as in recursive system identification techniques ([18]). 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 $\mathbf{p}_i$ 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 $\mathbf{u}_{i+1}$ that is orthogonal to $\mathbf{u}_i$. Thus, for r=n, the entire input space would be spanned by the projections $\mathbf{u}_i$ and the regression results would be identical to that of a traditional linear regression. Step j) will be discussed below.

There are several important properties in PLS. First, if all the input variables are statistically independent, PLS will find the optimal projection direction $\mathbf{u}_i$ in a *single* iteration—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.

The above update rule can be embedded in an incremental learning system that automatically allocates new locally linear models as needed ([13]):

```
Initialize the LWPR with no receptive field (RF);
For every new training sample (x,y):
   For k=1 to #RF:
      calculate the activation from (1)
      update according to (4)
   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;
```

In this pseudo-code algorithm, w_{gen} is a threshold that determines when to create a new receptive field, and $\mathbf{D}_{\text{def}}$ is the initial (usually diagonal) distance metric in (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 (4). If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, i.e.,

$$\frac{MSE_{i+1}}{MSE_i} > \phi, \text{ where } \phi \in [0,1] \qquad (6)$$

the algorithm will stop adding new projections to the local model.

It is even possible to learn the correct parameters for the distance metric $\mathbf{D}$ in each local model. The algorithm for this update was derived in ([13]) for normal locally linear regression based on an incremental cross validation technique. This algorithm is directly applicable to LWPR, and is strongly simplified, as it only needs to be done in the context of univariate regressions. Due to space limitations, we will not provide the update rules in this paper as they can be derived from ([13]).

4 Empirical Evaluations

In order to provide a graphical illustration of the learning algorithm, as a first test, we ran LWPR on 500 noisy training data drawn from the synthetic two dimensional function:

$$z = \max\left\{e^{-10x^2}, e^{-50y^2}, 1.25\,e^{-5\left(x^2+y^2\right)}\right\} + N(0, 0.01) \qquad (7)$$

shown in Figure 3a. 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 Figure 3. 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. Figure 3b illustrates the reconstruction of the original function from the 20-dimensional test—an almost perfect approximation. The rising thin lines in Figure 3c 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 appropriate for this originally two dimensional data set.

In the second evaluation, we approximated the inverse dynamics model of a 7-degree-of-freedom anthropomorphic robot arm (Figure 4a) 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 the 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 the shoulder 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. Figure 4b shows the learning results in comparison to a global linear regression of the data. From the very beginning, LWPR outperformed the global linear regression. Within about 500,000 training points, 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 325 models. This increase is due to the adjustment of the distance metric **D** in Equation (1), which 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. In comparison to the robot learning results in Figure 2, it is noteworthy that LWPR required much fewer local dimensions than the PCA-based algorithm in ([7]). This is due to the fact the LWPR chooses projections much more efficiently than PCA, e.g., for independent input data distributions, only one projection would suffice, as mentioned before.

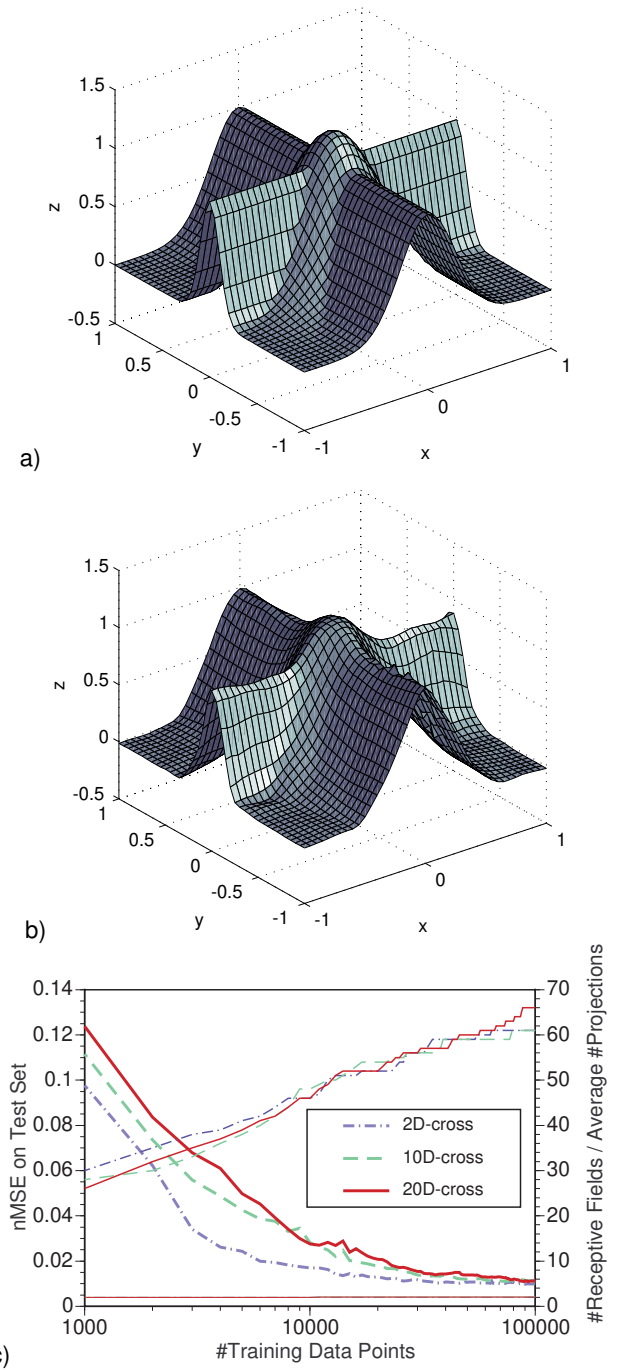

Figure 3: a) Target and b) learned nonlinear cross function. c) Learning curves for 2-D, 10-D, and 20-D data.

5 Conclusions

This paper presented a new learning algorithm, Locally Weighted Projection Regression (LWPR), a nonlinear function approximation network that is particularly suited for problems of on-line incremental motor learning. The essence of LWPR is to achieve function approximation

with piecewise linear models by finding efficient local projection to reduce the dimensionality of the input space. High-dimensional learning problems can thus be dealt with efficiently: 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.. Moreover, the mechanisms of LWPR to select low dimensional projections are capable of excluding irrelevant and redundant dimensions from the input data. As an example, we demonstrated how LWPR leads to excellent function approximation results in up to 50 dimensional data sets, extracted from a 7 degree-of-freedom anthropomorphic robot arm. To our knowledge, this the first incremental learning system that can efficiently work in high dimensional spaces.

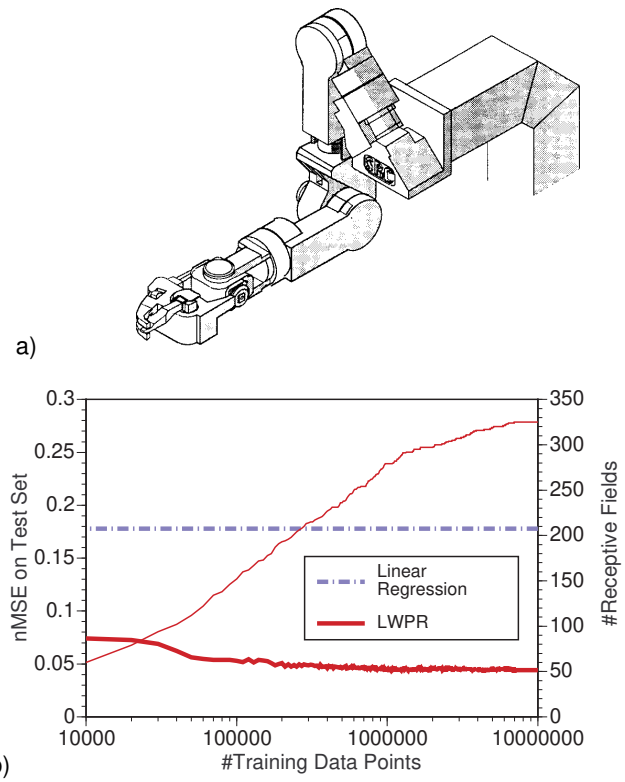

Figure 4: a) Sarcos Dexterous Robot Arm; b) Learning curve for learning the inverse dynamics model of the robot from a 50 dimensional data set that included 29 irrelevant dimensions.

6 Acknowledgments

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

7 References

[1] J. H. Friedman and W. Stützle, "Projection pursuit regression," *Journal of the American Statistical Association, Theory and Models*, vol. 76, pp. 817-823, 1981.

[2] T. J. Hastie and R. J. Tibshirani, *Generalized additive models*. London: Chapman and Hall, 1990.

[3] S. E. L. C. Fahlman, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems II*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524-532.

[4] T. Hastie and C. Loader, "Local regression: Automatic kernel carpentry," *Statistical Science*, vol. 8, pp. 120-143, 1993.

[5] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, pp. 11-73, 1997.

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

[7] S. Vijayakumar and S. Schaal, "Local adaptive subspace regression," *Neural Processing Letters*, vol. 7, pp. 139-149, 1998.

[8] M. Kawato, "Computational schemes and neural network models for formation and control of multijoint arm trajectory," in *Neural Networks for Control*, W. T. Miller Iii, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press, 1990, pp. 197-228.

[9] M. I. Jordan, "Computational aspects of motor control and motor learning," in *Handbook of perception and action*, H. Heuer and S. W. Keele, Eds. New York: Academic Press, 1996.

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

[11] A. H. Fagg, N. Sitkoff, A. G. Barto, and J. C. Houk, "Cerebellar learning for control of a two-link arm in muscle space," presented at IEEE International Conference on Robotics and Automation (ICRA'97), Albuquerque, NM, 1997.

[12] M. Cannon and J. E. Slotine, "Space-frequency localized basis function networks for nonlinear system estimation and control," *Neurocomputing*, vol. 9, pp. 719-726, 1995.

[13] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, pp. 2047-2084, 1998.

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

[15] R. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, pp. 213-225, 1990.

[16] C. M. Bishop, *Neural networks for pattern recognition*. New York: Oxford University Press, 1995.

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

[18] L. Ljung and T. Söderström, *Theory and practice of recursive identification*: Cambridge MIT Press, 1986.