Proc. of Seventeenth International Conference on Machine Learning (ICML2000), Stanford, pp.191-198 (2000)

On-line Learning for Humanoid Robot Systems

Jörg Conradt CONRADT@CLMC.USC.EDU
Gaurav Tevatia TEVATIA@USC.EDU
Sethu Vijayakumar SETHU@USC.EDU
Stefan Schaal SSCHAAL@USC.EDU

Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089-2520 USA
Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02, Kyoto JAPAN

Abstract

Humanoid robots are high-dimensional movement systems for which analytical system identification and control methods are insufficient due to unknown nonlinearities in the system structure. As a way out, supervised learning methods can be employed to create model-based nonlinear controllers which use functions in the control loop that are estimated by learning algorithms. However, internal models for humanoid systems are rather high-dimensional such that conventional learning algorithms would suffer from slow learning speed, catastrophic interference, and the curse of dimensionality. In this paper we explore a new statistical learning algorithm, locally weighted projection regression (LWPR), for learning internal models in real-time. LWPR is a nonparametric spatially localized learning system that employs the less familiar technique of partial least squares regression to represent functional relationships in a piecewise linear fashion. The algorithm can work successfully in very high dimensional spaces and detect irrelevant and redundant inputs while only requiring a computational complexity that is linear in the number of input dimensions. We demonstrate the application of the algorithm in learning two classical internal models of robot control, the inverse kinematics and the inverse dynamics of an actual seven degree-of-freedom anthropomorphic robot arm. For both examples, LWPR can achieve excellent real-time learning results from less than one hour of actual training data.

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.

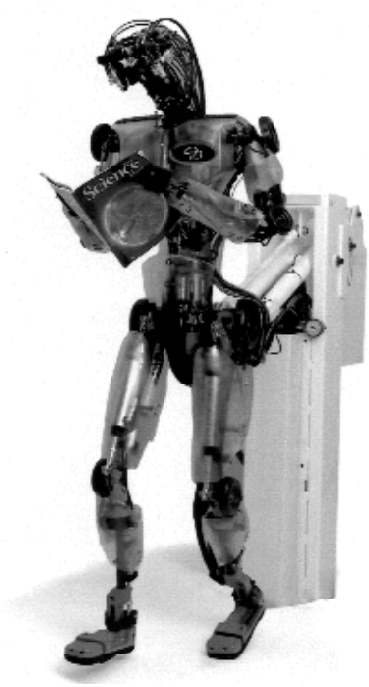

Figure 1: Humanoid robot in our laboratory

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 regressions.

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 (Friedman & Stützle, 1981), and its generalization in the 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, 1990). 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 (Hastie & Loader, 1993, Atkeson, Moore, and Schaal, 1997). Counterintuitive to the curse of dimensionality (Scott, 1992), local regression methods can work successfully in high dimensional spaces (Vijayakumar & Schaal, 1998), as we will empirically demonstrate below. The justification for this statement comes from empirical investigations of movement data generated by human subjects and humanoid robots (Vijayakumar & Schaal, 2000a). This data demonstrated that actual physical systems generate *locally* low dimensional data distributions, despite that the data is high dimensional when viewed from a global perspective. Thus, an algorithm capable of exploiting such locally low dimensional distributions is able to avoid the curse of dimensionality.

In the next section, we will introduce our new learning algorithm, Locally Weighted Projection Regression (LWPR) that can efficiently deal with high-dimensional learning problems. Afterwards, we will describe learning results from applying LWPR to high dimensional learning tasks of classical internal model learning problems in robotics, the inverse kinematics and inverse dynamics problem. LWPR will be shown to master these learning tasks with excellent accuracy while achieving learning speed and computational complexity that make it suitable for real-time implementations on state of the art computing hardware.

2. Locally Weighted Projection Regression

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

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

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:

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

Previous work (Schaal & Atkeson, 1998) 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 per learning iteration which became infeasible for more than about 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 will drop the index k from now on unless it is necessary to distinguish explicitly between different linear models):

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

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 explained below, this step allows finding more efficient projections $\mathbf{u}_i$ at subsequent univariate regressions.

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 (Wold, 1975). 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) $\mathbf{u}_i^{n+1} = \lambda\,\mathbf{u}_i^n + w\,\mathbf{z}\,res$

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

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

d) $SR_i^{n+1} = \lambda\,SR_i^n + w\,s\,res$ (4)

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 (Ljung & Söderström, 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 $\mathbf{p}_i$ 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. All recursive variables in (4) (i.e., those with a superscript *n*) are initialized to zero before the algorithm is started.

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. Readers are referred to Vijayakumar and Schaal (2000b) for a detailed description of the PLS algorithm which we defer here due to space considerations.

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

Table 1. Pseudocode of the LWPR algorithm

```
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.,

$$\frac{MSE_{i+1}}{MSE_i} > \phi, \text{ where } \phi \in [0,1] \qquad (5)$$

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 (Schaal & Atkeson, 1998) for normal locally linear regression based on an incremental cross validation technique. This algorithm is directly applicable to LWPR, and it is strongly simplified since updates are only needed in the context of univariate regressions. A gradient descent update rule for $\mathbf{D}$ can be calculated from minimizing the cost function in (6). Note that gradient descent must be performed in the Cholesky-decomposed distance metric, $\mathbf{M}$ in order to guarantee that $\mathbf{D}$ remains positive definite.

***Gradient descent update of* D:**

$\mathbf{D} = \mathbf{M}^T\mathbf{M}$, where $\mathbf{M}$ is upper triangular

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \qquad (6)$$

$$J = \frac{1}{W}\sum_{i=1}^{p} w_j \left(y_i - \hat{y}_{i,-i}\right)^2 + \gamma \sum_{i,j=1}^{n} D_{ij}^2$$

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

The key statistical ingredient of (6) is the penalized leave-one-out cross validation error, denoted by the "*i,-i*" subscript. The penalty term was motivated in Schaal and Atkeson (1998) and can be shown to lead to a local function approximation technique that estimates the Hessian of the function to be approximated in order to determine the region of validity of the linear model. Employing the PRESS residual error (Myers, 1990), leave-one-out cross validation for linear systems can be reformulated such that eliminating data from the data set can be avoided; this formulation also holds for locally linear models (Schaal & Atkeson, 1994). Since the PRESS residual error does not exist for PLS, equation (6) approximates it by the sum of PRESS residual errors for every projection k using the residual error, $res_{k+1,i}$ from this stage and the projected data point $s_{k,i}$. The vector $\mathbf{s}_k$ contains all $s_{k,i}$, and the matrix $\mathbf{W}$ is a diagonal matrix will all the weights w_i of all p data points. Although the cost in (6) looks like it requires batch updates for learning, a stochastic incremental update can be derived, too. Since the derivation of the incremental update follows exactly the development in Schaal and Atkeson (1998), we omit discussions and update formulae due to space constraints.

3. Empirical Evaluations with an Anthropomorphic Robot

In the following section we will demonstrate the applicability of LWPR to two classical problems in robot learning, the learning of an inverse dynamics and an inverse kinematics model. Data for these learning problems were generated from a seven degree-of-freedom anthropomorphic robot arm, a Sarcos Dexterous Master Arm (Figure 2). This robot arm can serve as both an autonomous robot and an exoskeleton for force reflecting teleoperation. The robot is actuated by 3000psi hydraulic motors and resembles human performance in its dexterity and movement speed.

The benchmark evaluations for the proposed algorithm and comparisons to other state-of-the-art techniques have been carried out in Vijayakumar and Schaal (2000b) Instead, here we look at performance indexes for actual tracking robot implementations.

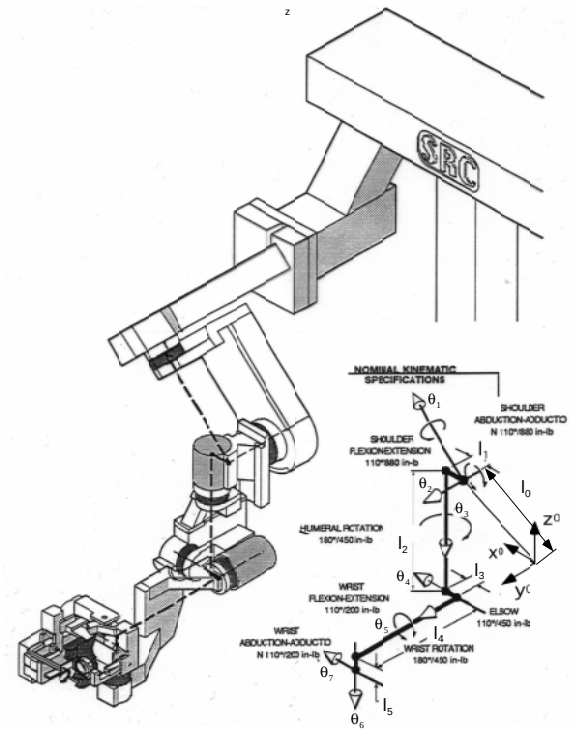

Figure 2: Sarcos Dexterous Master Robot Arm with explanations of the names of the degrees of freedom in the inset on the bottom right.

3.1 Inverse Dynamics Learning

Inverse dynamics models are a typical element of nonlinear control (e.g., Slotine & Li, 1991). The inverse dynamics relates the vectors of position θ, velocity $\dot{\theta}$, and acceleration $\ddot{\theta}$ of a robot to the torque vector τ that is necessary to accomplish the acceleration in the given state (i.e., position and velocity). Ideal mechanical system obey inverse dynamics equations from rigid body dynamics of the form

$$\mathbf{H}\left(\ddot{\theta}\right) + \mathbf{C}\left(\dot{\theta}, \theta\right) + \mathbf{G}(\theta) = \tau \qquad (7)$$

which are highly nonlinear equations that would extend over 1500 lines of C-code for the Sarcos Master Arm. Nevertheless, rigid body dynamic systems can still be dealt with analytically with specialized recursive mathematics packages, and the open parameters in (7) (not shown), composed of the inertial and geometric quantities of the robot, can be estimated from data (An, Atkeson, & Hollerbach, 1988, Featherstone, 1987). Given that the Sarcos Robot is hydraulically actuated, several unknown strong nonlinearities are added to the complexity of (7), stemming from fluid dynamics, nonlinear friction terms of the hydraulic motors, and nonlinear saturation of the actuators. From previous work, we already experienced that

rigid body dynamics is an insufficient model for the Sarcos Robot.

The goal of the experimental evaluation was to approximate the inverse dynamics with LWPR and compare it against parametric estimation techniques based on rigid body dynamics (An et al., 1988). For this purpose, seven LWPR networks were trained, one for each torque motor of the robot. The input to each network was composed of 21 variables, i.e., seven joint positions, velocities, and accelerations. 54,000 data vectors were generated from the robot at 50Hz from randomized smooth rhythmic movements. The robot has position sensors in all degrees-of-freedom (DOF) which allow deriving velocities and accelerations by numerical differentiation. Torque values can be measured from load cells in each joint.

Parametric estimation was carried out using standard estimation methods (An et al., 1988) on the entire data set. This estimation resulted in an analytical rigid body dynamics model with estimated inertial and geometric parameters—this model incorporates a large amount of prior knowledge about the physics of robot systems.

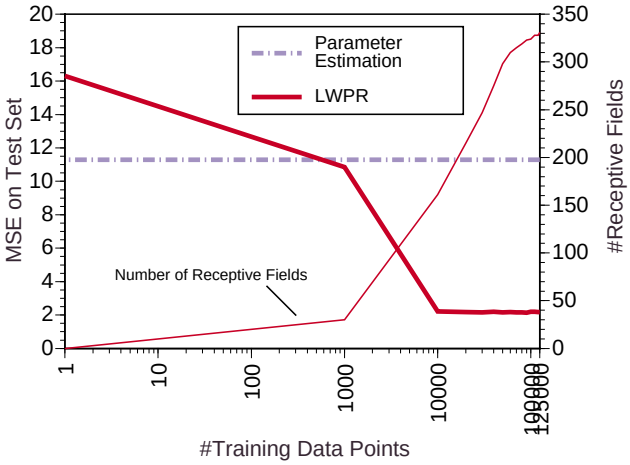

Figure 3: Learning curve of inverse dynamics learning for the elbow joint of the robot.

For LWPR learning, a random subset of 10% of the collected data was excluded to serve as a test set. Each of the seven network was then trained for 125,000 incremental updates from data randomly drawn from the training set—this corresponds to approximately 20 minutes of actual movement time of the robot. Typical learning results for one of the degrees-of-freedom, the "elbow-flexion-extension" joint, are shown in Figure 3. This figure demonstrates that within a few hundred iterations, LWPR outperformed the approximation quality of the parametric approximation technique and converged to a mean squared error (MSE) that is about 5 times lower. About 350 receptive fields were allocated during learning. On average, only 2.6 dimensions are used in a receptive field, demonstrating that partial least squares chose projection direction remarkably well in this example. A comparison of the MSE of the other joints against parametric estimation is given in Table 2. Except for the shoulder joints (SFE, SAA), LWPR learning was always superior to rigid body parameter estimation. Training the network for 125,000 iterations took about 45 minutes on a 500MHz personal computer, indicating that LWPR achieved approximately real-time updating given that the data was collected at 50Hz.

Table 2. Comparison of MSE between LWPR and Parameter Estimation based on rigid body dynamics.

JOINT NAME	PARAMETER ESTIMATION	LWPR
SFE	10.12	10.70
SAA	17.89	13.21
HR	8.74	2.57
EB	11.30	2.22
WR	7.56	0.17
WFE	6.91	0.09
WAA	9.43	0.02

3.2 Inverse Kinematics problem

One of the core issues of robot control is movement planning. Most movement tasks are defined in coordinate systems that are different from the actuator space of the robot. Hence, a coordinate transformation from task to actuator space must be performed before motor commands can be computed. On a system with redundant degrees-of-freedom (DOFs), this transformation from external plans to internal coordinates is often ill posed and is known as the classic inverse kinematics problem. If we define the intrinsic coordinates of a manipulator as the n-dimensional vector of joint angles $\boldsymbol{\theta} \in \Re^n$, and the position and orientation of the manipulator's end effector as the m-dimensional vector $\mathbf{x} \in \Re^m$, the forward kinematic function can generally be written as:

$$\mathbf{x} = \mathrm{f}(\boldsymbol{\theta})$$

while what we need is the inverse relationship :

$$\boldsymbol{\theta} = \mathrm{f}^{-1}(\mathbf{x})$$

For redundant manipulators, like our Sarcos robot, solutions to the above equation are usually non-unique and multiple solutions can exist. Traditional inverse kinematics algorithms address how to determine a particular solution in face of multiple solutions by optimizing additional optimization criterion. These approaches favor local methods that compute an optimal change in $\boldsymbol{\theta}, \Delta\boldsymbol{\theta}$, for a small change in $\mathbf{x}, \Delta\mathbf{x}$ and then integrate $\Delta\boldsymbol{\theta}$ to generate the entire joint space path. Resolved Motion Rate Control (RMRC) is one such local methods which uses the Jacobian $\mathbf{J}$ of the forward kinematics to describe a change of the endeffector's position as:

$$\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \quad (8)$$

This equation can be solved for $\dot{\boldsymbol{\theta}}$ by taking the inverse of $\mathbf{J}$ if it is square i.e. $m=n$, and non-singular, or by using pseudo-inverse computations.

3.2.1 MOTIVATION FOR LEARNING

Traditional pseudo-inverse methods suffer from the problems of singular postures and the need for additional optimization criterion to resolve the redundancy of the system. At singular postures, i.e. when the Jacobian becomes rank deficient these methods cannot yield a solution and suffer from numerical explosions. Due to these problems we suggest a learning scheme to learn this inverse mapping function: a learning system can only reproduce solutions it was trained on, i.e., problems with singularities of the Jacobian $\mathbf{J}$ cannot arise since it is impossible to encounter "singular training data." However, due to the redundancy of the robot arm, learning the inverse problems is usually not possible if the redundant solutions $\boldsymbol{\theta}$ for one $\dot{\mathbf{x}}$ form a non-convex set (Jordan & Rumelhart, 1992). This problem can be avoided by a specific input representation to the learning network. Consider two solutions for $\dot{\boldsymbol{\theta}}$ from the set of solution vectors that produce the same endeffector velocity :

$$\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_1$$

$$\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_2$$

Since the Jacobian relates the $\mathbf{x}$ and $\dot{\boldsymbol{\theta}}$ in linear form, even for a redundant system the average of the two solutions will result in the desired $\dot{\mathbf{x}}$, i.e.,:

$$\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})(\dot{\boldsymbol{\theta}}_1 + \dot{\boldsymbol{\theta}}_2)/2 = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}_{AVE} \quad (9)$$

Thus, if one considers only a small local region of the $\boldsymbol{\theta}$ space, the set of solutions of joint velocity vectors for one particular $\dot{\mathbf{x}}$ form a convex set. Averaging over this local convex set—this is what neural network learning essentially does—will lead to a valid solution to the inverse kinematics problem as essentially proofed in (9). Thus we propose to learn the inverse mapping function with a spatially localized learning system, i.e., LWPR, based on the input/output representation

$$(\dot{\mathbf{x}}, \boldsymbol{\theta}) \rightarrow (\dot{\boldsymbol{\theta}})$$

This approach will automatically resolve the redundancy problem without resorting to any other optimization approach; the inverse solution picked is simply the local average over solutions that were experienced. The algorithm will also perform well near singular posture since, as mentioned before, it cannot generate joint movement that it has never experienced.

3.2.2 LEARNING THE INVERSE KINEMATICS MODEL

In order to apply LWPR to inverse kinematics learning, we learn a separate model for each of the joint angles, such that each of the models is a 7+3 to 1 mapping:

$$(\dot{\mathbf{x}}, \boldsymbol{\theta}) \rightarrow (\dot{\theta}_L), \text{ L = 1....7}$$

Since it is almost impossible to explore the complete 10-dimensional space by random movements, we created a small default joint velocity for the arm that was used in areas where no receptive fields had been generated yet:

$$\dot{\theta}_D = \alpha(\theta_R - \theta)\sqrt{\dot{\mathbf{x}}'\dot{\mathbf{x}}}$$

where θ_R denotes a "rest-angle" for each joint, usually chosen to be the mid-point of the possible range of motion. The advantage of this approach is that it avoids the need to separate a training and an execution phase since the exploratory movement from this default movement automatically generates sufficient training data to improve performance. Additionally, the default movement biases the exploration towards generating reasonable distributions that will help to resolve the robot's redundancy in a "natural" way (i.e., by avoiding extreme postures).

In applying LWPR, the coefficients of the distance metric in (1) that corresponded to the direction of the $\mathbf{x}$ input were set to rather small values, incorporating our prior knowledge that spatial localization was only necessary in $\boldsymbol{\theta}$ (cf. Equation (9)). Additionally, in order to ensure fast learning, we re-scaled the $\dot{\mathbf{x}}$ input dimensions with a large multiplier such that PLS would prefer projections in the subspace spanned by the $\dot{\mathbf{x}}$ data. This modification expressed our domain knowledge that the $\boldsymbol{\theta}$ inputs are only needed for spatial localization, but not for the regression (cf. Equation (9)).

3.2.3 EXPERIMENTAL RESULTS

The robot arm was required to track a ball, created from simulated visual input, that moved in a pseudo-random smooth pattern in Cartesian space, generated by superimposing sinusoids of various amplitudes and frequencies. Among the presented training patterns was also the figure eight, drawn in the x-z plane in Figure 4. At the beginning of learning, when almost no data had been experienced, the tracking is inaccurate. As shown in the right part of Figure 4, the robot improves very quickly if the same pattern is presented for some time. After about half an hour of learning, trajectories of arbitrary complexity can be traced with excellent accuracy in various parts of the workspace. The resolution of redundancy, entirely dominated by the distributions of training data, resulted in a natural looking, human-like arm posture during the tracking. We are not aware of any previous work in machine learning that accomplished the learning of inverse kinematics functions for redundant manipulators in a comparable way.

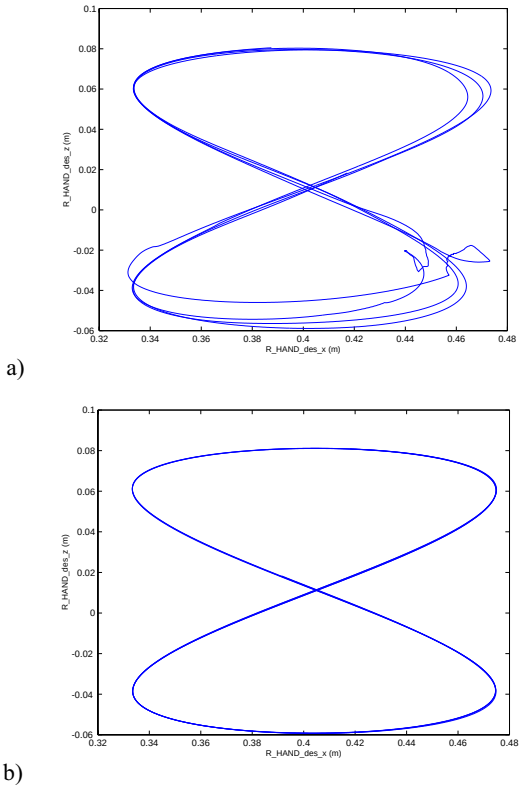

Figure 4: Tracking a figure eight with learned inverse kinematics. a): performance during the first 20 seconds of learning, b) Performance after one minute of training.

4. 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 projections 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 2-4 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 on classical problems of robot learning, the inverse dynamics and inverse kinematics problem. For inverse dynamics learning, LWPR outperformed traditional model identification methods by a large margin. This result by itself is not surprising since it was known in advance that our robot data did not conform to the traditional rigid body assumption. What was remarkable, however, was that LWPR achieved good learning results from data that represented less than 20 minutes of actual robot movement, and that only 2-3 local projections were needed on average. Thus, the 21-dimensional input space of the learning task was drastically reduced.

The second evaluation concerned learning the inverse kinematics for a redundant robot arm. We demonstrated how a change in input representation allows learning a normally ill-posed learning problem, and how LWPR could be biased to accommodate the domain knowledge of this task. Learning of LWPR could be applied on-line and lead to very rapid convergence of the robot performance to accurate redundancy resolution.

We are currently working on creating a complete real-time implementation of these two robot learning tasks in a multi-processor environment. All our benchmark tests indicated that it will be unproblematic to apply LWPR in real-time. To our knowledge, no other research groups have accomplished similar fast and accurate learning results in such high-dimensional learning problems.

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.

References

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

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

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

Featherstone, R. (1987*). Robot dynamics algorithms.* Kluwer Academic Publishers.

Friedman, J.H., & Stützle, W. (1981). *Projection pursuit regression.* Journal of the American Statistical Association, Theory and Models 76:817-823.

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

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

Jordan, I.M., & Rumelhart (1992). *Supervised learning with a distal teacher*. Cognitive Science, pp 307-354.

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

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

Schaal, S., & Atkeson, C.G. (1994). *Assessing the quality of learned local models.* In J. Cowan, G. Tesauro, & J. Alspector (Eds.), Advances in Neural Information Processing Systems 6. San Mateo, CA: Morgan Kaufmann, pp. 160-167.

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

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

Slotine, J.J.E., & Li, W. (1991). *Applied nonlinear control. Englewood Cliffs*. NJ: Prentice Hall.

Vijayakumar, & S., Schaal, S. (1998). *Local adaptive subspace regression.* Neural Processing Letters 7:139-149.

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

Vijayakumar, S., & Schaal, S. (2000 b). Locally Weighted Projection Regression: An O(*n*) Algorithm for Incremental Learning in High Dimensional Space. *Proceedings of the Seventeenth International Conference on Machine Learning (in press)*. San Francisco (2000).

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