Real-Time Statistical Learning For Robotics and Human Augmentation

Stefan Schaal, Sethu Vijayakumar, Aaron D'Souza, Auke Ijspeert and Jun Nakanishi

Computational Learning and Motor Control Laboratory
Computer Science and Neuroscience, HNB-103, Univ. of Southern California, Los Angeles, CA 90089
Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan
{sschaal,sethu,dsouza,ljspeert,nakanisi}@usc.edu
http://www-clmc.usc.edu

***Abstract**: Real-time modeling of complex nonlinear dynamic processes has become increasingly important in various areas of robotics and human augmentation. To address such problems, we have been developing special statistical learning methods that meet the demands of on-line learning, in particular the need for low computational complexity, rapid learning, and scalability to high-dimensional spaces. In this paper, we introduce a novel algorithm that possesses all the necessary properties by combining methods from probabilistic and nonparametric learning. We demonstrate the applicability of our methods for three different applications in humanoid robotics, i.e., the on-line learning of a full-body inverse dynamics model, an inverse kinematics model, and imitation learning. The latter application will also introduce a novel method to shape attractor landscapes of dynamical system by means of statistical learning.*

1 Introduction

An increasing number of problems in robotics and human augmentation involve real-time modeling of complex high-dimensional processes. Typical examples include the on-line modeling of dynamic objects observed by visual surveillance for improved tracking and recognition, user modeling for advanced computer interfaces, learning of value functions for reinforcement learning, and learning of control policies and internal models for adaptive control of complex dynamical systems, for instance, as needed for humanoid robots and autonomous airplane control. When approaching such learning problems, there are many alternative learning methods that can be chosen, either from the neural network, the statistical, or the machine learning literature. However, the current focus in learning research lies more on algorithms for the *off-line* analysis of *finite* data sets, without too severe constraints on the computational complexity of the algorithms. Examples of such algorithms include the revival of Bayesian inference ([1], [2]) and the new algorithms developed in the framework of structural risk minimization ([3], [4]). Mostly, these methods target problems in *classification* and *diagnostics*, although several extensions to regression problems exist (e.g., [5]).

In *on-line* modeling, however, special constraints need to be taken into account. Most learning problems require *regression* networks, for instance, as in the learning of internal models, coordinate transformations, control policies, or evaluation functions. Data in on-line modeling is usually not limited to a finite data set—sensors keep on producing new data that should be included in the learning system immediately. Thus, computationally inexpensive training methods are important in this domain, and incremental learning is mandatory. Among the most significant additional problems of on-line learning is that the distributions of the learning data may change continuously. Input distributions change due to the fact that the same dynamic process may work around different setpoints at different times, thus creating different kinds of training data. Moreover, the input-output relationship of the data—the conditional distribution—may change if the dynamic process is nonstationary or learning involves nonstationary training data as in reinforcement learning or error-based adaptive control. Such changing distributions easily lead to catastrophic interference in many neural network paradigms, i.e., the unlearning of useful information when training on new data ([6]). As a last element, learning tasks described above can be rather high dimensional in the number of input dimensions, thus amplifying the need for efficient learning algorithms. The

current trend in learning research is largely orthogonal to the problems of on-line modeling.

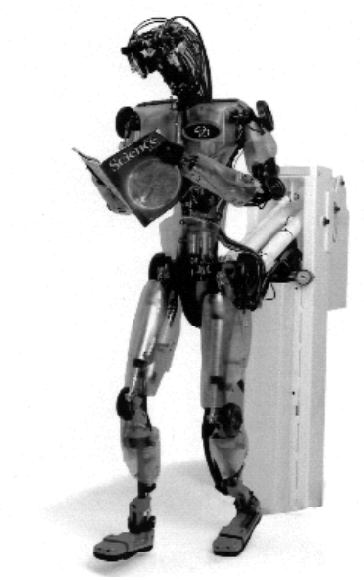

Figure 1: 30 Degree-of-freedom humanoid robot in our laboratory.

Over the past years, we have been developing statistical learning methods that are particularly well suited for on-line learning in robotics and human augmentation. These techniques are generally classified as locally weighted learning (LWL) ([7], [8]) as they emphasize statistical inference solely based on a carefully selected set of spatial neighbors around a point of interest. In the most recent advancements, we succeeded in combining locally weighted learning ([9], [10], [11],) with probabilistic learning to achieve a novel form of on-line learning with piecewise linear models that excels in its computational simplicity and statistical soundness. In this paper, we will first outline our learning system, and subsequently demonstrate its applicability to several different domains, including the on-line acquisition of an inverse dynamics model, an inverse kinematics transformation, and learning movement imitation, all conducted on our 30DOFs humanoid robot (Figure 1). The application to movement imitation will also introduce a novel approach to directly shape attractor landscape of nonlinear dynamical systems by means of statistical learning.

2 Probabilistic LWL

In our learning algorithms we assume that the data generating model for our regression problems has the standard form $y = f(\mathbf{x}) + \varepsilon$, where $\mathbf{x} \in \Re^d$ is a d-dimensional input vector, the noise term ε has mean zero, $E\{\varepsilon\} = 0$, and the output is one-dimensional. The key concept of our LWL methods is to approximate nonlinear functions by means of piecewise linear models ([12]), similar to a first-order Taylor series expansion. Locally linear models have been demonstrated to be an excellent statistical compromise among the possible local polynomials that can be fit to data ([13]). The scientific problem in LWL is to determine the region of validity in which a local model can be trusted, and how to fit the local model in this region robustly.

We compute the region of validity, called a *receptive field*, of each linear model from a Gaussian kernel:

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

where $\mathbf{c}_k$ is the center of the k^{th} linear model, and $\mathbf{D}_k$ corresponds to a positive semi-definite distance metric that determines the size and shape of region of validity of the linear model. Other kernel functions are possible ([7]) but add only minor differences to the quality of function fitting. The most straightforward development of our probabilistic LWL algorithm is by reviewing memory-based Locally Weighted Regression (LWR) ([14]), as summarized in the following pseudo-code:

***The LWR Algorithm*:**

Given: a query point $\mathbf{x}_q$

p training points $\{\mathbf{x}_i, y_i\}$ in memory

Compute Prediction**:**

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

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

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

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

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

c) compute locally linear model

$\beta = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{Y}$

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

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

β_{d+1} denotes the $(d+1)^{\text{th}}$ element of the regression vector β. The parameters of the distance

metric **D** need to be optimized in computationally expensive off-line cross validation ([15]).

The problems of LWR for on-line learning are easily visible from the pseudo-code. First, the off-line optimization of **D** is inappropriate, second, the processing of all data in memory for a lookup becomes computationally increasingly infeasible with large amounts of data, and third, the matrix inversion in step c) of the pseudo-code can be numerically unstable and computationally too complex in high-dimensional input spaces. In the following, we will first combat the latter problem and then show that the other two points are subsequently easily dealt with.

2.1 EM-based Projection Regression

Despite being one of the most successful tools in statistical data analysis, linear regression suffers from large computational complexity and numerical problems of matrix inversion in high dimensional input spaces. Several methods have been suggested to remedy this problem, including principle component regression, ridge regression, factor analysis, and partial least squares regression, all of which can easily be modified in the context of local linear models ([16]). Among these methods, one of the most interesting is partial least squares regression (PLS) ([17]). In PLS, the input data is first projected onto a one-dimensional projection in input space, and subsequently a one-dimensional regression is performed along this direction. The projection direction is chosen to be the vector of maximal correlation between input and output data. If this algorithm is applied recursively using the residual error of the previous projection as target for the next projection, surprisingly accurate and numerically robust function approximation can be achieved ([18]). The number of adequate projection is usually determined heuristically or by off-line cross validation.

Using the idea of projection regression in PLS as an inspiration, one can ask whether there is an *optimal* projection direction to simplify linear regression. The answer is easy: the gradient of a linear function is the optimal projection direction, and linear regression is the way to obtain it—but, as mentioned above, this method is too expensive and numerically brittle. Using a probabilistic formulation of projection regression, however, we can derive a new incremental algorithm for linear regression that finds the optimal projection direction cheaply without numerical problems.

For every data point **x**, let us introduce a new vector valued variable **z** in linear regression such that

$$z_m = \beta_m x_{mi} \quad \text{where} \quad y = \sum_{m=1}^{d} z_m \tag{3}$$

Thus, each coefficient of **z** can be interpreted as the contribution that the corresponding x coefficient has to the output *y* after x was projected onto β. If for every data point **x** the corresponding **z** were known, the entire linear regression could be decomposed simply into *n* univariate linear regressions, a process that is $O(d)$, i.e., only of linear computational complexity in the number of inputs *d*, compared to $O(d^3)$ of regular linear regression. Despite that the introduction of an unknown quantity **z** may look useless at the first glance, there exists an efficient algorithm to estimate it statistically by means of an Expectation-Maximization (EM) algorithm ([19]). For this purpose, the learning approach needs to be reformulated as a maximum likelihood estimation problem where our goal is to maximize:

$$\log p(\mathbf{y} \mid \mathbf{X}) = \log \int p(\mathbf{y}, \mathbf{Z} \mid \mathbf{X})\, d\mathbf{Z} \tag{4}$$

i.e., an optimization function in which we integrate our the unknown quantities **Z**, a matrix of the same dimensionality and coefficient indexing as defined for **X** in the LWR algorithm above. Such a problem formulation allows a standard application of EM after assuming normal distributions for all random variables, i.e., $z_m \sim N(\beta_m x_m, \psi_{z,m})$ and $y \sim N(\mathbf{1}^T \mathbf{z}, \psi_y)$. Omitting derivations due to space limitations, the resulting iterative procedure for maximum likelihood estimation of all unknown parameters becomes the Probabilistic Partial Least Squares (PPLS) regression algorithm outlined below in (5). In this algorithm, we introduce $\langle \cdot \rangle$ as the statistical expectation operator that is taken with respect to the posterior distribution $p(\mathbf{Z} \mid \mathbf{X}, \mathbf{y})$ (e.g., see [20]). By iterating between the E-step and M-step of (5) until the cost (4) is approximately converged, exactly the same results for β can be obtained as in ordinary least squares solutions.

Probabilistic Partial Least Squares (*PPLS*)**:**

Iterate until convergence:

M - Step:

$$\beta_m = \frac{\sum_{i=1}^{p} \langle z_{im} \rangle x_{im}}{\sum_{i=1}^{p} x_{im}^2}$$

$$\psi_y = \frac{1}{p}\left[\sum_{i=1}^{p} y_i^2 - 2\mathbf{1}^T \sum_{i=1}^{p} \langle \mathbf{z}_i \rangle y_i + \mathbf{1}^T \left(\sum_{i=1}^{p} \langle \mathbf{z}_i \mathbf{z}_i^T \rangle \right) \mathbf{1}\right]$$

$$\psi_{z,m} = \frac{1}{p}\left[\sum_{i=1}^{p} \langle z_{im}^2 \rangle - 2\beta_m \sum_{i=1}^{p} \langle z_{im} \rangle x_{im} + \beta_m^2 \sum_{i=1}^{p} x_{im}^2\right] \quad (5)$$

E - Step:

$$\mathbf{1}^T \langle \mathbf{z}_i \rangle = \frac{1}{s} y_i \sum_{m=1}^{d} \psi_{z,m} + \left(1 - \frac{1}{s}\sum_{m=1}^{d} \psi_{z,m}\right) \beta^T \mathbf{x}_i$$

$$\mathbf{1}^T \langle \mathbf{z}_i \mathbf{z}_i^T \rangle \mathbf{1} = \sum_{m=1}^{d} \psi_{z,m} - \left(\frac{1}{s}\sum_{m=1}^{d} \psi_{z,m}\right)^2 + \left(\mathbf{1}^T \langle \mathbf{z}_i \rangle\right)^2$$

$$\langle z_{im} \rangle = \beta_m x_{im} + \frac{1}{s} \psi_{z,m} \left(y_i - \beta^T \mathbf{x}_i\right)$$

$$\langle z_{im}^2 \rangle = \psi_{z,m} - \frac{1}{s}\psi_{z,m}^2 + \langle z_{im} \rangle^2$$

$$s = \psi_y + \sum_{m=1}^{d} \psi_{z,m}$$

Two properties of PPLS are remarkable. First, one iteration of EM is computationally linear in the number of inputs *d* and the algorithms converges usually within a moderate number of iterations—this allows PPLS to operate in really high dimensional spaces (we tested up to 100,000 inputs in preliminary work). Second, the update equation for β in (5) simply corresponds to univariate linear regression between each input dimension and the expected value of the corresponding coefficient of **z**—exactly what we had hoped for at the beginning of this section. As a last point, PPLS is easily modified to become a fully Bayesian estimation technique that guards against overfitting, and it is equally straightforward to include the weighting factor in (1) for spatially localizing PPLS.

2.2 Learning The Distance Metric D

As we demonstrated in previous work ([21],[6]), local linear models fitted by least squares allow deriving statistically robust methods to adjust the coefficients of the distance metric **D** in (1) by means of on-line gradient descent techniques that mimic leave-one-out crossvalidation:

Gradient descent update of* D*:

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

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

$$J = \frac{1}{W}\sum_{i=1}^{p}\sum_{k=1}^{d} \frac{\frac{w_i}{s}\psi_k \left(y_i - \beta^T \mathbf{x}_i\right)}{\left(1 - w_i \frac{x_{k,i}^2}{\sum_{j=1}^{p} w_j x_{k,j}^2}\right)^2} + \gamma \sum_{i,j=1}^{d} D_{ij}^2 \quad (7)$$

The cost function in (7) was derived in ([6]). It corresponds to a weighted mean squared error, corrected by a term in the denominator that ensures good generalization properties of learning in the spirit of leave-one-out crossvalidation. In high dimensions, **D** is usually assumed to be a diagonal matrix to reduce computational complexity.

2.3 Incremental Updating

The batch updates of the preceding two sections can be reformulated as recursive update equations where all sums over training data in (5) is accumulated incrementally, usually by employing a forgetting factor ([22]).

2.4 The Final Algorithm

The above learning rules can be embedded in an incremental learning system—Locally Weighted Probabilistic Projection Regression (LWPPR)—that automatically allocates new locally linear models as needed ([6]):

Initialize LWPPR 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 (5) and (7)
  end;
  If no linear model was activated by more than w_{gen};
    create a new RF with **c**=**x**, **D**=$\mathbf{D}_{def}$
  end;
end;

In this pseudo-code algorithm, w_{gen} is a threshold that determines when to create a new receptive field, and $\mathbf{D}_{\text{def}}$ is the initial (usually diagonal) distance metric.

The prediction for a query point is then formed as the weighted average of the predictions of all local models:

$$\hat{y}_q = \frac{\sum_{k=1}^{K} w_k \hat{y}_{q,k}}{\sum_{k=1}^{K} w_k} \quad (8)$$

3 Empirical Evaluations

In the following sections, we will provide several examples of applying our learning methods to on-line learning with our humanoid robot (Figure 1).

3.1 Inverse Dynamics Learning

The goal of this learning task was to approximate the inverse dynamics model of our 30-degree-of-freedom humanoid robot by on-line learning while tracking a figure-8 pattern in Cartesian space ([23]). The input data consisted of 90 dimensions: 30 joint positions, velocities, and accelerations. The goal of learning was to approximate the appropriate feedforward torque command of all hydraulic robot motor in response to the input vector. The inverse model was part of a computed torque controller, distributed over 4 parallel processors in a VME bus, running the real-time operating system vxWorks ([24]).

The LWPPR models were trained on-line while the robot performed a pseudo randomly drifting figure-8 pattern in front of its body. Lookup proceeded at 480Hz, while updating the learning model was achieved at about 70Hz. At certain intervals, learning was stopped and the robot attempted to draw a planar figure-8 at 2Hz frequency for the entire pattern. The quality of these drawing patterns is illustrated in Figure 2. In Figure 2a, x_des denotes the desired figure-8 pattern, x_sim illustrates the figure-8 performed by our robot simulator that uses a perfect inverse dynamics model (but not necessarily a perfect tracking and numerical integration algorithm), x_param is the performance of the estimated rigid body dynamics model ([25]), and x_lwpr shows the results of LWPPR. While the rigid body model has the worst performance, LWPPR obtained the results comparable to the simulator.

Figure 2b illustrates the speed of LWPPR learning. The x_nouff trace demonstrates the figure-8 patterns performed without any inverse dynamics model, just using a low gain PD controller. The other traces show how rapidly LWPPR learned the figure-8 pattern during training: they denote performance after 10, 20, 30, and 60 *seconds* of training. After 60 seconds, the figure-8 is hardly distinguishable from the desired trace.

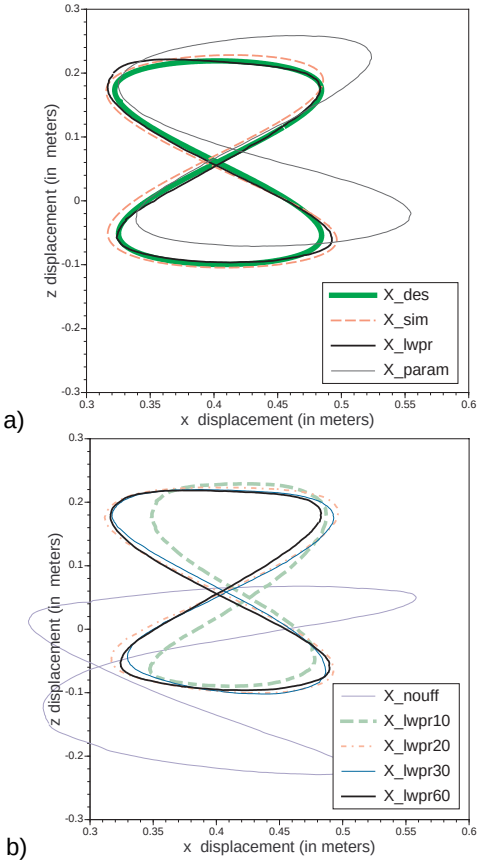

Figure 2: On-line earning of tracking a figure-8 with LWPPR. a Comparison of learning controller with a perfect simulation (sim) and a controller using an estimated rigid body dynamics model (param), b) Progress of tracking performance over 60 seconds of learning.

3.2 Inverse Kinematics Learning

Learning of inverse kinematics is useful when the kinematic model of a robot is not accurately available, when Cartesian information is provided in uncalibrated camera coordinates, or when the computational complexity of analytical solutions becomes too high. For instance, in our humanoid robot we observed that offsets in sensor readings and inaccurate knowledge of the exact kinematics of the robot can lead to significant error accumulations for analytical inverse kinematics computations, and that it is hard to maintain an accurate calibration of the active vision system. Instead of re-calibrating the entire system frequently, we would rather

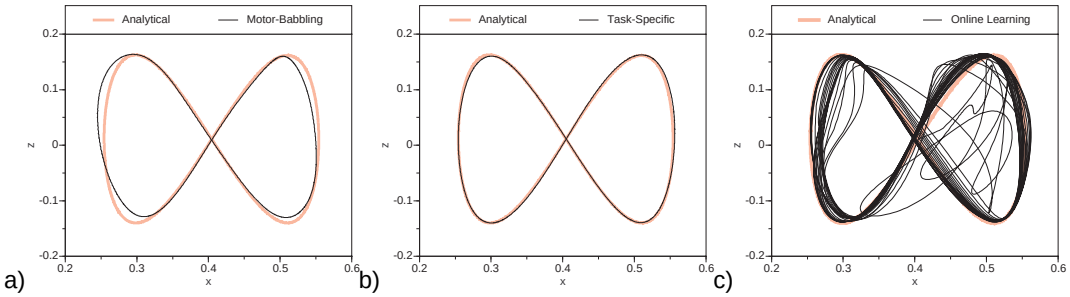

Figure 3: Inverse kinematics learning: a) performance of after being trained on data collected from pseudo-random sinusoidal movement, b) performance after 60 seconds of additional training on the figure-8, c) time course of learning.

employ a self-calibrating, i.e., learning approach. An additional appealing feature of learning inverse kinematics is that it avoids problems due to kinematic singularities—learning works out of experienced data, and such data is always physically correct and does not demand impossible postures as can result from an ill-conditioned matrix inversion.

We applied LWPPR to learning a model for resolved motion rate control for inverse kinematics ([26]). A principled resolution of the high level of redundancy can be achieved with a simple trick: we use LWPPR to acquire the direct kinematics $\mathbf{x} = f(\boldsymbol{\theta})$, and within each local model also learn the inverse mapping $\dot{\mathbf{x}} \rightarrow \dot{\boldsymbol{\theta}}$. It can be shown that this spatially localized inversion in velocity space gives a well-defined solution to the inverse kinematics problem ([27]), and that it is also possible to include a kinematic optimization criterion to bias the local inverse solution to a particular solution ([26]).

Figure 3a shows the performance of inverse kinematics learning for the right endeffector of the robot after approximately 10 minutes of training on pseudo-random sinusoidal arm movement data. The figure-8 test pattern demonstrates that the robot accomplished decent, but not quite accurate performance. Training the robot directly on the figure-8 test patter for one more minute results in highly accurate tracking (Figure 3b). The final experiment in Figure 3c started with an untrained system, and learn the inverse kinematics from scratch, while performing the figure-eight task itself. Figure 3c shows the progression of the system's performance from the beginning of the task to about 3 minutes into the learning. One can see that the system initially starts out making slow inaccurate movements. As it collects data, however, it rapidly converges towards the desired trajectory. Within a few more minutes of training on the task, the performance approached that seen in Figure 3b.

3.3 Imitation Learning

We also applied LWPPR in the context of imitation learning of point-to-point movements ([28]). For this purpose, we developed a novel representation of movement plans in form of nonlinear dynamical system theory:

$$
\begin{aligned}
\dot{z} &= \alpha_z\left(\beta_z(g-y)-z\right) \\
\dot{y} &= \alpha_y\left(f(s,v)+z\right) \\
&\text{where} \\
\dot{s} &= \alpha_s\left(-s+(1-s)\frac{b}{g^2}(g-r)^2\right) \\
\dot{r} &= \alpha_r(g-r) \\
\dot{v} &= \alpha_v\left(\beta_v(g-x)-v\right) \\
\dot{x} &= v
\end{aligned}
\qquad (9)
$$

In these equations, the variable y denotes the desired trajectory in joint space for our humanoid, g denotes the goal state, the variable s corresponds to a timing variable, and the variable v to a time varying scaling factor. Omitting the details of the development of (9) due to space limitation, the most important element to emphasize is the function $f(s,v)$ in the second equation of (9): $f(s,v)$ is our function approximation system that is employed here to change the attractor landscape of the dynamical system: s is used to localize function approximation in time, while v is the input to the local linear models. $f(s,v)$ acts thus as a time-variant gain to the integration of y. The entire dynamical system can be shown to be globally convergent to the goal state g, and globally

Figure 4: Imitation learning of a tennis forehand. The left column shows the teacher's movement, the right column the robot's movement towards a visually observed tennis ball.

Lyapunov stable for most of the component equations.

Given a joint-space representation of a teacher's movement, LWPPR can learn to imitate this trajectory essentially in one-shot learning using as learning output $\dot{y}_{des} = \dot{y}_{demo} / \alpha_y - z$. The learning result is nonlinear attractor landscape that retains the spatiotemporal pattern of the demonstration, but that can also be re-used for targets g at different positions and for different movement speeds—this property is guaranteed due to topological invariance of (9) under a change of the goal. Figure 4 illustrates learning movement imitation for a tennis forehand. The teacher's movement was recorded as joint angle trajectories by a special exoskeleton. A separate copy of the dynamic system in (9) was instantiated for each joint angle and the teacher's movement was approximated with LWPPR. The second column of Figure 4 show

the robot performance re-using the learned movement to hit a tennis ball, i.e., the learned movement was properly generalized to a novel target.

4 Conclusions

We presented a statistical learning approach for on-line learning of complex nonlinear functions. The suggested system was demonstrated to be useful in various complex learning task in humanoid robotics. We are currently working on Bayesian techniques to replace the gradient descent part of our learning methods, and also on inserting our learning approach in provably stable adaptive control methods.

5 Acknowledgments

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

6 References

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

[2] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996, pp. 514-520.

[3] V. N. Vapnik, *Estimation of dependences based on empirical data*. Berlin: Springer, 1982.

[4] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273-297, 1995.

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

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

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

[8] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," *Artificial Intelligence Review*, vol. 11, pp. 75-113, 1997.

[9] C. G. Atkeson and S. Schaal, "Memory-based neural networks for robot learning," *Neurocomputing*, vol. 9, pp. 1-27, 1995.

[10] W. S. Cleveland and C. Loader, "Smoothing by local regression: Principles and methods," : AT&T Bell Laboratories Murray Hill NY, 1995.

[11] T. J. Hastie and R. J. Tibshirani, "Nonparametric regression and classification: Part I: Nonparametric regression," in *From Statistics to Neural Networks: Theory and Pattern Recognition Applications. ASI Proceedings, subseries F, Computer and Systems Sciences*, V. Cherkassky, J. H. Friedman, and H. Wechsler, Eds.: Springer, 1994, pp. 120-143.

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

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

[14] C. G. Atkeson, "Memory-based approaches to approximating continuous functions," in *Nonlinear Modeling and Forecasting*, M. Casdagli and S. Eubank, Eds. Redwood City, CA: Addison Wesley, 1992, pp. 503-521.

[15] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Real-time robot learning with locally weighted statistical learning," presented at International Conference on Robotics and Automation (ICRA2000), San Francisco, April 2000, 2000.

[16] S. Schaal, S. Vijayakumar, and C. G. Atkeson, "Local dimensionality reduction," in *Advances in Neural Information Processing Systems 10*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds. Cambridge, MA: MIT Press, 1998, pp. 633-639.

[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] I. E. Frank and J. H. Friedman, "A statistical view of some chemometric regression tools," *Technometrics*, vol. 35, pp. 109-135, 1993.

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

[20] G. J. McLachlan and K. E. Basford, *Mixture models*. New York: Marcel Dekker, 1988.

[21] S. Vijayakumar and S. Schaal, "Locally weighted projection regression : An O(n) algorithm for incremental real time learning in high dimensional spaces," presented at Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford, CA, 2000.

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

[23] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, and S. Schaal, "Statistical learning for humanoid robots," *Autonomous Robots*, in press.

[24] C. G. Atkeson, J. Hale, M. Kawato, S. Kotosaka, F. Pollick, M. Riley, S. Schaal, S. Shibata, G. Tevatia, and A. Ude, "Using Humanoid Robots to Study Human Behaviour," *IEEE Intelligent Systems*, vol. 15, pp. 46-56, 2000.

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

[26] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," presented at IEEE International Conference on Intelligent Robots and Systems (IROS 2001), 2001.

[27] D. Bullock, S. Grossberg, and F. H. Guenther, "A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm," *J. of Cognitive Neuroscience*, vol. 5, pp. 408-435, 1993.

[28] A. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," presented at IEEE International Conference on Intelligent Robots and Systems (IROS 2001), 2001.