Learning Inverse Dynamics: a Comparison

Duy Nguyen-Tuong, Jan Peters, Matthias Seeger, Bernhard Schölkopf

Max Planck Institute for Biological Cybernetics
Spemannstraße 38, 72076 Tübingen - Germany

Abstract. While it is well-known that model can enhance the control performance in terms of precision or energy efficiency, the practical application has often been limited by the complexities of manually obtaining sufficiently accurate models. In the past, learning has proven a viable alternative to using a combination of rigid-body dynamics and handcrafted approximations of nonlinearities. However, a major open question is what nonparametric learning method is suited best for learning dynamics? Traditionally, locally weighted projection regression (LWPR), has been the standard method as it is capable of online, real-time learning for very complex robots. However, while LWPR has had significant impact on learning in robotics, alternative nonparametric regression methods such as support vector regression (SVR) and Gaussian processes regression (GPR) offer interesting alternatives with fewer open parameters and potentially higher accuracy. In this paper, we evaluate these three alternatives for model learning. Our comparison consists out of the evaluation of learning quality for each regression method using original data from SARCOS robot arm, as well as the robot tracking performance employing learned models. The results show that GPR and SVR achieve a superior learning precision and can be applied for real-time control obtaining higher accuracy. However, for the online learning LWPR presents the better method due to its lower computational requirements.

1 Introduction

Model-based robot control, e.g., feedforward nonlinear control [1], exhibits many advantages over traditional PID-control such as potentially higher tracking accuracy, lower feedback gains, lower energy consumption etc. Within the context of automatic robot control, this approach can be considered as an inverse problem, where the plant model, e.g, the dynamics model of a robot described by rigid-body formulation, is used to predict the joint torques given the desired trajectory (i.e., the joint positions, velocities, and accelerations), see, e.g., [1]. However, for many robot systems a sufficiently accurate plant model is hard to achieve using the pure rigid-body formulation due to unmodeled nonlinearities such as friction or actuator nonlinearities [2]. In such cases, the imprecise model can lead to large tracking errors which can only be avoided using high-gain control or more accurate models. As high-gain control would turn the robot into a danger for its environment, the latter is the preferable option. For this, one important alternative is the inference of inverse models from measured data using regression techniques.

While this goal has been considered in the past [3, 4], given recent progress in regression techniques and increased computing power for online computation,

Fig. 1: Anthropomorphic SARCOS master robot arm.

it is time that we reevaluate this issue using state-of-the-art methods. In this paper, we compare three different nonparametric regression methods for learning the dynamics model, i.e., the locally weighted projection regression (LWPR) [5], the full Gaussian processes regression (GPR) [6] and the ν-support vector regression (ν-SVR) [7]. The approximation quality is evaluated using (i) simulation data and (ii) real data taken from a 7 degree-of-freedom (DoF) SARCOS master robot arm, as shown in Figure 1. Furthermore, we will examine the tracking performances of the robot using the learned models in the setting of feedforward nonlinear control [1].

Our main focus during these evaluations is to answer two questions: a) which of the presented methods is suited best for our problem domain, and b) whether policies learned by support vector machines and Gaussian process can work in a real-time control scenario.

In the following, we will describe the role of inverse dynamics in nonlinear, feedforward robot control and, subsequently, the regression algorithms used for model approximation. Afterwards, we will discuss the results of model learning and how these can be used for control. Finally, we will show the performance during a real-time tracking task explaining our real-time robot control setup.

2 Inverse Dynamics Models in Feedforward Control

In model-based control, the controller command is computed using apriori knowledge about the system expressed in an inverse dynamics model [1, 8], which is traditionally given in the rigid-body formulation [1]: $\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{u}$, where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are joint angles, velocities and accelerations of the robot. $\mathbf{M}(\mathbf{q})$ denotes the inertia matrix and $\mathbf{F}(\mathbf{q}, \dot{\mathbf{q}})$ all internal forces, including Coriolis and centripetal forces, gravity as well as unmodel-able nonlinearities.

The motor command $\mathbf{u} = \mathbf{u}_{\mathrm{FF}} + \mathbf{u}_{\mathrm{FB}}$ is the applied joint torques and consists out of a feedforward component $\mathbf{u}_{\mathrm{FF}}$ and a feedback component $\mathbf{u}_{\mathrm{FB}}$. The feedforward component predicts the torques required to follow a desired trajectory given by desired joint angles $\mathbf{q}_d$, velocities $\dot{\mathbf{q}}_d$ and accelerations $\ddot{\mathbf{q}}_d$. If we have a sufficiently accurate analytical model, we can compute the feedforward component by $\mathbf{u}_{\mathrm{FF}} = \mathbf{M}(\mathbf{q_d})\,\ddot{\mathbf{q}}_\mathbf{d} + \mathbf{F}(\mathbf{q_d}, \dot{\mathbf{q}}_\mathbf{d})$. The feedback component is required to ensure that a tracking error cannot accumulate and destabilize the system. Linear feedback controllers $\mathbf{u}_{\mathrm{FB}} = \mathbf{K_p}\mathbf{e} + \mathbf{K_v}\dot{\mathbf{e}}$, with $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$ being tracking error, are commonly used in the feedforward control setting, where the feedback gains $\mathbf{K}_p$ and $\mathbf{K}_v$ are chosen such that they remain low for compliance while sufficiently high for stability [1].

However, for many robot systems the dynamics model presented by rigid-body equation as given is not sufficiently accurate, especially in case of unmodeled nonlinearities, complex friction and actuator dynamics [2]. This imprecise model leads to a bad prediction of joint torques $\mathbf{u}_{\mathrm{FF}}$ which can result in poor

control performances or even damage the system. Thus, learning more precise inverse dynamics models from measured data using regression methods poses an interesting alternative. In this case, the feedforward component is generally considered as a function of desired trajectories, hence, $\mathbf{u}_{\mathrm{FF}} = \mathbf{f}\left(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d\right)$.

3 Nonparametric Regression Methods for Model Learning

Learning the feedforward function is a straightforward regression problem as we can observe the trajectories resulting from our motor commands $\mathbf{u}$. Thus, we have to learn the mapping from inputs $\mathbf{x} = [\mathbf{q}^T, \dot{\mathbf{q}}^T, \ddot{\mathbf{q}}^T] \in \mathbb{R}^{3n}$ to targets $\mathbf{y} = \mathbf{u} \in \mathbb{R}^n$. With the learned function, the feedforward torque $\mathbf{u}_{\mathrm{FF}}$ can be predicted for a query input point $\mathbf{x}_d = [\mathbf{q}_d^T, \dot{\mathbf{q}}_d^T, \ddot{\mathbf{q}}_d^T]$. In the remainder of the section, we discuss three nonparametric regression techniques used for learning inverse dynamics models, i.e., the current standard method LWPR [5], ν-SVR [7] and GPR [6].

3.1 Locally Weighted Projection Regression (LWPR)

In LWPR, the predicted value $\hat{y}$ is given by a combination of N individually weighted locally linear models normalized by the sum of all weights [2, 5], Thus,

$$\hat{y} = \frac{\sum_{k=1}^{N} w_k \bar{y}_k}{\sum_{k=1}^{N} w_k}\,, \tag{1}$$

with $\bar{y}_k = \bar{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_k$ and $\bar{\mathbf{x}}_k = [(\mathbf{x} - \mathbf{c}_k)^T, 1]^T$, where w_k is the weight, $\hat{\boldsymbol{\theta}}_k$ contains the regression parameter and $\mathbf{c}_k$ is the center of the k-th linear model. For the weight determination, a Gaussian kernel is often used: $w_k = \exp(-0.5(\mathbf{x} - \mathbf{c_k})^{\mathbf{T}} \mathbf{D_k} (\mathbf{x} - \mathbf{c_k}))$, where $\mathbf{D_k}$ is a positive definite distance matrix. During the learning process, the main purpose is to adjust $\mathbf{D_k}$ and $\hat{\boldsymbol{\theta}}_k$, such that the errors between predicted values and targets are minimal [5].

3.2 Gaussian Processes Regression (GPR)

GPR is performed using a linear model: $y = f(\mathbf{x}) + \epsilon$ with $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T \mathbf{w}$, where $\mathbf{w}$ is the weight vector [6]. The linear computation is done after transforming the input $\mathbf{x}$ with a kernel function $\boldsymbol{\phi}(\bullet)$, for which the Gaussian kernel, as given in Section 3.1, can be taken. It is further assumed that the target value y is corrupted by a noise ϵ with zero mean and variance σ_n^2.

To make a prediction for a new input $\mathbf{x}_*$ the outputs of all linear models are averaged and additionally weighted by their posterior [6]. The predicted value $\bar{f}(\mathbf{x}_*)$ and corresponding variance $\mathbf{V}(\mathbf{x}_*)$ can be given as follow [6]

$$\begin{aligned} \bar{f}(\mathbf{x}_*) &= {\mathbf{k}_*}^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y} = {\mathbf{k}_*}^T \boldsymbol{\zeta}\,, \\ \mathbf{V}(\mathbf{x}_*) &= \mathbf{k}(\mathbf{x}_*, \mathbf{x}_*) - {\mathbf{k}_*}^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{k}_*\,, \end{aligned} \tag{2}$$

where $\mathbf{k}_* = \boldsymbol{\phi}(\mathbf{x}_*)^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi}$, $\mathbf{k}(\mathbf{x}_*, \mathbf{x}_*) = \boldsymbol{\phi}(\mathbf{x}_*)^T \boldsymbol{\Sigma}_p \boldsymbol{\phi}(\mathbf{x}_*)$ and $\mathbf{K} = \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_p \boldsymbol{\Phi}$. The matrix $\boldsymbol{\Phi}$ denotes an aggregation of columns $\boldsymbol{\phi}(\mathbf{x})$ for all cases in the training set and $\boldsymbol{\Sigma}_p$ the variance of the weights.

nMSE	Joint [i]						
[%]	1	2	3	4	5	6	7
LWPR	3.9	1.6	2.1	3.1	1.7	2.1	3.1
GPR	0.7	0.2	0.1	0.5	0.1	0.4	0.6
ν-SVR	0.4	0.3	0.1	0.6	0.2	0.5	0.4

Table 1: Learning error in percent for each DoF using simulation data.

nMSE	Joint [i]						
[%]	1	2	3	4	5	6	7
LWPR	1.7	2.1	2.0	0.5	2.5	2.4	0.7
GPR	0.5	0.3	0.1	0.1	1.5	1.2	0.2
ν-SVR	0.8	0.6	0.5	0.1	0.5	1.2	0.1
RBM	5.9	226.3	111.3	3.4	2.7	1.3	1.4

Table 2: Learning error in percent for each DoF using real SARCOS data.

Joint [i]	GPR	ν-SVR	LWPR
1	0.78	1.17	1.45
2	1.05	1.01	1.63
3	0.24	0.19	0.19
4	2.42	2.34	3.24
5	0.23	0.14	0.23
6	0.31	0.21	0.29
7	0.23	0.24	0.26

Table 3: Tracking error as nMSE in percent for each DoF using test trajectories.

3.3 ν-Support Vector Regression (ν-SVR)

For ν-SVR the predicted value $f(\mathbf{x})$ for a query point $\mathbf{x}$ is given by [7]

$$f(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\, k(\mathbf{x}_i, \mathbf{x}) + b\,, \tag{3}$$

with $k(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x})$ and m denotes the number of training points. The transformation $\boldsymbol{\phi}(\bullet)$ of the input vector can also be done with an appropriate kernel function as in the case of GPR. The quantities α_i^*, α_i and b are determined through an optimization procedure parameterized by $C \geq 0$ and $\nu \geq 0$ [7]. The parameter ν implies the width of the tube around the hyperplane (3) and C denotes the regularization factor for training [7].

4 Evaluations on Data Sets & Application in Control

In this section, we compare the learning performance of LWPR, GPR and ν-SVR using (i) simulation data and (ii) real SARCOS robot data. Generating the simulation data, we use a model of the 7-DoF SARCOS master arm created with the SL-software package [9].

4.1 Evaluation on Simulation Data

For the input data, a trajectory is generated such that it is sufficiently rich. Subsequently, we control the robot arm tracking those trajectory in a closed-loop control setting, where we sample the corresponding controller commands for the target data, i.e., the joint torques. In so doing, a training set and a test set with 21 inputs and 7 targets are generated which consist of 14094 examples for training and 5560 for testing. The training takes place for each DoF separately, employing LWPR, GPR and ν-SVR. Table 1 gives the normalized mean squared error (nMSE) in percent of the evaluation on the test set, where the normalized mean squared error is defined as: nMSE = Mean squared error/Variance of target.

It can be seen that GPR and ν-SVR yield better model approximation compared to LWPR, since GPR and ν-SVR are a global methods. A further advantage of these methods is that there are only some hyperparameters to be determined, which makes the learning process more practical. However, the main drawback is the computational cost. In general, the training time for GPR and ν-SVR is about 2-time longer compared to LWPR. The advantage of LWPR is the fast computation, since the model update is done locally. However, due to many meta parameters which have to be set manually for the LWPR-training, it is fairly tedious to find an optimal setting for those by trial-and-error.

4.2 Evaluation on Real Robot Data

The data is taken from the real anthropomorphic SARCOS master arm with 7 DoF, as shown in Figure 1. Here, we have 13622 examples for training and 5500 for testing. Table 2 shows the nMSE after learning with real robot data for each DoF. Additionally, we also determine the nMSE of a linear regression using the rigid-body robot model (RBM). The resulting error will indicate, how far the analytical model can explain the data.

Compared to LWPR, GPR and ν-SVR provide better results for every DoF. Considering the rigid-body model, the linear regression yields very large approximation error for the 2. and 3. DoF. Apparently, for these DoF the nonlinearities (e.g., hydraulic cables, complex friction) cannot be approximated well using just the rigid-body functions. This example shows the difficulty using the analytical model for control in practice, where the imprecise dynamics model will result in poor control performance for real system, e.g., large tracking error.

4.3 Application to Control

Using the offline-learned models from Section 4.1, the SL-model of the SARCOS robot arm [9] is controlled to accomplish a tracking task. For desired trajectories, i.e., joint angles, velocities and accelerations, we generate test trajectories which are similar to training trajectories, comparing the generalization ability of each regression method. Table 3 gives the tracking error of each joint as nMSE for the test trajectories. The Figure 2 shows the corresponding tracking performance for the joint 1 and 2, other joints are similar. It's necessary to emphasize that the control task is done in real-time where the system is sampled with 480 Hz.

It can be seen that the tracking error of GPR and ν-SVR is only slightly smaller than LWPR in spite of better learning accuracy. This is due to the reason that in case of GPR and ν-SVR, the controller command $\mathbf{u}$ can only be updated at every 4th sampling step due to more involved calculations for prediction, see Equations (2) and (3). In spite of those limitations, we are able to control the robot arm in real-time achieving a competitive performance. For LWPR, we are able to calculate the controller command for *every sampling step*, since evaluation of the prediction values (1) is quite fast. Furthermore, the results show that the learned models are able to generalized well in present of unknown trajectories similar to training data.

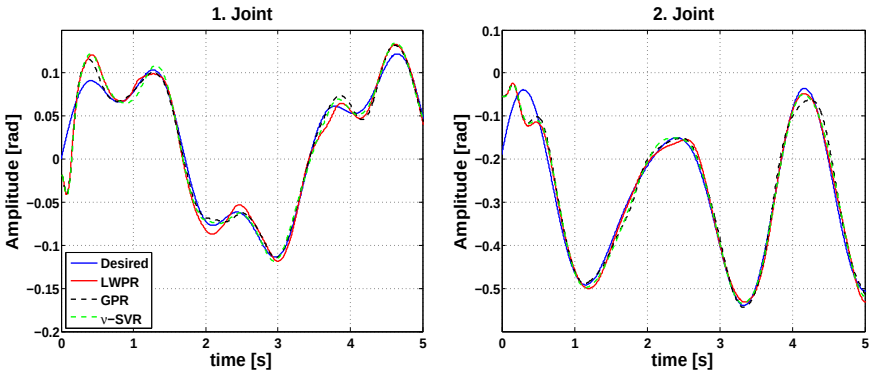

Fig. 2: Tracking performance for joint 1and 2. Other joints are similar.

5 Conclusion

Our results indicate that GPR and ν-SVR can be made to work for control applications in real-time, and that it is easier to apply to learning problems achieving a higher learning accuracy compared to LWPR. However, the computational cost is prohibitively high for online learning. Our next step is to modify GPR and ν-SVR, so that they can be used for an online regression and thus is capable for real-time learning. Here, the problem of expensive computation has to be overcome using other techniques, such as sparse or local models [10].

References

[1] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Prentice Hall, 3. edition edition, 2004.

[2] J. Nakanishi, Jay A. Farrell, and S. Schaal. Composite adaptive control with locally weighted statistical learning. *Neural Networks*, 2005.

[3] E. Burdet and A. Codourey. Evaluation of parametric and nonparametric nonlinear adaptive controllers. *Robotica*, 16(1):59–73, 1998.

[4] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard. Gaussian process model based predictive control. *Proceeding of the American Control Conference*, 2004.

[5] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. *International Conference on Machine Learning, Proceedings of the Sixteenth Conference*, 2000.

[6] Carl E. Rasmussen and Christopher K. Williams. *Gaussian Processes for Machine Learning*. MIT-Press, Massachusetts Institute of Technology, 2006.

[7] Bernhard Schölkopf and Alex Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT-Press, Cambridge, MA, 2002.

[8] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley and Sons, New York, 2006.

[9] S. Schaal. The SL simulation and real-time control software package. University of Southern California.

[10] D. Nguyen-Tuong. Machine learning for robot motor control. Thesis Proposal (unpublished). Max Planck Institute of Biological Cybernetics, 2007.