Cognitive Science (submitted) manuscript No.
(will be inserted by the editor)

Model Learning for Robot Control: A Survey

Duy Nguyen-Tuong · Jan Peters

the date of receipt and acceptance should be inserted later

Abstract Models are among the most essential tools in robotics, such as kinematics and dynamics models of the robot's own body and controllable external objects. It is widely believed that intelligent mammals also rely on internal models in order to generate their actions. However, while classical robotics relies on manually generated models that are based on human insights into physics, future autonomous, cognitive robots need to be able to automatically generate models that are based on information which is extracted from the data streams accessible to the robot. In this paper, we survey the progress in model learning with a strong focus on robot control on a kinematic as well as dynamical level. Here, a model describes essential information about the behavior of the environment and the influence of an agent on this environment. In the context of model based learning control, we view the model from three different perspectives. First, we need to study the different possible model learning architectures for robotics. Second, we discuss what kind of problems these architecture and the domain of robotics imply for the applicable learning methods. From this discussion, we deduce future directions of real-time learning algorithms. Third, we show where these scenarios have been used successfully in several case studies.

Keywords Model learning · Robot Control · Machine Learning · Regression

1 Introduction

Machine learning may allow avoiding the need to pre-program all possible scenarios, but rather learns the system during operation. There have been many attempts at creating learning frameworks, enabling robots to autonomously learn complex skills ranging from task imitation to motor control [132, 175, 133]. However, learning is not an easy task. For example, reinforcement learning can require more trials and data

D. Nguyen-Tuong · J. Peters
Max-Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany
Tel.: +49-7071-601-585
E-mail: duy.nguyen-tuong@tuebingen.mpg.de
E-mail: jan.peters@tuebingen.mpg.de

(a) Humanoid Robot DB (b) Mobile LAGR Robot (c) Boston Dynamics Little Dog

Fig. 1 Platforms with well-known applications of model learning: (a) Schaal et al. learned the complete inverse dynamics model for Humanoid DB [134]; (b) Angelova et al. predicted the slip of the mobile LAGR robot based on learned models that required visual features as input [4]; (c) Kalakrishnan et al. estimated foothold quality models based on terrain features for the Boston Dynamics little dog [60].

than one can generate in the life-time of a robot, and black box imitation learning can at best reproduce the desired behavior. Thus, it is essential to study how the basic, underlying mechanisms of the world can be learned. This approach is commonly known as *model learning*.

In recent years, methods to learn models from data have become interesting tools for robotics, as they allow straightforward and accurate model approximation. The reason for this increasing interest is that accurate analytical models are often hard to obtain due to the complexity of modern robot systems and their presence in unstructured, uncertain and human-inhabited environments [105, 93]. Model learning can be a useful alternative to manual pre-programming, as the model is estimated directly from measured data. Unknown nonlinearites can be directly taken in account, while they are neglected by the standard physics-based modeling techniques and by hand-crafted models. In order to generalize the learned models to a larger state space and to adapt the models for time dependent changes, online learning of such models is necessary. Model learning has been shown to be an efficient tool in a variety of scenarios, such as inverse dynamics control [105], inverse kinematics [124, 162, 52], robot manipulation [151, 66], autonomous navigation [4] or robot locomotion [60]. Figure 1 shows several examples of well-known applications of model learning. While there are many more applications of model learning in robotics, such as learning 3D-models of objects or map-models of the environment [117], this survey focuses on the core application of *model learning for control*. Here, a model can be used to describe the kinematics and dynamics of the robot's own body and controllable external objects. In the context of control, a model contains essential information about the system and describes the influence of an agent on this system. Thus, modeling a system is inherently connected with the question how the model can be used to manipulate, i.e., to control, the system on a kinematic as well as dynamical level.

1.1 Model Learning for Control: Problem Statement

Accurate models of the system and its environment are crucial for planning, control and many other applications. In this paper, we focus on generating learned models of dynamical systems that are in a state $\boldsymbol{s}_k$ taking an action $\boldsymbol{a}_k$ and transfer to a next state $\boldsymbol{s}_{k+1}$, where we can only observe an output $\boldsymbol{y}_k$ that is a function of the current state and action. Thus, we have

$$\begin{aligned} \boldsymbol{s}_{k+1} &= f(\boldsymbol{s}_k, \boldsymbol{a}_k) + \boldsymbol{\epsilon}_f \,, \\ \boldsymbol{y}_k &= h(\boldsymbol{s}_k, \boldsymbol{a}_k) + \boldsymbol{\epsilon}_y \,, \end{aligned} \tag{1}$$

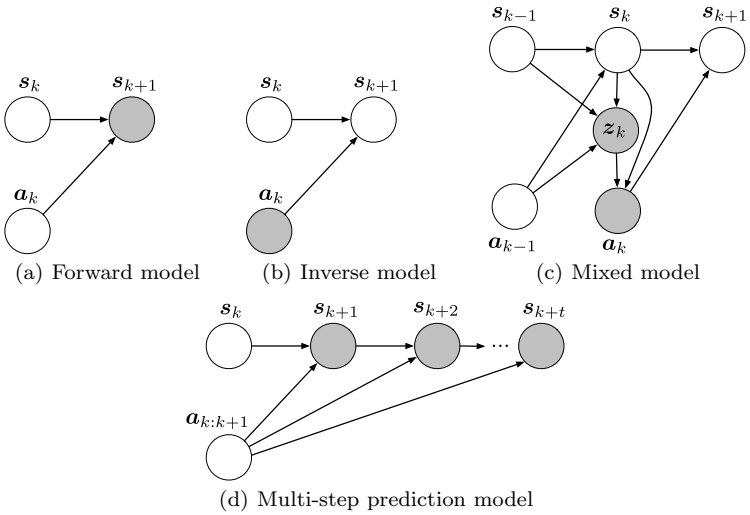
where f and h represent the state transition and the output function, $\boldsymbol{\epsilon}_f$ and $\boldsymbol{\epsilon}_y$ denote the noise components. In practice, state estimation techniques are often needed to reduce the noise of the state estimate and to obtain complete state information [67]. While the output function h can often be described straightforwardly by an algebraic equation, it is more difficult to model the state transition function f, as it includes more complex relationship between states and actions.

The state transition model f predicts the next state $\boldsymbol{s}_{k+1}$ given the current state $\boldsymbol{s}_k$ and action $\boldsymbol{a}_k$. Application of such state transition models in robotics and control has a long history. With the increasing speed of computation and its decreased cost, models have become common in robot control, e.g., in feedforward control and state feedback linearization. At the same time, due to the increasing complexity of robot systems, analytical models are more difficult to obtain. This problem leads to a variety of model estimation techniques which allow the roboticist to acquire models from data. Combining model learning with control has drawn much attention in the control community [37]. Starting with the pioneering work in adaptive self-tuning control [5], model based learning control has been developed in many aspects ranging from neural network controllers [111] to more modern control paradigms using statistical methods [68,95].

In early days of adaptive control [100,5], models are learned by fitting open parameters of pre-defined parametric models. Estimating such parametric models from data has been popular for a long time [6,64] due to the applicability of well-known system identification techniques and adaptive control approaches [77]. However, estimating the open parameters is not always straightforward, as several problems can occur, such as persistent excitation issues, i.e., optimal excitation of the system for data generation [99]. Furthermore, the estimated parameters are frequently not physically consistent (e.g., violating the parallel axis theorem or having physically impossible values) and, hence, physical consistency constraints have to be imposed on the regression problem [161]. Nonparametric model learning methods can avoid many of these problems. Modern nonparametric model learning approaches do not pre-define a fixed model structure but adapt the model structure to the data complexity. There have been strong efforts to develop nonparametric machine learning techniques for model learning in robotics and, especially, for robot control [94,37].

1.2 Overview

The aim of this paper is to give a comprehensive overview of past and current research activities in model learning with a particular focus on robot control. The remainder



(a) Forward model (b) Inverse model (c) Mixed model

(d) Multi-step prediction model

Fig. 2 Graphical illustrations for different types of models. The white nodes denote the *observed* quantities, while the grey nodes represent the quantities to be *inferred*. (a) The forward model allows inferring the next state given current state and action. (b) The inverse model determines the action required to move the system from the current state to the next state. (c) The mixed model approach combines forward and inverse models in problems where a unique inverse does not exist. Here, the forward and inverse models are linked by a latent variable $\boldsymbol{z}_t$. (d) The multi-step prediction model is needed when dealing with finite sequences of future states.

of this paper is organized as follows. First, we discuss different types of models in Section 2.1 and investigate how they can be incorporated into different learning control architectures in Section 2.2. In Section 2.3, we further discuss the challenges that arise from the application of learning methods in the domain of robotics. In Section 2.4, we provide an overview on how models can be learned using machine learning techniques with a focus on statistical regression methods. In Section 3, we highlight examples where model learning has proven to be helpful for the action generation in complex robot systems. The paper will be summarized in Section 4.

2 Model Learning

Any rational agent will decide how to manipulate the environment based on its observations and predictions on its influence on the system. Hence, the agent has to consider two major issues. First, it needs to deduce the behavior of the system from some observed quantities. Second, having inferred this information, it needs to determine how to manipulate the system.

The first question is a pure modeling problem. Given some observed quantities, we need to predict the missing information to complete our knowledge about the action and system's reaction. Depending on what kind of quantities are observed (i.e., what kind of missing information we need to infer), we distinguish between *forward models*, *inverse models*, *mixed models* and *multi-step prediction models*. Section 2.1 describes these models in more detail. The second question is related to the learning

control architectures which can be employed in combination with these models. In this case, we are interested in architectures that incorporate learning mechanisms into control frameworks. Section 2.2 presents three different model learning architectures for control, i.e., *direct modeling*, *indirect modeling* and *distal teacher learning*. In practice, model learning techniques cannot be used straightforwardly for many real-world applications, especially, for robot control. Section 2.3 gives an overview of challenges that appear when model learning is used in robotics. Section 2.4 approaches the model learning problem from the algorithmic viewpoint, showing how models can be learned using modern statistical regression methods. Here, we will distinguish between *local* and *global* learning approaches.

2.1 Prediction Problems and Model Types

To understand the system's behavior and how it reacts due to the agent's actions, we need information about the states and actions (of the past, the presence and sometimes the expected future). However, we have only access to a limited number of these quantities in practice. Thus, we need to predict the missing information given the known information.

If we can observe the current state $\boldsymbol{s}_k$, and the current action $\boldsymbol{a}_k$ is given, we can attempt to predict the next state $\boldsymbol{s}_{k+1}$. Here, the *forward model* can be used to predict the next state given current state and action. The forward model describes the mapping $(\boldsymbol{s}_k, \boldsymbol{a}_k) \rightarrow \boldsymbol{s}_{k+1}$. We can use the *inverse model* to infer the current action, i.e., the relation $(\boldsymbol{s}_k, \boldsymbol{s}_{k+1}) \rightarrow \boldsymbol{a}_k$, if we know the current state and the desired or expected future state. There are also approaches combining forward and inverse models for prediction, which we will refer to as *mixed model* approaches. However, for many applications the system behavior has to be predicted for the next t-steps rather than for the next single step. Here, we need models to predict a series of states; we call such models *multi-step prediction models*. Figure 2 illustrates these introduced models.

Given a model that provides predictions of the missing information, actions can be generated by the agent. The way how actions are generated based on the models is called control policy. Thus, a policy is understood as a decision for controlling the system, while a model reflects the behavior of the system. Depending on the model, appropriate policies should be taken to control the system. In this section, we additionally describe how the different models can be used in various control scenarios.

2.1.1 Forward Models

Forward models predict the next state of a dynamic system given the current action and current state. Note that the forward model directly corresponds to the state transfer function f shown in Equation (1). As this function expresses the physical properties of the system, the forward model represents a causal relationship between states and actions. Thus, if such causal mappings have to be learned, it will result in a well-defined problem and learning can be done straightforwardly using standard regression techniques. While forward models of classical physics are unique mappings, there are several cases where forward models alone do not provide sufficient information to uniquely determine the next system's state [69]. For instance, when a pendulum is located at an unstable equilibrium point, it is more likely to go to the left or right than to stay at the center. Nevertheless, the center point would be the prediction of a

forward model. Here, the modes of a conditional density may be more interesting than the mean function f [69,145,70].

An early application of forward models in classical control is the Smith predictor, where the forward model is employed to cancel out delays imposed by the feedback loop [147]. Later, forward models have been applied, for example, in the context of model reference adaptive control (MRAC) [100]. MRAC is a control system in which the performance of an action is predicted using a forward model (i.e., a reference model). The controller adjusts the action based on the resulting error between the desired and current state. Hence, the policy $\boldsymbol{\pi}$ for the MRAC can be written as

$$\boldsymbol{\pi}(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} \ \| \, f_{\text{forward}}(\boldsymbol{s}_t, \boldsymbol{a}) - \boldsymbol{s}_{t+1}^{des} \, \| \,, \tag{2}$$

where $\boldsymbol{s}_t^{\text{des}}$ denotes the desired trajectory, $\boldsymbol{a}$ the action chosen by the policy π and $\boldsymbol{s}_t$ represents the observed state at time t. MRAC was originally developed for continuous-time system and has been extended later for discrete and stochastic systems [100]. Applications of MRAC can be found numerously in robot control literature, such as adaptive manipulator control [108]. Further application of forward models can be found in the wide class of model predictive control (MPC) [82]. MPC computes optimal actions by minimizing a given cost function over a certain prediction horizon N in the future. The MPC control policy can be described by

$$\boldsymbol{\pi}(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}_{t:t+N}} \sum_{k=t}^{t+N} F_{\text{cost}} \left(f_{\text{forward}}(\boldsymbol{s}_k, \boldsymbol{a}_k) - \boldsymbol{s}_{k+1}^{\text{des}} \right) , \tag{3}$$

where F_{cost} denotes the cost function to be minimized, for $k > t$, $\boldsymbol{s}_k$ are predictions using the forward model, and $\boldsymbol{a}_{t:t+N}$ denotes the next N actions. MPC is widely used in the industry, as it can deal with constraints in a straightforward way. MPC was first developed for linear system models and, subsequently, extended to more complex nonlinear models [82]. Forward models have also been essential in model based reinforcement learning approaches, which relate to the problem of optimal control [156,9, 103]. Here, the forward models describe the so-called transition dynamics determining the probability of reaching the next state given current state and action. In contrast to previous applications, the forward models incorporate a probabilistic description of the system dynamics in this case [121,128]. More details about the applications of forward models for optimal control will be given in the case studies in Section 3.1. It is worth noting that from the biological view point forward models can be seen as a body schema, i.e., a sensorimotor representation of the body used for action [54]. Inspired by the biologically motivated body representations, the concept of body schema has been exploited in robotics, such as for data generation [85] or incorporating structural prior knowledge [168].

2.1.2 Inverse Models

Inverse models predict the action required to move the systems from the current state to a desired future state. In contrast to forward models, inverse models represent an anti-causal relationship. Thus, inverse models do not always exist or at least are not always well-defined. However, for several cases, such as for the robot's inverse dynamics, the inverse relationship is well-defined. Ill-posedness in learning inverse models can happen when the data space is not convex [58], for example, in multi-valued mappings. In such

cases, the model cannot be learned using standard regression techniques, as they tend to average over the non-convex solution space resulting in invalid predictions. General, potentially ill-posed inverse modeling problems can be solved by introducing additional constraints, as will be discussed in Section 2.1.3 in more detail.

For control, applications of inverse models can be traditionally found in computed torque robot control [29], where the inverse dynamics model is used to predict the torques required to move the robot along a desired joint space trajectory. The computed torque control policy can be described by

$$\boldsymbol{\pi}(\boldsymbol{s}) = f_{\text{inverse}}(\boldsymbol{s}, \boldsymbol{s}_{\text{des}}) + k(\boldsymbol{s} - \boldsymbol{s}_{\text{des}})\,, \tag{4}$$

where $k(\boldsymbol{s} - \boldsymbol{s}_{\text{des}})$ is an error correction term (for example, a PD-controller as both positions, velocities and accelerations may be part of the state) needed for stabilization of the robot. If an accurate inverse dynamics model is given, the predicted torques are sufficient to obtain a precise tracking performance. The inverse dynamics control approach is closely related to the computed torque control method. Here, the error correction term acts through the inverse model of the system [29] and, hence, we have a control policy given by

$$\boldsymbol{\pi}(\boldsymbol{s}) = f_{\text{inverse}}(\boldsymbol{s}, \boldsymbol{s}_{\text{des}}, k(\boldsymbol{s} - \boldsymbol{s}_{\text{des}}))\,. \tag{5}$$

If the inverse model perfectly describes the inverse dynamics, inverse dynamics control will perfectly compensate for all nonlinearities occurring in the system. Control approaches based on inverse models are well-known in the robotics community. For example, in motion control inverse dynamics models gain increasing popularity, as the rising of computational power allows to compute more complex models for real-time control. The concept of feedback linearization is another, more general way to derive inverse dynamics control laws and offers possibly more applications for learned models [146, 79].

2.1.3 Mixed Models

In addition to forward and inverse models, there are also methods which combine both types of models. As pointed out in preceding sections, modeling the forward relationship is well-defined, while modeling the inverse relation can lead to an ill-posed problem. The ill-posedness can occur when the mapping to be learned is not unique. A typical ill-posed inverse modeling problem is the inverse kinematics of redundant robots. Given a joint configuration $\boldsymbol{q}$, the task space position $\boldsymbol{x}$ can be determined exactly (i.e., the forward kinematic model is well-defined), but there may be many possible joint configurations $\boldsymbol{q}$ for a given task space position $\boldsymbol{x}$ (i.e., the inverse model could have infinitely many solutions and their combination is not straightforward). Due to the multi-valued relationship, the mapping $\boldsymbol{x} \rightarrow \boldsymbol{q}$ is ill-posed and $\boldsymbol{q}$ may form a non-convex solution space. Thus, when naively learning such inverse mapping from data, the learning algorithm will potentially average over non-convex sets of the solutions. The resulting mapping will contain invalid solutions which can cause poor prediction performance.

The basic idea behind the combination of forward and inverse models is that the information encoded in the forward model can help to resolve the non-uniqueness, i.e., the ill-posedness, of the inverse model. The ill-posedness of the inverse model can be resolved when it is combined with the forward model, such that the composite of these

models yields an identity mapping [58]. In this case, the inverse model will provide those solutions which are consistent with the unique forward model.

The mixed model approach, i.e., the composite of forward and inverse models, was first poposed in conjunction with the distal teacher learning approach [58], which will be discussed in details in Section 2.2.3. The proposed mixed models approach has subsequently evoked significant interests and has been extensively studied in the field of neuroscience [175,62]. Furthermore, the mixed model approach is supported by evidence that the human cerebellum can be modeled using forward-inverse composite models, such as MOSAIC [176,12]. While the mixed models have become well-known in the neuroscience community, the application of such models in robot control is not yet widespread. Pioneering work on mixed models in the control community can be found in [98,97], where the mixed models are used for model reference control of an unknown dynamical system. Even though mixed model approaches are not widely used in control, with the appearance of humanoid robots in the last few years, biologically inspired robot controllers are gaining more popularity. Controllers based on mixed models may present a promising approach [49,114,162].

The concept of mixed models can also be found in recently developed techniques for learning dynamical systems, such as learning predictive state distributions [76]. Here, a dynamical system is represented by a set of observable experiments, i.e., the so-called tests, which include sequences of action-state pairs. The prediction includes the task to find those test which match the observations produced by the real system. Thus, learning predictive state distribution incorporates the generation of a set of tests and the evaluations of these test samples. On a high level, learning how to generate and evaluate test samples can be can be seen as a model learning problem, where two models (i.e., for test generation and evaluation) need to be approximated [15]. Formulated in a probabilistic framework, these probabilistic models can be estimated empirically from samples [15].

2.1.4 Multi-step Prediction Models

The models introduced in preceding sections are mainly used to predict a single future state or action. However, in problems such as open-loop control, one would like to have information of the system for the next t-steps in the future. This problem is the multi-step ahead prediction problem, where the task is to predict a sequence of future values without the availability of output measurements in the horizon of interest. We call the models which are employed to solve this problem as *multi-step prediction models*. It turns out that such multi-step prediction models are difficult to develop because of the lack of measurements in the prediction horizon. A straightforward idea is to apply single-step prediction models t times in sequence, in order to obtain a series of future predictions. However, this approach seems to be susceptible to the error accumulation problem, i.e., errors made in the past are propagated into future predictions. An alternative to overcome the error accumulation problem is to apply autoregressive models which are extensively investigated in time-series prediction [2]. Here, the basic idea is to use models which employ past predicted values to predict future outcomes.

Combining multi-step prediction models with control was originally motivated by the need of extension of forward models for multi-step predictions [63]. In more recent work, variations of traditional ARX and ARMAX models for nonlinear cases have been proposed for multi-step prediction models [13,92]. However, multi-step prediction

Model Type	Learning Architecture	Example Applications
Forward Model	Direct Modeling	Prediction, Filtering, Learning simulations, Optimization
Inverse Model	Direct Modeling, Indirect Modeling	Inverse dynamics control, Computed torque control, Feedback linearization control
Mixed Model	Direct Modeling (if invertible), Indirect Modeling, Distal-Teacher	Inverse kinematics, Operational space control, Multiple-model control
Multi-step Prediction Model	Direct Modeling	Planning, Optimization, Model predictive control, Delay compensation

Table 1 Overview on model types associated with applicable learning architectures and example applications.

models based on some parametric structures, such as ARX or ARMAX are too limited for sophisticated, complex robot systems. The situation is even worse in the presence of noise or complex nonlinear dynamics. These difficulties are reasons to employ non-parametric multi-step prediction models for multi-step predictions [68, 43].

2.2 Learning Architectures

In previous section, we have presented different prediction problems that require different types of models. Depending on what quantities are observed, we need different models to predict the missing information. Here, we distinguished between forward models, inverse models, mixed models and multi-step prediction models. A central question when incorporating these models into a learning control framework is how to learn and adapt the models while they are being used. We will distinguish between direct modeling, indirect modeling and the distal teacher approach. Table 1 shows an overview of model types associated with applicable learning architectures.

In *direct modeling* approaches, we attempt to learn a direct mapping from input data to output data. In most cases, direct model learning can be performed using standard regression techniques, where the model learning is driven by the approximation errors, i.e., the errors between predictions and target outputs. However, direct model learning is only possible, when the relationship between inputs and outputs is well-defined. In case the input-output relationship is ill-posed (for example, when learning an inverse model) indirect and distal learning techniques can be used instead. When employing *indirect modeling* techniques, the model learning is driven by a particular error measure. For example, the feedback error of a controller can be used in this case. In *distal teacher* learning approaches, the inverse model of the system is used for control, and the learning of this inverse model is guided by a forward model. Figure 3 illustrates these three learning architectures. Compared to the direct modeling approaches, the indirect model learning and the distal teacher learning are *goal-directed*

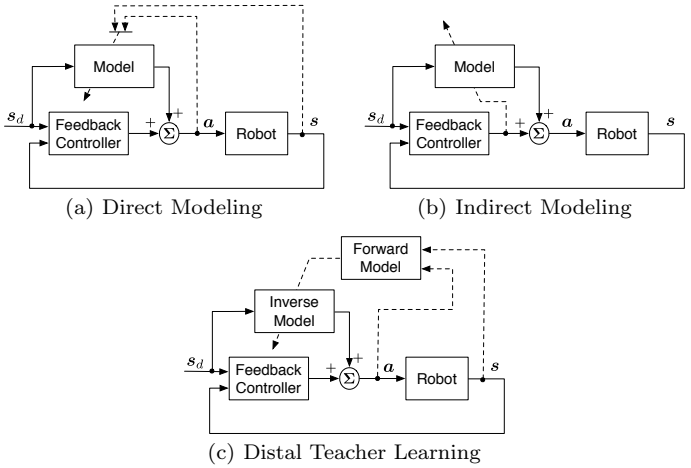


Fig. 3 Learning architectures in model learning applied to control. (a) In the direct modeling approach, the model is learned directly from the observations. (b) Indirect modeling approximates the model using the output of the feedback controller as error signal. (c) In the distal teacher learning approach, the inverse model's error is determined using the forward model. The resulting composite model will converge to an identity transformation.

learning techniques. Instead of learning a global mapping from inputs to outputs (as done by direct modeling), goal-directed learning approximates a particular solution in the output space. Due to this property, indirect and distal teacher learning approaches can be used for learning when confronting with an ill-posed mapping problem.

2.2.1 Direct Modeling

Direct learning is probably the most straightforward way to obtain a model but is not always applicable. In this learning paradigm, the model is directly learned by observing the inputs and outputs. Direct modeling as shown in Figure 3 (a) is probably the most frequently employed learning technique for model approximation in control, such as in vision-based control [87,86] or in inverse dynamics control [106]. Direct model learning can be implemented using most standard regression techniques, such as least square methods [77], neural networks [51,152,18] or statistical approximation techniques [122, 138,78].

An early example of direct learning in control was the self-tuning regulator that generates a forward model and adapts it online [5]. Using the estimated forward model, the self-tuning regulator will estimate an appropriate control law online. However, the forward model in the traditional self-tuning regulator has a fixed parametric structure and, hence, it cannot deal automatically with unknown nonlinearities [92,28]. The main reason why parametric models need to be used in direct modeling techniques is that such model parametrization is necessary for a convenient formulation of the control law and, more importantly, for the rigorous stability analysis. As parametric models are often too restrictive for complex robot systems, learned models with more degrees of freedom are needed, such as neural networks or fuzzy logic [170,75]. However, sophisticated learning algorithms for control are difficult to analyze if not impossible. Most work on the analysis of learning control has been done in neural control [111] and

model predictive control [47,102,96]. The operator model is an extension of forward models to multi-step prediction used in model predictive control. Direct learning of multi-step prediction models has been done with neural networks [25]. In more recent work, probabilistic methods are employed to learn such multi-step prediction models [43,68].

Inverse models can be learned straightforwardly in a direct manner, if the inverse mapping is well-defined. A well-known example is the inverse dynamics model required by computed torque and inverse dynamics control [29,150]. If direct modeling is applicable, learning becomes straightforward and can be achieved using standard regression techniques [134,105,22]. Early work in learning inverse models for control attempts to adapt a parametric form of the rigid body dynamics model. This model is linear in its parameters and, hence, it can be estimated from data straightforwardly using linear regression [6,17]. In practice, the estimation of dynamics parameters is not always straightforward. It is hard to create sufficiently rich data sets so that physically plausible parameters can be identified [93], and when identified online, additional persistent excitation issues occur [99]. Due to the fixed parametric structures, these models are not capable of capturing the structured nonlinearities of the real inverse dynamics. Physically implausible values often arise from such structural errors that result from a lack of representation for unmodeled nonlinearities. Hence, more sophisticated models have been introduced for learning inverse dynamics, such as neural networks [22,111] or statistical nonparametric models [134,105,106]. There have also been attempts to combine parametric rigid body dynamics model with nonparametric model learning for approximating the inverse dynamics [107]. Similar to inverse dynamics control, feedback linearization control can also be used in conjunction with direct model learning. Again, the nonlinear dynamics can now be approximated using neural networks or other nonparametric learning methods [41,94]. Stability analysis of feedback linearization control with learned models is possible, extending the cases where the nonlinear dynamics could not be canceled perfectly [94].

If the inverse mapping is ill-posed, the inverse models can be learned using indirect modeling or distal teacher learning as described in the next sections. However, there are also approaches which attempt to deal with the ill-posedness in a direct manner [33,78,114,127]. One approach deals with the question how to generate the data in an intelligent way, such that the ill-posedness can be avoided [33,127]. These approaches additionally requires explorations in the data space, where the explorations can be performed in the input space [33] or the target space [127]. Other techniques attempt to learn the manifold of solutions in the input-output space [32,78]. In [114, 162], the authors attempt to directly approximate the multi-valued relationship using local learning approaches. The core idea is to approximate different output solutions with different local models and, thus, resolving the ill-posedness in the prediction.

While direct learning is mostly associated with learning a single type of model, it can also be applied to mixed models. The mixed model approach (e.g., combining inverse and forward models) find its application in learning control for multi-module systems. The basic idea is to decompose a (probably) complex system into many simpler sub-systems which can be controlled individually [97]. The problem is how to choose an appropriate architecture for the multiple controllers, and how to switch between the multiple modules. Employing the idea of mixed models, each controller module consists of a pair of inverse and forward models. The intuition is that the controller can be considered as an inverse model, while the forward model is essentially used to switch between the different modules [175]. Such multiple pairs of forward and

inverse models can be learned directly from data using gradient-descent methods or expectation-maximization [49].

2.2.2 Indirect Modeling

Direct model learning works well when the input-output relationship is well-defined as in inverse dynamics. However, there can be situations where this relationship is not well-defined, such as in the differential inverse kinematics problem. In such cases, these models can often still be learned indirectly. One indirect modeling technique which can solve some of such ill-posed problems is known as feedback error model learning [61]. Feedback error learning relies on the output of a feedback controller that is used to generate the error signals employed to learn the feedforward controller, see Figure 3 (b). In several problems, such as feedforward inverse dynamics control [29], this feedback error learning approach can be understood particularly well. If the inverse dynamics model in the feedforward loop is a perfect model, the corresponding feedback controller is silent (and its output will be zero). If the feedback error is non-zero, it corresponds to the error of the inverse model in the feedforward loop [29]. The intuition behind feedback error learning is that by minimizing the feedback errors for learning the inverse model, the feedback control term will decrease as the model converges. Thus, the inverse model will describe the inverse dynamics of the system, while the feedback control part becomes irrelevant.

Compared to the direct model learning, feedback error learning is a goal-directed model learning approach resulting from the minimization of feedback errors. Here, the model learns a *particular* output solution for which the feedback error is zero. Another important difference between feedback error learning and direct learning is that feedback error learning has to perform online, while direct model learning can be done both online and offline. Feedback error learning is biologically motivated due to its inspiration from cerebellar motor control [62]. It has been further developed for control with robotics applications, originally employing neural networks [144, 88]. Feedback error learning can also be used with various nonparametric learning methods [95]. Conditions for the stability of feedback error learning control in combination with nonparametric approaches have also been investigated [95]. It is worth noting that indirect modeling can also be performed with other error measure, such as output error [118].

Indirect model learning can also be used in the mixed model approach [45]. Here, the attempt has been made to combine the feedback error learning with the mixture of experts architecture to learn multiple inverse models for different manipulated objects, where the inverse models are learned indirectly using the feedback error learning approach [45]. In this approach, the forward model is used for training a gating network, as it is well-defined. The gating network subsequently generates a weighted prediction of the multiple inverse models, where the predictors determine the locally responsible models.

2.2.3 Distal Teacher Learning

The distal teacher learning approach was motivated by the necessity to learn general inverse models, which suffer from the problem of ill-posedness [58]. Here, the non-uniqueness of the inverse model is resolved when combined with an unique forward model. The forward model is understood as a “distal teacher” which guides the learning

Data Challenges	Algorithmic Constraints	Real-World Challenges
High-dimensionality, Smoothness, Richness of data, Noise, Outliers, Redundant data, Missing data	Incremental updates, Real-time, Online learning, Efficiency, Large data sets, Prior knowledge, Sparse data	Safety, Robustness, Generalization, Interaction, Stability, Uncertainty in the environment

Table 2 Challenges of real-world problems for machine learning

of the inverse model. In this setting, the unique forward model is employed to determine the errors made by the inverse model during learning. The aim is to learn the inverse model such that this error is minimized. The intuition behind this approach is that the inverse model will learn a correct solution for a *particular* desired trajectory when minimizing the error between the output of the forward model and the input of the inverse model. Thus, the inverse model will result in solutions that are consistent with the unique forward model. However, distal teacher learning has several potential disadvantages, such as learning stability, numerical stability and error accumulation.

Nevertheless, the distal teacher approach has successfully learned particular solutions for multi-valued mappings, such as inverse kinematics of redundant robots [58]. Similar to feedback error model learning [61], distal teacher learning is also a goal-directed learning method applicable for various robot control scenarios. However, unlike the feedback error learning approach, distal teacher learning allows directly aiming at a globally consistent inverse model instead of local on-policy optimization. In practice, the distal teacher employs two interacting learning process: one process where the forward model is learned, and another process where the learned forward model is used for determining the error of the inverse model. In the original distal learning approach, the inverse model's output is validated by the forward model, as the composite of these models yields an identity mapping if perfectly learned [58].

The distal learning approach is particularly suitable for control when combining with the mixed models, as it naturally incorporates the mixed model principle. The distal teacher learning approach with mixed models has motivated a number of follow-up projects with several robot control applications [33, 114, 162].

2.3 Challenges and Constraints

In previous sections, we give an overview of different types of models and how these models can be incorporated into various learning architectures. However, employing machine learning methods, such as statistical methods [122, 138, 172] for learning such models, is not always straightforward. Several important problems need to be tackled in order to customize general learning algorithms for an application in robotics. In this section, we give an overview of these problems and discuss how these problems can be approached in order to bring machine learning algorithms into robotics. In particular, we consider the problems that arise from *data*, from employed *algorithms* and from *real-world* challenges. These are summarized in Table 2.

2.3.1 Data Challenges

In order to learn a "good" model for applications in robotics, the sampled data has to cover a large region of the model state space though, of course, it can never cover the complete state space. For the purpose of generalization, the generated data has to be sufficiently rich, i.e., it should contain as much information about the system as possible. Thus, generating large and rich data sets for model learning is an essential step (which is sometimes not easy in practice). This step often requires additional excitation of the robot system during data generation, which is known as persistent excitation in classical system identification [99]. For several systems, the persistent excitation condition is naturally given, such as aircraft systems. For other systems, the persistent excitation condition has to be generated artificially, e.g., by adding small random movements into to the output of the system.

For learning inverse dynamics, for example, rich data can be sampled from trajectories by approximately executing desired random point-to-point and rhythmic movements [157,135]. In other approaches [40,110], the generation of an optimal robot excitation trajectory involves nonlinear optimization with motion constraints (e.g., constraints on joint angles, velocity and acceleration). These approaches distinguish in the way of the trajectory parameterization. For example in [110], trajectories are used which are a combination of a cosine and a ramp. The parameters for these excitation trajectories are optimized from the data sampled in the robot's work space. In order to enhance the system's excitation, input and output noise can be further added to the generated trajectories.

The data used for model learning has to be sufficiently smooth, which is a key assumption for most of machine learning methods. However, there are many applications in robotics where the approximated functions are known to be non-smooth. For example, stiction-friction models are often non-smooth. Such non-smooth functions can sometimes be approximated using kernel methods. As a kernel implicitly incorporates the smoothness of the approximated function, special kernels can be defined in order to take the expected non-smoothness in account [138,122]. An example of such types of kernels are Matern-kernels which are widely used in Bayesian inference methods [122]. Discontinuities in a non-smooth function can also be approximated by local models and by learning how to switch discontinuously between these local models [164].

Robot systems that have a large number of DoFs pose a challenging problem due to the high dimensionality of the generated data. For example, in inverse dynamics the learning methods have to deal with data in a space with $4n$ dimensions (incorporating the information of position, velocity, acceleration and torques of the system), where n is the number of DoFs. This difficulty can be tackled by preprocessing the data using dimensionality reduction, which is a well-studied technique in machine learning [158,129]. The application of dimensionality reduction is based on the insight that the useful information in the data often lies on a low-dimensional manifold of the original input space. Dimensionality reduction methods have proven to be a powerful method for model learning in high dimensional robot systems [134,53].

As the data is sampled over a possibly long period of time in many robotics applications [160], problems of redundant and irrelevant data can occur. In such cases, redundant and irrelevant data can bias the model which severely hurts the generalization. Here, the data can be filtered in an appropriate way by selecting only data points that are informative for the learning process. This filtering step can be combined with

the learning step, for example, by using information criteria for inserting and deleting data points [106, 35].

Noise and outliers have always been a challenging problem for machine learning and for robot learning. Naively learning a model from noisy data can make the model fit the noise (i.e., an over-fit) and, thus, adulterate the model learning performance. In the past decade, considerable efforts have been made in the machine learning community to deal with this problem. In particular, regularization frameworks are developed based on statistical learning theory. The basic idea is to constrain the model to be learned in an appropriate way, attenuating the contributions made by the noisy components in the data. This leads to a variety of model learning methods, such as support vector regression or Gaussian process regression [149, 119]. One step in these methods is to estimate the noise-level in the data represented by a regularization parameter, which can either be done by cross-validation or by maximizing the marginal likelihood function [142]. By controlling the noise in the data, these methods can significantly improve the generalization performance.

2.3.2 Algorithmic Constraints

There are two scenarios for model learning in robotics: large data and small data. In the first case, learning algorithms have to deal with massive amounts of data, such as in learning inverse dynamics. In this scenario, the algorithms need to be efficient in terms of computation without sacrificing the learning accuracy [16]. In the second scenario, there is only few data available for learning, as the data generation may be too tedious and expensive. Here, we need algorithms which allow us to improve the learning performance in the presence of sparse data, for example, by incorporating additional prior knowledge [137, 73] or using active learning [27, 84].

For machine learning techniques, fast, real-time computation is challenging. Standard model learning approaches, such as Gaussian process regression, for example, scale cubically in the number of training data, preventing a straightforward usage in robotics. Sparse and reduced set methods smartly reduce the size of training data and, thus, decrease the computational effort for the learning and the prediction step [20]. In recent years, there have been serious efforts to speed up machine learning algorithms with efficient implementations using, for example, parallel computation [42].

Online learning is also a strong requirement of the domain of robotics. Most of machine learning methods are developed for learning in batch mode, i.e., offline learning using pre-sampled data sets, while online learning requires incremental approximation of the model. However, online learning has found increasing interest over the last decade giving rise to a number of real-time online machine learning approaches, such as in [171, 22, 105, 124]. A major motivation for online model learning is the insight that it is not possible to cover the complete state space with data beforehand, but that only the interesting state space regions are only known during the execution. Thus, online model learning will require incremental acquisition of knowledge and, possibly, even partial forgetting of the recorded information in order to cope with errors as well as change. Furthermore, online learning presents an essential step towards continuous adaptation to a changing world which is essential to make robots more autonomous [160].

Incorporating prior knowledge into the learning process can be obtained straightforwardly, when statistical learning approaches are used. In kernel methods, prior knowledge can be specified by feature vectors which can be used to define appropriate kernels

[137]. In contrast, probabilistic frameworks allow one to specify priors to capture a priori information [122]. If prior knowledge is given as a parametric model, it can be inserted into nonparametric models in a straightforward way, yielding semiparametric learning approaches [107,148]. Semiparametric models have shown to be capable in learning competitive models, when only few data is available [107].

Supervised learning requires labeled training data. When the labeling of training data is expensive, such as in path planning [84] or grasping [71], active learning can be used to label training data for model learning. Active learning requires an interaction with the human for the data labeling process. The goal is to obtain a sparse and informative labeled training set by intelligent adaptive querying [27,31]. Active learning can help to learn a good model with significantly fewer labels than one would need in a regular supervised learning framework.

2.3.3 Real-World Challenges

In order to ensure safe interaction of robots with human beings in everyday life, machine learning algorithms developed for robotics applications have to be fail-safe or at least have to minimize the risk of damage. For critical applications, such as medical or service robotics, robustness and reliability are among the most important criteria which have to be fulfilled by model learning. Model learning can become more robust when feature selection is employed as a preceding step. Feature selection methods remove irrelevant and redundant data and, thus, make the model learning more robust. Feature selection has an inherent connection to sparse and reduced set methods, where the purpose is to filter the information which is crucial for the model approximation [30,136]. Feature selection has been an active research field in machine learning for many years and has now found its ways to several robot applications both in robot vision and control [72, 106].

Robustness also requires the learning algorithm to deal with missing data. This problem is encountered in every robotics set-up where the sensor information is imperfect, such as in terrain modeling or autonomous navigation. In particular, measurement errors often result in missing data. In the recent years, the problem of missing data has attracted much attention with the rise of probabilistic learning methods. As the models are probabilistic, it is now possible to infer the missing "pieces" in the training data [163]. A further advantage of the probabilistic methods consists of a straightforward way to assign its uncertainty to each predicted value and, hence, make it easier to deal with insufficient data.

Modeling non-stationary systems is also a requirement for several robot applications, for example, when dealing with time dependent dynamics [115] or planning in a changing environment [154]. In such cases, the learned models need to be adapted to the changes and, thus, online model learning is necessary [30,105].

2.4 Applicable Regression Methods

In preceding section, some problems are summarized which need to be overcome when employing machine learning in robotics. In this section, the model learning is approached from the algorithmic point of view providing an overview of how models can

be learned using machine learning techniques. Here, the focus will be on modern statistical methods for learning models from data. However, connections to other popular learning approaches such as neural networks will also be discussed [152, 51].

Although model learning can be performed with unsupervised learning in few cases [174, 14], supervised learning is the most frequently employed learning technique. Supervised learning enables a fast and robust model approximation. However, it requires labeled training data as ground truth. In cases where labeled data is hard to obtain or not available, unsupervised learning can be employed instead [131, 39]. Unsupervised model learning requires only input data points in most cases, corresponding target outputs can be inferred from observations of the system. Usually, unsupervised model learning needs additional explorations in order to generate sufficient informative data for the learning process. Therefore, model learning using unsupervised learning techniques may require a long training process and it can be difficult to obtain a sufficiently good model. Nevertheless, unsupervised learning has found its way to several model learning applications [174, 14, 155]. In [14, 155], unsupervised learning is employed to learn robot's kinematic models. In [174], models for object recognition are learned using unsupervised learning techniques based on density estimation.

In the following section, we focus on model learning using supervised learning approaches. Here, it is assumed that the input $\boldsymbol{x}$ and target output $\boldsymbol{y}$ are given, where the true output data is corrupted by noise $\boldsymbol{\epsilon}$, i.e.,

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}\,. \tag{6}$$

Approximating the underlying function f is the goal of supervised learning methods. Given unknown input data the learned model should be able to provide precise predictions of the output values. Different supervised learning techniques make different assumptions on how to model the function f.

Here, we distinguish between *global* and *local* techniques used to model the underlying function f. Global regression techniques model the underlying function f using *all* observed data to construct a single global prediction model [50]. In contrast to global methods, the local regression estimates the underlying function f *within* a local neighborhood around a query input point. Beyond the local and global types of model learning, there are also approaches which combine both ideas. An example of such hybrid approaches is the mixture of experts [55, 109]. Here, the data is partitioned into smaller local models in a first step and, subsequently, a gating network is used to fuse these local models for global prediction. Mixture of experts approaches have been further embedded into the Bayesian framework giving rise to a number of Bayesian hybrid approaches such as committee machines [165], mixtures of Gaussian models [166, 19] or infinite mixtures of experts [120]. Table 3 provides several examples of machine learning methods which have been applied in model learning for robotics.

2.4.1 Global Regression

A straightfoward way to model the function f in Equation (6) is to assume a parametric structure, such as linear or polynomial models or multilayer perceptron neural networks, and, subsequently, fit the model parameters using training data [50, 48, 51]. However, fixing the model with a parametric structure beforehand may not suffice to explain the sampled data, which motivates nonparametric model learning frameworks

Method	Type	Mode	Online	Complexity	Learning Applications
Locally Weighted Projection Regression [172]	Local	Incremental	Yes	$\mathcal{O}(n)$	Inverse dynamics [134], Foothold quality model [60]
Local Gaussian Process Regression [105]	Local	Incremental	Yes	$\mathcal{O}(m^2)$	Inverse dynamics [105]
Gaussian Mixture Model [55]	Semi-Local	Batch	No	$\mathcal{O}(Mn)$	Human motion model [19]
Bayesian Comittee Machine [165]	Semi-Local	Batch	No	$\mathcal{O}(m^2n)$	Inverse dynamics [122]
Sparse Gaussian Process Regression [30]	Global	Incremental	Yes	$\mathcal{O}(n^2)$	Transition dynamics [128], Task model [46]
Gaussian Process Regression [142]	Global	Batch	No	$\mathcal{O}(n^3)$	Terrain model [117], State estimation model [67]
Support Vector Regression [138]	Global	Batch	No	$\mathcal{O}(n^2)$	ZMP control model [38], Grasp stability model [112]
Incremental Support Vector Machine [81]	Global	Incremental	Yes	$\mathcal{O}(n^2)$	Inverse dynamics [24]

Table 3 A large variety of machine learning methods have been applied in model learning for robotics. We distinguish between global and local methods, as well as semi-local methods which combine both approaches. The methods differ in the training mode and their online capabilities. For computational complexity, n denotes the total number of training points, m is number of data points in a local model, and M is the number of local models. We further provide several application examples for model learning in robotics.

[48, 138, 122]. In the modern parametric and nonparametric regression, the function f is usually modeled as

$$f(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{x}) \,, \tag{7}$$

where $\boldsymbol{\theta}$ is a weight vector and $\boldsymbol{\phi}$ is a nonlinear function projecting the input $\boldsymbol{x}$ into some high-dimensional spaces. The basic idea behind nonparametric regression is that the optimal model structure should be obtained from the training data. Hence, the size of the weight vector $\boldsymbol{\theta}$ is not fixed but can increase with the number of training data points in most statistical learning methods. It determines the structure of the model given in Equation (7). Compared to nonparametric statistical approaches, traditional neural networks fix the model structure beforehand [51]. For instance, the number of nodes and their connections have to be determined before starting the training procedure [51]. However, there are attempts to develop neural networks which can change their structures dynamically, such as in reservoir computing [140, 80, 125] and echo state neural networks [153, 59, 126]. It is worth noting that there have been attempts to put artificial neural networks into the Bayesian framework [83, 101] and, thus, establishing the connection between the two learning approaches. It has also been observed that certain neural networks with one hidden layer converge to a Gaussian process prior over functions [101].

In kernel methods, the model structure is determined by the model complexity [138]. Learning a model includes finding a tradeoff between the model's complexity and the best fit of the model to the observed data. It is desirable to have a model which is simple but at the same time can explain the data well. Using kernel methods, the weight vector $\boldsymbol{\theta}$ in Equation (7) can be first expanded in term of n training data points and, subsequently, regularized in an optimization step. Intuitively, the weight vector $\boldsymbol{\theta}$ represents the complexity of the resulting model.

Having a close link to the kernel framework, probabilistic regression methods additionally provide a Bayesian interpretation of nonparametric kernel regression [122]. Instead of expanding the weight vector as done in kernel methods, probabilistic methods place a prior distribution over $\boldsymbol{\theta}$. The prior parameters can be subsequently obtained by optimizing the corresponding marginal likelihood. Thus, the trade-off between data-fit and model complexity can be obtained in a straightforward and plausible way [122].

Kernel and probabilistic methods have proven to be successful tools for model learning over the last decade, resulting in a number of widely applied regression methods, such as support vector regression [139, 149, 81] or Gaussian process regression [119]. These methods are known to be capable of being applicable to high-dimensional data. They can also deal well with noisy data, as the noise is taken in account indirectly by regularizing the model complexity. Furthermore, they are relatively easy to use, as several black-box implementations are available. However, the major drawback of these methods are the computational complexity. Thus, one active research line in machine learning is to reduce the computational cost of those approaches. Due to several advances in customizing machine learning techniques for robotics, kernel and probabilistic regression techniques have aroused increasing interests and have found their ways to several robotics applications, such as modeling inverse dynamics [24], grasping [112], robot control [38, 106], imitation learning [46], sensor modeling [116, 117] and state estimation [67].

2.4.2 Local Learning

The basic idea of local regression techniques is to estimate the underlying function f *within* a local neighborhood around a query input point $\boldsymbol{x}_q$. The data points in this neighborhood can then be used to predict the outcome for the query point. Generally, local regression models can be obtained by minimizing the following cost function J using n training data points

$$J = \sum_{k=1}^{n} w\left(\frac{\boldsymbol{x}_k - \boldsymbol{x}_q}{h}\right)\left(\boldsymbol{y}_k - \hat{f}(\boldsymbol{x}_k)\right)^2 . \tag{8}$$

As indicated by the Equation (8), the essential ingredients for a local regression model are the neighborhood function w and the local model $\hat{f}$. The neighborhood function w, which is controlled by a width parameter h, basically measures the distance between a query point $\boldsymbol{x}_q$ to the points in the training data. The local model $\hat{f}$ describes the function structure used to approximate f within the neighborhood around $\boldsymbol{x}_q$ [26, 36]. Depending on the complexity of the data, different function structures can be assumed for the local model $\hat{f}$, such as a linear or a polynomial model. The open-parameters of $\hat{f}$ can be estimated straightforwardly by minimizing J with respect to these parameters. However, the choice of the neighborhood function and its width parameter is more involved. Several techniques have been suggested for estimating the width parameters

for a given w, including the minimization of the leave-one-out cross validation error and adaptive bandwidth selection [57,90].

Because of their simplicity and computational efficiency, local regression techniques have become widespread in model learning for robotics [89,8,159]. In the last decade, novel local regression approaches have been further developed in order to cope with the demands in many robotics real-time applications, such as locally weighted projection regression [7,171]. Inspired by local regression techniques, these methods first employ a partitioning of the input space into smaller local regions, for which locally linear models are approximated. In addition to being computational efficient, local methods can deal with less smooth functions and do not require the same smoothness and regularity conditions as global regression methods. However, it has been shown in practice that local methods suffer from problems induced by high-dimensional data, as notions of locality break down for sparse, high-dimensional data. Furthermore, the learning performance of local methods may be sensitive to noise and heavily depends on the way how the input space is partitioned, i.e., the configuration of the neighborhood function w. These problems still present an active research topic [34,162].

Several attempts have been made to scale local regression models to higher dimensional problems as required for many modern robotics systems. For example, locally weighted projection regression combines local regression with dimensionality reduction by projecting the input data into a lower dimensional space, where local regression is employed afterwards [171]. Other methods combine nonparametric probabilistic regression, such as Gaussian process regression, with the local approaches while exploiting the strength of probabilistic methods for model learning in high-dimensions [105,162, 169].

3 Application of Model Learning

In this section, we discuss three case studies on model learning in different robot applications. The presented cases illustrate several different aspects of model learning discussed in previous sections. This list of examples is obviously not exhaustive but gives an overview on possible applications of model learning in robotics. In Section 3.1, an application of forward models is illustrated from several examples. In Sections 3.2 and 3.3, we highlight cases where inverse models and mixed models are useful.

3.1 Simulation-based Optimization

As forward models directly describe the dynamic behavior of the system, learning such models has evoked much attention in the field of robot control for a long time. A key application of learned forward models is the optimization of control problems. In this situation, a policy that has been optimized for a hand-crafted model is likely to be biased by the large model errors, while optimization on the real system is too costly. Hence, policy optimization based on learned forward models is an interesting alternative.

Atkeson et al. [9] were among the first to explore this approach using differential dynamic programming [56] for optimizing open-loop control policies. The basic idea of Atkeson et al. is to use receptive field-weighted regression (a type of locally weighted regression) to learn the models of both cost and state transition. Differential

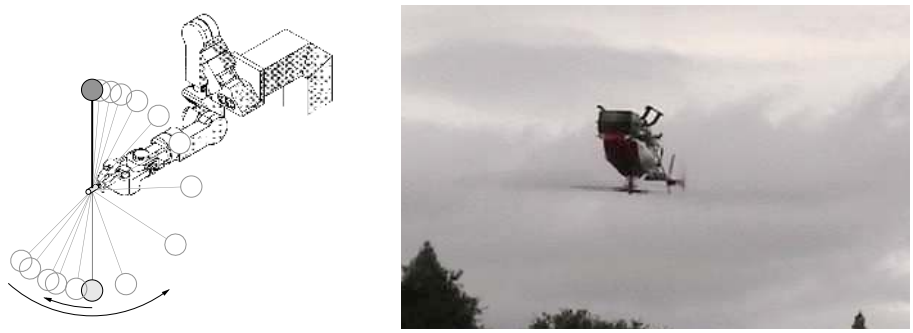


Fig. 4 Learning the pendulum swing up task (with permission of Stefan Schaal) and learning inverted flight with a helicopter (with permission of Andrew Y. Ng). In both cases, the forward model is used to learn the dynamics of the system for policy optimization. In the pendulum swing up task, the robot learns to bring the pole from an hanging to an upright position. In the inverted helicopter flight, the learned forward model is employed as simulator for generating complete roll-outs for learning a control policy. After learning, the helicopter is able to fly inverted while remaining perfectly still.

dynamic programming locally linearizes the state transition model and generates a local quadratic approximation of the cost. These approximations are used to improve an open-loop policy where the linearizations are also updated after every policy update [9]. Atkeson et al. used the method to learn the underactuated pendulum swing up task, where a pole is attached to the endeffector of the robot and maximal torque has been limited to a fixed value. The goal of the robot is to bring the pole from an hanging to an upright position. Hence, the system needs to "pump" energy into the pendulum in order to swing it up. Subsequently, it needs to limit the energy so that it can stabilize the pole at the upright position [10]. Starting from an unconstrained human demonstration, the robot was able to successfully learn the swing up and balance task after three trials [10]. Figure 4 shows an illustration for the pole swinging task. The local trajectory optimization technique has been further extended to biped robot walking [91]. More recently, a related approach with parametric function approximation has been applied by Abbeel et al. to learn autonomous helicopter flight [1]. The authors also reported fast convergence of this approach when learning different moves for the helicopter, such as flip and roll movements [1].

While Atkeson [9] and Abbeel [1] used the forward model as an implicit simulator, Ng et al. [103] use it as an explicit simulator (as originally suggested by Sutton [156] in form of the DYNA model). Here, the forward model acts as a simulator for generating complete trajectories or roll-outs. The predictions of the forward model are further perturbed by Gaussian noise with a repeating, fixed noise history (e.g., by resetting the random seed, a trick well-known in simulation optimization [44] which is the most common way to implement PEGASUS for complex systems [104]). This perturbation step is required to make the system more robust to noise and model errors, while the re-use of the noise history limits the variance in the policy updates (which results in a major speed-up). This simulator based on a learned forward model is used for generating complete roll-outs from a similar start-state set for a control policy. The resulting performance of different control policy can be compared, which allows policy updates both by pair-wise comparison or by gradient-based optimization. The approach has been able to successfully stabilize a helicopter in an inverted flight. Figure 4 shows

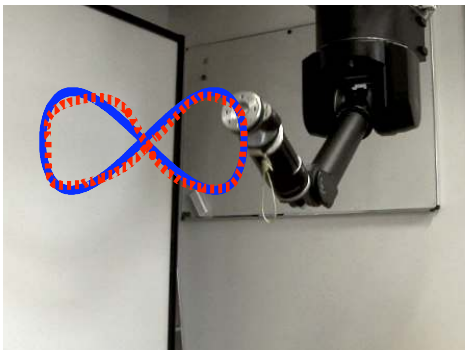


Fig. 5 The 7-DoF anthropomorphic Barrett arm used in learning inverse dynamics. The robot learns an inverse dynamics model online that is used to control the arm following a figure-8 in task space. After few seconds of online learning, the robot is able to follow the desired trajectory (blue thick line). When the robot's dynamics is changed online, e.g., by attaching a water bottle to the arm, the inverse dynamics model can be adapted in real-time [106] resulting in more accurate tracking performance.

the inverted helicopter flight after learning. A few similar examples in model predictive control (see Section 2.1 for an explanation) exist which employ a variety of different learning approaches such as statistical learning methods [68], neural networks [3] both for robot navigation [47] and helicopter control [173].

3.2 Approximation-based Inverse Dynamics Control

Inverse models, such as inverse dynamics models, are frequently used in robotics [150]. Inverse dynamics models characterize the required joint torques $\boldsymbol{\tau} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ to achieve a desired configuration $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, where $\boldsymbol{q}$ is the joint position and $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are the corresponding velocity and acceleration, respectively. In classical robot control, inverse dynamics model can be analytically given by the rigid body dynamics model

$$\boldsymbol{\tau}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\right) = \mathbf{M}\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}} + \mathbf{F}\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) , \tag{9}$$

where $\mathbf{M}(\boldsymbol{q})$ is the generalized inertia matrix of the robot, $\mathbf{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a vector defined by the forces, such as Coriolis forces, centripetal forces and gravity. This model relies on series of assumptions, such as that the robot's links are rigid, there is no friction or stiction, nonlinearities of actuators are negligible etc. [150]. However, modern robotics systems with highly nonlinear components, such as hydraulic tubes or elastic cable drive, can no longer be accurately modeled with the rigid body dynamics. An inaccurate dynamics model can lead to severe losses in control performance and, in the worst case, instability. Instead of modeling the inverse dynamics manually based on physics and human insight, an inverse dynamics model can be learned from sampled data. Such a data-driven approach has the advantage that all nonlinearities encoded in the data will be approximated by the model [105].

As the inverse model is a unique mapping from joint space into torque space, learning inverse dynamics models is a standard regression problem. In order to generalize the learned models for a larger state space and to adapt the models for time dependent changes in the dynamics, real-time online learning becomes necessary. However,

online learning poses difficult challenges for any regression method. These problems have been addressed by real-time learning methods such as locally weighted projection regression [172]. Nguyen-Tuong et al. [105,106] combine the basic ideas behind the locally weighted projection regression method with the global probabilistic Gaussian process regression method [122], attempting to combine the efficiency of local learning with the high accuracy of Gaussian process regression. The resulting method has shown to be capable of real-time online learning of the robot's inverse dynamics. Instead of using local models, data sparsification methods can be employed to speed up kernel regression approaches for real-time learning [106]. Figure 5 shows a Barrett arm while learning to follow a figure-8 in task space.

It is worth noting that such inverse dynamics model learning approaches can also be motivated from a biological point of view. Kawato et al [62] have suggested that the cerebellum may act as an inverse dynamics model. Motivated by this insight, Shibata et al. [144] proposed a biologically inspired vestibulo-oculomotor control approach based on feedback-error learning of the inverse dynamics model. The problem is to stabilize the gaze in the face of perturbations due to body movement, where the cerebellum is known to predict the forces required to keep image stabilized on the retina (based on efferent motor signals and inputs from the vestibular system). In this work, Shibata et al. employ the locally weighted projection regression approach to learn the inverse model of the eye dynamics online [172]. The same locally weighted projection regression technique has also been used to learn a complete inverse dynamics model for the humanoid DB [134].

3.3 Learning Operational Space Control

Operational space control (OSC) allows the robot to follow given desired trajectories in task space [65,93]. Before explaining how OSC can be viewed as a learning problem, we will review the most basic form of OSC laws from a classical robotics point of view.

The relationship between the task space and joint space of the robot is defined by the classical forward kinematics models $\boldsymbol{x} = f(\boldsymbol{q})$, where $\boldsymbol{q}$ denotes a joint space configuration and $\boldsymbol{x}$ represents the corresponding task space position. The task space velocity and acceleration are given by $\dot{\boldsymbol{x}} = \mathbf{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{x}} = \dot{\mathbf{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \mathbf{J}(\boldsymbol{q})\ddot{\boldsymbol{q}}$, respectively, where $\mathbf{J}(\boldsymbol{q}) = \partial f / \partial \boldsymbol{q}$ is the Jacobian. To obtain the joint torques required for the task space control, the dynamics model (as given in Equation (9)) is needed. The direct combination of dynamics and kinematics model yields one possible operational space control law

$$\boldsymbol{u} = \mathbf{M}\mathbf{J}_W^+(\ddot{\boldsymbol{x}} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \mathbf{F}\,, \tag{10}$$

where $\mathbf{J}_W^+$ denotes the weighted pseudo-inverse of $\mathbf{J}$ [141,113] and $\boldsymbol{u}$ represents the joint control torques. Equation (10) can be employed to generate the joint torques necessary for tracking a task space trajectory determined by a reference task-space acceleration [93]. Note that the practical application of such control laws often requires further terms, such as the so-called null-space control law for joint-space stabilization [93].

As discussed in Section 3.2, dynamics models can be hard to obtain and, thus, learning can be an attractive alternative. Learning an operational space control law corresponds to learning an inverse model such as $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{x}}) \rightarrow \boldsymbol{u}$ [114]. However, learning such OSC models is an ill-posed problem, as there are infinitely many inverse models

possible. For example, we could create infinitely many solutions for a redundant robot analytically by simply varying the metric W of the weighted pseudo-inverse in Equation (10). As the space of possible solutions is not convex, such OSC models cannot be learned straightforwardly using regression models (unless the system has no redundant degrees of freedom). Similar problems appear in the limited case of differential inverse kinematics [33].

Both D'Souza et al. [33] and Peters et al. [114] noticed that local linearizations of the mapping in Equation (10) will always form a convex space. Hence, data sets generated by such systems will also be locally convex. They furthermore realized that the predictive abilities of forward models allows determining local regions, where a locally consistent forward model can be learned. However, extremely different and inconsistent local models may form, depending on the local data distribution. As a result, the global consistency can no longer be ensured. This insight leads to two significantly different approaches. D'Souza [33] created a differential inverse kinematics learning system (i.e., a limited special case of an operational space control law) and chose to bias the learning system by selectively generating data. However, he also realized that such an approach will generically be limited by the trade-off between this intentional bias and the inverse model's accuracy. Peters et al. [114] treated learning of complete operational space control laws. They realized that a re-weighting of the data using an additional reward function allows regularizing these inverse models towards a globally consistent solution. Inspired by a result in analytical OSC [113], they suggested appropriate reward functions both for learning full OSC and differential inverse kinematics. The resulting mapping was shown to work on several robot systems. Ting et al. [162] presented an implementation of Peters et al.'s [114] approach with modern Bayesian machine learning which sped up the performance significantly.

Instead of learning a direct OSC control law as done by Peters et al. [114], Salaün et al. [130] attempt to learn the well-defined differential forward kinematics as a first step (i.e., learning the Jacobian) using locally weighted projection regression. The corresponding weighted pseudo-inverse of the Jacobian is subsequently computed using SVD decomposition techniques. The obtained differential inverse kinematics model is combined with an inverse dynamics model to generate the joint space control torques [130]. Approximating inverse kinematics models has also been investigated using neural network learning [58,123]. More recently, Reinhart et al. [123] employ a reservoir computing architecture which allows to jointly learn the forward and inverse kinematics.

4 Future Directions and Conclusion

In this paper, we have surveyed model learning for robotics. In this section, we round this review paper off by giving our perspective on open problems and an outlook on future developments as well as a conclusion.

4.1 Open Problems and Outlook

A key question in robotics is how to deal with uncertainty in the environment. In the context of control and robotics, uncertainty has not yet been investigated extensively despite its central role in risk-sensitive control, sensori-motor control and robust control. Uncertainty is an essential component when dealing with stochastic policies,

incomplete state information and exploration strategies. As probabilistic machine learning techniques (e.g., Bayesian inference) have reached a high level of maturity [122], it has become clear how beneficial Bayesian machine learning may become for robot learning, especially, for model learning in presence of uncertainty. However, machine learning techniques based on Bayesian inference is well-known to suffer from high computational complexity. Thus, special approximations will need to be made as illustrated by the problems discussed in Section 2.3. Recently developed approximate inference methods such as in [21, 20] may become interesting new tools for robotics.

In order for robots to enter everyday life, the robot needs to continuously learn and adapt itself to new tasks. Recent research on learning robot control has predominantly focused on learning single tasks that were studied in isolation. However, there is an opportunity to transfer knowledge between tasks which is known as transfer learning in the field of machine learning [11]. To achieve this goal, robots need to learn the invariants of the individual tasks and environments and, subsequently, exploit them when learning new tasks. Such task-independent knowledge can be employed to make the system more efficient when learning a new task. In this context, similarities between tasks also need to be investigated and how they can be employed to generalize to new tasks. Furthermore, it may be useful to further study the underlying structures between tasks which can help to accelerate the model learning process [167].

In most of the model learning approaches, supervised learning methods are used. However, for many robot applications, target outputs are not always available. One example is learning models for terrain classification using vision features. In that case, exact labeling of the vision features is not always possible and, furthermore, manually labeling such features is susceptive to errors. Here, semi-supervised learning techniques can be useful to learn such models [23]. Semi-supervised learning employs labeled as well as unlabeled data for model learning and can help to overcome the sparse labeling problem. It would also be beneficial to develop online versions of semi-supervised approaches for real-time adaptation and learning.

Approximation based control often still suffers from a lack of proper analysis of stability and convergence properties despite the pioneering work of Nakanishi et al. and Farrell et al. [94, 37]. Here, modern statistical learning theory might offer appropriate tools by using error bounds that enable a general approach to the discussed analysis problems [138]. For example, generalization bounds can be used to estimate the learning performance of the controller and, from this insight, further statements and conditions about stability can be made.

Learning non-unique mappings is a key problem for several robot applications, such as inverse kinematics or operational space control, as pointed out in Section 3.3. This problem can be considered as a non-unique labeling problem, i.e., many input points may have the same target outputs. Here, statistical machine learning techniques, such as conditional random fields [74], may offer a tool to solve the problem. Conditional random fields is a framework for building probabilistic models to segment and label sequence data. A conditional model specifies the probabilities of possible target outputs given an input observation sequence. As the target outputs are conditioned on the (current) input observations. Non-uniqueness in the mappings can be resolved. It would be beneficial to investigate how conditional random fields can be incorporated into learning control to learn an OSC model. Furthermore, it can be interesting to extend such models to a hierarchical control framework, as proposed by Sentis et al. [143].

4.2 Conclusion

In this paper, we give a survey of past and current research activities of model learning for robotics. Model learning is gaining increasing interests in the robotics community, as physically modeling of complex, modern robot systems become more difficult. It can be a useful alternative to manual pre-programming, as the model is estimated directly from measured data. Unknown nonlinearities are taken in account, while they are neglected by the standard physics-based modeling techniques. Model learning has been shown to be an efficient tool in variety of application scenario. Especially, for robot control model learning has proven to be useful, as it provides accurate models of the system allowing the application of compliant, energy-efficient controls. First, we discussed different model learning architectures. Subsequently, we pointed out what kind of problems these architectures and the domain of robotics imply for the learning methods. We further discussed the challenges that arise from the application of learning methods in the domain of robotics. An overview on how models can be learned using machine learning techniques with a focus on statistical regression methods was given. In several case studies, we showed where the model learning scenarios have been used successfully.

References

1. P. Abbeel, A. Coates, M. Quigley, A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in Neural Information Processing Systems*, 2007.
2. H. Akaike. Autoregressive model fitting for control. *Annals of the Institute of Statistical Mathematics*, 23:163–180, 1970.
3. B. M. Akesson, H. T. Toivonen. A neural network model predictive controller. *Journal of Process Control*, 16(9):937–946, 2006.
4. A. Angelova, L. Matthies, D. Helmick, P. Perona. Slip prediction using visual information. *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.
5. K. J. Aström, B. Wittenmark. *Adaptive Control*. Addison Wesley, Boston, MA, USA, 1995.
6. C. G. Atkeson, C. H. An, J. M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. *International Journal of Robotics Research*, 5(3), 1986.
7. C. G. Atkeson, A. W. Moore, S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73, 1997.
8. C. G. Atkeson, A. W. Moore, S. Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11(1-5):75–113, 1997.
9. C. G. Atkeson, J. Morimoto. Nonparametric representation of policies and value functions: A trajectory-based approach. *Advances in Neural Information Processing Systems*, 2002.
10. C. G. Atkeson, S. Schaal. Robot learning from demonstration. *Proceedings of the 14-th International Conference on Machine Learning*, 1997.
11. S. Ben-David, R. Schuller. Exploiting task relatedness for multiple task learning. *Proceedings of the Conference on Learning Theory*, 2003.
12. N. Bhushan, R. Shadmehr. Evidence for a forward dynamics model in human adaptive motor control. *Advances in Neural Information Processing Systems*, 1999.
13. S. S. Billings, S. Chen, G. Korenberg. Identification of mimo nonlinear systems using a forward-regression orthogonal estimator. *International Journal of Control*, 49:2157–2189, 1989.
14. J. Bongard, V. Zykov, H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314:1118–1121, 2006.
15. B. Boots, S. M. Siddiqi, G. J. Gordon. Closing the learning-planning loop with predictive state representations. *Robotics: Science and Systems*, 2010.
16. L. Bottou, O. Chapelle, D. DeCoste, J. Weston. *Large-Scale Kernel Machines*. MIT Press, Cambridge, MA, 2007.

17. E. Burdet, B. Sprenger, A. Codourey. Experiments in nonlinear adaptive control. *International Conference on Robotics and Automation*, 1:537–542, 1997.
18. M. Butz, M. Herbort, J. Hoffmann. Exploiting redundancy for flexible behavior: Unsupervised learning in a modular sensorimotor control architecture. *Psychological Review*, 114(3):1015–1046, 2007.
19. S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, A. Billard. A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation. *IEEE Robotics and Automation Magazine*, 17:44–54, 2010.
20. J. Q. Candela, C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 2005.
21. J. Q. Candela, C. E. Rasmussen, C. K. Williams. *Large Scale Kernel Machines*. MIT-Press, Cambridge, MA, 2007.
22. H. Cao, Y. Yin, D. Du, L. Lin, W. Gu, Z. Yang. Neural network inverse dynamic online learning control on physical exoskeleton. *13th International Conference on Neural Information Processing*, 2006.
23. O. Chapelle, B. Schölkopf, A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
24. Y. Choi, S. Y. Cheong, N. Schweighofer. Local online support vector regression for learning control. *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2007.
25. C. M. Chow, A. G. Kuznetsov, D. W. Clarke. Successive one-step-ahead predictions in multiple model predictive control. *International Journal of Control*, 29:971–979, 1998.
26. W. S. Cleveland, C. L. Loader. Smoothing by local regression: Principles and methods. *Statistical Theory and Computational Aspects of Smoothing*, 1996.
27. D. A. Cohn, Z. Ghahramani, M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.
28. F. J. Coito, J. M. Lemos. A long-range adaptive controller for robot manipulators. *The International Journal of Robotics Research*, 10:684–707, 1991.
29. J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Prentice Hall, New Jersey, 2004.
30. L. Csato, M. Opper. Sparse online gaussian processes. *Neural Computation*, 2002.
31. S. Dasgupta. Analysis of a greedy active learning strategy. *Advances in Neural Information Processing Systems*, 2004.
32. D. Demers, K. Kreutz-Delgado. Learning global direct inverse kinematics. *Advances in Neural Information Processing Systems*, strony 589–595, 1992.
33. A. D'Souza, S. Vijayakumar, S. Schaal. Learning inverse kinematics. *IEEE International Conference on Intelligent Robots and Systems*, 2001.
34. N. U. Edakunni, S. Schaal, S. Vijayakumar. Kernel carpentry for online regression using randomly varying coefficient model. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.
35. Y. Engel, S. Mannor, R. Meir. Sparse online greedy support vector regression. *European Conference on Machine Learning*, 2002.
36. J. Fan, I. Gijbels. *Local Polynomial Modelling and Its Applications*. Chapman and Hall, 1996.
37. J. A. Farrell, M. M. Polycarpou. *Adaptive Approximation Based Control*. John Wiley and Sons, New Jersey, 2006.
38. J. P. Ferreira, M. Crisostomo, A. P. Coimbra, B. Ribeiro. Simulation control of a biped robot with support vector regression. *IEEE International Symposium on Intelligent Signal Processing*, 2007.
39. M. A. F. Figueiredo, A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.
40. M. Gautier, W. Khalil. Exciting trajectories for the identification of base inertial parameters of robots. *International Journal of Robotics Research*, 11(4):362–375, 1992.
41. S. S. Ge, T. H. Lee, E. G. Tan. Adaptive neural network control of flexible joint robots based on feedback linearization. *International Journal of Systems Science*, 29(6):623–635, 1998.
42. R. Genov, S. Chakrabartty, G. Cauwenberghs. Silicon support vector machine with online learning. *International Journal of Pattern Recognition and Articial Intelligence*, 17:385–404, 2003.
43. A. Girard, C. E. Rasmussen, J. Q. Candela, R. M. Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. *Advances in Neural Information Processing Systems*, 2002.

44. P. W. Glynn. Likelihood ratio gradient estimation: an overview. *Proceedings of the 1987 Winter Simulation Conference*, 1987.
45. H. Gomi, M. Kawato. Recognition of manipulated objects by motor learning with modular architecture networks. *Neural Networks*, 6(4):485–497, 1993.
46. D. H. Grollman, O. C. Jenkins. Sparse incremental learning for interactive robot control policy estimation. *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008.
47. D. Gu, H. Hu. Predictive control for a car-like mobile robot. *Robotics and Autonomous Systems*, 39:73–86, 2002.
48. W. K. Haerdle, M. Mueller, S. Sperlich, A. Werwatz. *Nonparametric and Semiparametric Models*. Springer, New York, USA, 2004.
49. M. Haruno, D. M. Wolpert, M. Kawato. Mosaic model for sensorimotor learning and control. *Neural Computation*, 13(10):2201–2220, 2001.
50. T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.
51. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 1999.
52. O. Herbort, M. V. Butz, G. Pedersen. The SURE_REACH model for motor learning and control of a redundant arm: From modeling human behavior to applications in robots. *From motor to interaction learning in robots*, strony 85–106, 2010.
53. H. Hoffman, S. Schaal, S. Vijayakumar. Local dimensionality reduction for non-parametric regression. *Neural Processing Letters*, 2009.
54. M. Hoffmann, H. G. Marques, A. H. Arieta, H. Sumioka, M. Lungarella, R. Pfeifer. Body schema in robotics: A review. *IEEE Transactions on Autonomous Mental Development*, 2(4):304–324, 2010.
55. R. Jacobs, M. Jordan, S. Nowlan, G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.
56. D. H. Jacobson, D. Q. Mayne. *Differential Dynamic Programming*. New York, American Elsevier, 1973.
57. J.Fan, I.Gijbels. Data driven bandwidth selection in local polynomial fitting. *Journal of the Royal Statistical Society*, 57(2):371–394, 1995.
58. I. Jordan, D. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.
59. P. Joshi, W. Maass. Movement generation with circuits of spiking neurons. *Neural Computation*, 17(8):1715–1738, 2005.
60. M. Kalakrishnan, J. Buchli, P. Pastor, S. Schaal. learning locomotion over rough terrain using terrain templates. *IEEE International Conference on Intelligent Robots and Systems*, 2009.
61. M. Kawato. Feedback error learning neural network for supervised motor learning. *Advanced Neural Computers*, 1990.
62. M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, 1999.
63. R. D. Keyser, A. V. Cauwenberghe. A self-tuning multistep predictor application. *Automatica*, 17:167–174, 1980.
64. W. Khalil, E. Dombre. *Modeling, Identification and Control of Robots*. Taylor & Francis, Inc., Bristol, PA, USA, 2002.
65. O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Journal of Robotics and Automation*, 3(1):43–53, 1987.
66. S. Klanke, D. Lebedev, R. Haschke, J. J. Steil, H. Ritter. Dynamic path planning for a 7-dof robot arm. *Proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems*, 2006.
67. J. Ko, D. Fox. GP-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.
68. J. Kocijan, R. Murray-Smith, C. Rasmussen, A. Girard. Gaussian process model based predictive control. *Proceeding of the American Control Conference*, 2004.
69. M. Kopicki. *Prediction learning in robotic manipulation*. Praca doktorska, University of Birmingham, December 2010.
70. M. Kopicki, S. Zurek, R. Stolkin, T. Morwald, J. Wyatt. Learning to predict how rigid objects behave under simple manipulation,. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, 2011.

71. O. Kroemer, R. Detry, J. Piater, J. Peters. Active learning using mean shift optimization for robot grasping. *International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, 2009.
72. B. J. Kröse, N. Vlassis, R. Bunschoten, Y. Motomura. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, 19:381–391, 2001.
73. E. Krupka, N. Tishby. Incorporating prior knowledge on features into learning. *International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 2007.
74. J. D. Lafferty, A. McCallum, F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning*, 2001.
75. J. R. Layne, K. M. Passino. Fuzzy model reference learning control. *Journal of Intelligent and Fuzzy Systems*, 4:33–47,, 1996.
76. M. Littman, R. S. Sutton, S. Singh. Predictive representations of state. *Advances In Neural Information Processing Systems*, 2001.
77. L. Ljung. *System Identification - Theory for the User*. Prentice-Hall, New Jersey, 2004.
78. M. Lopes, B. Damas. A learning framework for generic sensory-motor maps. *Proceedings of the International Conference on Intelligent Robots and Systems*, 2007.
79. A. D. Luca, P. Lucibello. A general algorithm for dynamic feedback linearization of robots with elastic joints. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.
80. M. Lukocevicius, H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.
81. J. Ma, J. Theiler, S. Perkins. Accurate on-line support vector regression. *Neural Computation*, 15:2683–2703, 2005.
82. J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, New Jersey, 2002.
83. D. J. MacKay. A practical Bayesian framework for back-propagation networks. *Neural Computation*, 4(3):448–472, 1992.
84. R. Martinez-Cantin, O. D. Freitas, A. Doucet, J. A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. *In Proceedings of Robotics: Science and Systems*, 2007.
85. R. Martinez-Cantin, M. Lopes, L. Montesano. Body schema acquisition through active learning. *IEEE International Conference on Robotics and Automation*, 2010.
86. W. T. Miller III. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on System, Man and Cybernetics*, 19(4):825–831, 1989.
87. W. T. Miller III, F. H. Glanz, L. G. Kraft III. Application of a general learning algorithm to the control of robotic manipulators. *International Journal of Robotics Research*, 6(2):84–98, 1987.
88. H. Miyamoto, M. Kawato, T. Setoyama, R. Suzuki. Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, 1(3):251–265, 1988.
89. A. Moore. Fast, robust adaptive control by learning only forward models. *Advances in Neural Information Processing Systems*, 1992.
90. A. Moore, M. S. Lee. Efficient algorithms for minimizing cross validation error. *Proceedings of the 11th International Confonference on Machine Learning*, 1994.
91. J. Morimoto, G. Zeglin, C. G. Atkeson. Minimax differential dynamic programming: Application to a biped walking robot. *Proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems*, 2003.
92. E. Mosca, G. Zappa, J. M. Lemos. Robustness of multipredictor adaptive regulators: MUSMAR. *Automatica*, 25:521–529, 1989.
93. J. Nakanishi, R. Cory, M. Mistry, J. Peters, S. Schaal. Operational space control: a theoretical and emprical comparison. *International Journal of Robotics Research*, 27(6):737–757, 2008.
94. J. Nakanishi, J. A. Farrell, S. Schaal. Composite adaptive control with locally weighted statistical learning. *Neural Networks*, 18(1):71–90, 2005.
95. J. Nakanishi, S. Schaal. Feedback error learning and nonlinear adaptive control. *Neural Networks*, 17(10), 2004.
96. H. Nakayama, Y. Yun, M. Shirakawa. Multi-objective model predictive control. *Proceedings of the 19th International Conference on Multiple Criteria Decision Making*, 2008.

97. K. Narendra, J. Balakrishnan. Adaptive control using multiple models. *IEEE Transaction on Automatic Control*, 42(2):171–187, 1997.
98. K. Narendra, J. Balakrishnan, M. Ciliz. Adaptation and learning using multiple models, switching and tuning. *IEEE Control System Magazin*, 15(3):37–51, 1995.
99. K. S. Narendra, A. M. Annaswamy. Persistent excitation in adaptive systems. *International Journal of Control*, 45:127–160, 1987.
100. K. S. Narendra, A. M. Annaswamy. *Stable Adaptive Systems*. Prentice Hall, New Jersey, 1989.
101. R. M. Neal. Bayesian learning for networks. *Lecture Notes in Statistics*, 1996.
102. R. Negenborn, B. D. Schutter, M. A. Wiering, H. Hellendoorn. Learning-based model predictive control for markov decision processes. *Proceedings of the 16th IFAC World Congress*, 2005.
103. A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Proceedings of the 11th International Symposium on Experimental Robotics*, 2004.
104. A. Y. Ng, M. Jordan. Pegasus: A policy search method for large mdps and pomdps. *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, 2000.
105. D. Nguyen-Tuong, J. Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.
106. D. Nguyen-Tuong, J. Peters. Incremental sparsification for real-time online model learning. (in press). *Neurocomputing*, 2010.
107. D. Nguyen-Tuong, J. Peters. Using model knowledge for learning inverse dynamics. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, 2010.
108. S. Nicosia, P. Tomei. Model reference adaptive control algorithms for industrial robots. *Automatica*, 20:635–644, 1984.
109. S. Nowlan, G. E. Hinton. Evaluation of adaptive mixtures of competing experts. *Advances in Neural Information Processing Systems*, 1991.
110. K. Otani, T. Kakizaki. Motion planning and modeling for accurately identifying dynamic parameters of an industrial robotic manipulator. *International Symposium on Industrial Robots*, 1993.
111. H. D. Patino, R. Carelli, B. R. Kuchen. Neural networks for advanced control of robot manipulators. *IEEE Transactions on Neural Networks*, 13(2):343–354, 2002.
112. R. Pelossof, A. Miller, P. Allen, T. Jebara. An svm learning approach to robotic grasping. *In IEEE International Conference on Robotics and Automation*, 2004.
113. J. Peters, M. Mistry, F. E. Udwadia, J. Nakanishi, S. Schaal. A unifying methodology for robot control with redundant DoFs. *Autonomous Robots*, 24(1):1–12, 2008.
114. J. Peters, S. Schaal. Learning to control in operational space. *International Journal of Robotics Research*, 27(2):197–212, 2008.
115. G. Petkos, M. Toussaint, S. Vijayakumar. Learning multiple models of non-linear dynamics for control under varying contexts. *Proceedings of the International Conference on Artificial Neural Networks*, 2006.
116. C. Plagemann, K. Kersting, P. Pfaff, W. Burgard. Heteroscedastic gaussian process regression for modeling range sensors in mobile robotics. *Snowbird learning workshop*, 2007.
117. C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, W. Burgard. Learning predictive terrain models for legged robot locomotion. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2008.
118. J. Porrill, P. D. P, J. V. Stone. Recurrent cerebellar architecture solves the motor-error problem. *Proceedings Royal Society (B)*, 2004.
119. C. E. Rasmussen. Evaluation of gaussian processes and other methods for non-linear regression. *University of Toronto*, 1996.
120. C. E. Rasmussen, Z. Ghahramani. Infinite mixtures of gaussian process experts. *Advances in Neural Information Processing Systems*, 2002.
121. C. E. Rasmussen, M. Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, 2003.
122. C. E. Rasmussen, C. K. Williams. *Gaussian Processes for Machine Learning*. MIT-Press, Massachusetts Institute of Technology, 2006.
123. R. F. Reinhart, J. J. Steil. Recurrent neural associative learning of forward and inverse kinematics for movement generation of the redundant pa-10 robot. *Symposium on Learning and Adaptive Behavior in Robotic Systems*, 2008.

124. R. F. Reinhart, J. J. Steil. Attractor-based computation with reservoirs for online learning of inverse kinematics. *Proceedings of the European Symposium on Artificial Neural Networks*, 2009.
125. R. F. Reinhart, J. J. Steil. Reaching movement generation with a recurrent neural network based on learning inverse kinematics. *Proceedings of the Conference on Humanoid Robots*, 2009.
126. M. Rolf, J. J. Steil, M. Gienger. Efficient exploration and learning of whole body kinematics. *Proceedings of the International Conference on Development and Learning*, 2010.
127. M. Rolf, J. J. Steil, M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.
128. A. Rottmann, W. Burgard. Adaptive autonomous control using online value iteration with gaussian processes. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.
129. S. Roweis, L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.
130. C. Salaun, V. Padois, O. Sigaud. Control of redundant robots using learned models: an operational space control approach. *Proceedings of the 2009 IEEE International Conference on Intelligent Robots and Systems*, 2009.
131. T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(36):459–473, 1989.
132. S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 1999.
133. S. Schaal, C. G. Atkeson. Learning control in robotics: Trajectory-based optimal control techniques. *IEEE Robotics and Automation Magazine*, 2010.
134. S. Schaal, C. G. Atkeson, S. Vijayakumar. Scalable techniques from nonparametric statistics for real-time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.
135. S. Schaal, D. Sternad. Programmable pattern generators. *International Conference on Computational Intelligence in Neuroscience*, 1998.
136. B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.
137. B. Schölkopf, P. Simard, A. Smola, V. Vapnik. Prior knowledge in support vector kernel. *Advances in Neural Information Processing Systems*, Denver,CO, USA, 1997.
138. B. Schölkopf, A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT-Press, Cambridge, MA, 2002.
139. B. Schölkopf, A. Smola, R. Williamson, P. Bartlett. New support vector algorithms. *Neural Computation*, 12(5), 2000.
140. B. Schrauwen, D. Verstraeten, J. V. Campenhout. An overview of reservoir computing: Theory, applications and implementations. *Proceedings of the 15th European Symposium on Artificial Neural Networks*, strony 471–482, 2007.
141. L. Sciavicco, B. Siciliano. *Modeling and Control of Robot Manipulators*. McGraw-Hill, New York, 1996.
142. M. Seeger. Gaussian processes for machine learning. *International Journal of Systems*, 2004.
143. L. Sentis, O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4):505–518, 2005.
144. T. Shibata, C. Schaal. Biomimetic gaze stabilization based on feedback-error learning with nonparametric regression networks. *Neural Networks*, 14(2):201–216, 2001.
145. D. Skočaj, M. Kristan, A. Vrečko, A. Leonardis, M. Fritz, M. Stark, B. Schiele, S. Hongeng, J. L. Wyatt. Multi-modal learning. *Cognitive Systems*, 8:265–309, 2010.
146. J.-J. E. Slotine, W. Li. *Applied Nonlinear Control*. Prentice Hall, New Jersey, 1991.
147. O. J. Smith. A controller to overcome dead-time. *Instrument Society of America Journal*, 6:28–33, 1959.
148. A. Smola, T. Friess, B. Schoelkopf. Semiparametric support vector and linear programming machines. *Advances in Neural Information Processing Systems*, Denver,CO, USA, 1998.
149. A. J. Smola, B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.
150. M. W. Spong, S. Hutchinson, M. Vidyasagar. *Robot Dynamics and Control*. John Wiley and Sons, New York, 2006.

151. J. Steffen, S. Klanke, S. Vijayakumar, H. J. Ritter. Realising dextrous manipulation with structured manifolds using unsupervised kernel regression with structural hints. *ICRA 2009 Workshop: Approaches to Sensorimotor Learning on Humanoid Robots*, Kobe, Japan, 2009.
152. J. J. Steil. Backpropagation-decorrelation: online recurrent learning with O(n) complexity. *Proceedings of the International Joint Conference on Neural Networks*, Jul 2004.
153. J. J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Networks*, 20(3):353–364, 2007.
154. M. Stilman, J. J. Kuffner. Planning among movable obstacles with artificial constraints. *International Journal of Robotics Research*, 27(12):1295–1307, 2008.
155. J. Sturm, C. Plagemann, W. Burgard. Unsupervised body scheme learning through self-perception. *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008.
156. R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2(4):160–163, 1991.
157. J. Swevers, C. Ganseman, D. Tükel, J. D. Schutter, H. V. Brussel. Optimal robot excitation and identification. *IEEE Transaction on Robotics and Automation*, 13:730–740, 1997.
158. J. Tenenbaum, V. de Silva, J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.
159. G. Tevatia, S. Schaal. Efficient inverse kinematics algorithms for high-dimensional movement systems. *University of Southern California*, 2008.
160. S. Thrun, T. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 1995.
161. J. Ting, A. D'Souza, S. Schaal. A bayesian approach to nonlinear parameter identification for rigid-body dynamics. *Neural Networks*, 2009.
162. J. Ting, M. Kalakrishnan, S. Vijayakumar, S. Schaal. Bayesian kernel shaping for learning control. *Advances in Neural Information Processing Systems*, 2008.
163. M. K. Titsias, N. D. Lawrence. Bayesian gaussian process latent variable model. *Proceedings of the 13th International Conference on Articial Intelligence and Statistics*, 2010.
164. M. Toussaint, S. Vijayakumar. Learning discontinuities with products-of-sigmoids for switching between local models. *Proceedings of the 22nd International Conference on Machine Learning*, 2005.
165. V. Treps. A bayesian committee machine. *Neural Computation*, 12(11):2719 – 2741, 2000.
166. V. Treps. Mixtures of gaussian process. *Advances in Neural Information Processing Systems*, 2001.
167. I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.
168. S. Ulbrich, V. Angulo, T. Asfour, C. Torras, R. Dillmann. Rapid learning of humanoid body schemas with kinematic bezier maps. *International Conference on Humanoid Robots*, 2009.
169. R. Urtasun, T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. *International Conference in Computer Vision and Pattern Recognition*, Anchorage, Alaska, 2008.
170. P. Vempaty, K. Cheok, R. Loh. Model reference adaptive control for actuators of a biped robot locomotion. *Proceedings of the World Congress on Engineering and Computer Science*, 2009.
171. S. Vijayakumar, A. D'Souza, S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 12(11):2602 – 2634, 2005.
172. S. Vijayakumar, 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.
173. E. A. Wan, A. A. Bogdanov. Model predictive neural control with applications to a 6 dof helicopter model. *Proceedings of the 2001 American Control Conference*, 2001.
174. M. Weber, M. Welling, P. Perona. Unsupervised learning of models for recognition. *Proceedings of the 6th European Conference on Computer Vision*, strony 18–32, 2000.
175. D. M. Wolpert, M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.
176. D. M. Wolpert, R. C. Miall, M. Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2, 1998.