Neurocomputing 72 (2009) 3704–3712

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Variational Bayesian learning of nonlinear hidden state-space models for model predictive control

Tapani Raiko *, Matti Tornio

Adaptive Informatics Research Centre, Department of Information and Computer Science, Helsinki University of Technology, Finland

ARTICLE INFO

Article history:
Received 26 October 2007
Received in revised form
15 May 2009
Accepted 19 June 2009
Communicated by D. Barber
Available online 3 August 2009

Keywords:
Nonlinear system
State-space method
Stochastic optimal control
Neural network
Variational methods
Partially observable Markov decision process
Model predictive control

ABSTRACT

This paper studies the identification and model predictive control in nonlinear hidden state-space models. Nonlinearities are modelled with neural networks and system identification is done with variational Bayesian learning. In addition to the robustness of control, the stochastic approach allows for various control schemes, including combinations of direct and indirect controls, as well as using probabilistic inference for control. We study the noise-robustness, speed, and accuracy of three different control schemes as well as the effect of changing horizon lengths and initialisation methods using a simulated cart–pole system. The simulations indicate that the proposed method is able to find a representation of the system state that makes control easier especially under high noise.

1. Introduction

Learning is extremely important for control of complex systems [2] ranging from anthropomorphic robots [23] over military aircrafts in hazardous environments [10] to road traffic optimisation [28]. Nonlinear control is difficult even in the case that the system dynamics are known. If the dynamics are not known, the traditional approach is to make a model of the dynamics (system identification) and then try to control the simulated model (nonlinear model predictive control). The model learned from data is of course not perfect, but these imperfections are often ignored. The modern view of control sees feedback as a tool for uncertainty management [20], but managing it already in the modelling might have advantages. For instance, the controller can avoid regions where the confidence in model is not high enough [17].

The idea of learning probabilistic nonlinear state-space models for control is not new. The theory and different phenomena are already well covered in [7]. What has changed, though, is the range of models that can be used in practice, due to developments in Bayesian learning theory and computer performance. The issue remains challenging, i.e. investigation in [8] tells that observability and controllability properties of these systems cannot be easily deduced.

Recently, Rosenqvist and Karlström [29] presented a method for system identification and control in nonlinear hidden state-space models. This is the same as partially observable Markov decision process (POMDP) with continuous-valued states and observations. The nonlinearities are modelled as piecewise linear (or affine). The system identification is based on the prediction error method. In probability theory, this corresponds to a maximum likelihood estimate assuming a Gaussian process noise. This is the most closely related work to ours that basically applies only more sophisticated methods from the machine learning community to the control problem.

Our method of choice, nonlinear dynamical factor analysis (NDFA) [35] is a state-of-the-art tool for finding nonlinear state-space models with variational Bayesian learning. The state vector is completely hidden just like in hidden Markov models, and the relationship between the state and the observation must be learned from data. In NDFA, the parameters, the hidden states, and the observations are real-valued vectors that are modelled with parametrised probability distributions. Uncertainties from noisy observations and model imperfections are thus taken explicitly into account. Nonlinearities are modelled with multilayer perceptron (MLP) networks. Variational learning has many benefits compared to the maximum likelihood method. It is less prone to overfitting and can be used for selecting the model structure, e.g. the dimensionality of the state space.

* Corresponding author. Tel.: +358 505225750; fax: +358 94513277.
E-mail address: tapani.raiko@tkk.fi (T. Raiko).
URL: http://www.cis.hut.fi/projects/bayes/ (T. Raiko).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.06.009

One of the oldest nonlinear system identification by machine learning methods is [19] where predictions of future observations are done by fitting a mapping (a function approximator) to triplets of observations, control signals, and future observations. Then actions can be selected based on predictions of future observations. Similar model based predictive control has been done with Gaussian processes in [16] where the inputs of the mapping include the previous observations and control signals up to a given lag. In [36,30], a mapping is learned from observations and the goal to desired control signals directly. In [12], learning is done in a stochastic setting. In all of these methods, the mapping is learned between fully observed entities, while our approach estimates a hidden state and thus a richer model for the dynamics of the system.

Dynamic programming can be used to solve complex control problems in two different ways: Atkeson et al. [4,5] used dynamic programming in the state space by discretising the state space into cells and solving locally optimal trajectories for each cell. Local trajectories are combined into the globally optimal trajectory by dynamic programming, that is, propagating information between neighbouring cells in the state space. This is a good approach for systems with few dimensions but does not scale up since the number of cells grows exponentially with the number of dimensions. For this reason, this approach is not used in this work.

The other approach is to do the dynamic programming in system time, leading to model predictive control. Future observations are predicted up to some horizon in the future by propagating predictions through time, compare them to the reference signal, and update the future control signals by minimising an objective function by propagating the error backwards through time. Kocijan et al. [16] use Gaussian processes for model predictive control. Their model of system dynamics is linear and the state is fully observed. Kappen [14] considers a problem where one of the goal states must be reached at a certain time. The solution is worked out backwards in time from the goal state and by solving the optimal control signal for each possible state at each time.

The rest of the paper is structured as follows: In Section 2, the nonlinear state-space estimator is reviewed and in Section 3 its use as a controller is presented. After experiments in Section 4 matters are discussed and concluded.

2. Variational Bayesian learning of nonlinear hidden state-space models

Selecting actions based on a state-space model instead of the observation directly has many benefits: firstly, it is more resistant to noise because it implicitly involves filtering. Secondly, the observations (without history) do not always carry enough information about the system state. Thirdly, when nonlinear dynamics are modelled by a function approximator such as an multilayer perceptron (MLP) network, a state-space model can find a representation of the state that is more suitable for the approximation and thus more predictable.

Nonlinear dynamical factor analysis (NDFA) [35] is a powerful tool for system identification. It is based on a nonlinear hidden state-space model learned in a variational Bayesian setting. The computational complexity of NDFA scales only quadratically with the dimensionality of the observation space, so it is also suitable for modelling systems with fairly high dimensionality [35]. MLP networks are a good choice of function approximation when no prior information about the system is available. The structure or model equation of MLP networks contains very little information, instead, by adjusting the parameters of the network, the produced function may take practically any shape.

In our model, the observation (or measurement) vector $\mathbf{y}(t)$ is assumed to have been generated from the hidden state vector $\mathbf{x}(t)$ driven by the control $\mathbf{u}(t)$ by the following generative model:

$$\begin{bmatrix}\mathbf{u}(t)\\ \mathbf{x}(t)\end{bmatrix} = \mathbf{g}\left(\begin{bmatrix}\mathbf{u}(t-1)\\ \mathbf{x}(t-1)\end{bmatrix}, \boldsymbol{\theta}_{\mathbf{g}}\right) + \mathbf{v}(t), \tag{1}$$

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}_{\mathbf{f}}) + \mathbf{w}(t), \tag{2}$$

where $\boldsymbol{\theta}$ is a vector containing the model parameters and time t is discrete. The process noise $\mathbf{v}(t)$ and the measurement noise $\mathbf{w}(t)$ are assumed to be independent, Gaussian, and white. During learning, only the observations $\mathbf{y}$ and the control signals $\mathbf{u}$ are known beforehand, whereas both the states $\mathbf{x}$ and the mappings $\mathbf{f}$ and $\mathbf{g}$ are learned from the data. In the context of system identification this model can be considered task-oriented identification because of its internal forward model to predict $\mathbf{u}(t)$ (see Fig. 1). Note that the uncertainty of the process noise $\mathbf{v}(t)$ leaves the exact selection of the control signal $\mathbf{u}(t)$ open.

Multilayer perceptron (MLP) networks [11] are well suited to modelling both strong and mild nonlinearities. The MLP network models for $\mathbf{f}$ and $\mathbf{g}$ are

$$\mathbf{g}(\mathbf{x}(t), \boldsymbol{\theta}_{\mathbf{g}}) = \mathbf{x}(t) + \mathbf{B}\tanh[\mathbf{A}\mathbf{x}(t) + \mathbf{a}] + \mathbf{b}, \tag{3}$$

$$\mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}_{\mathbf{f}}) = \mathbf{D}\tanh[\mathbf{C}\mathbf{x}(t) + \mathbf{c}] + \mathbf{d}, \tag{4}$$

where the sigmoidal tanh nonlinearity is applied component-wise to its argument vector. The parameters $\boldsymbol{\theta}$ include: (1) the weight matrices $\mathbf{A}, \ldots, \mathbf{D}$, the bias vectors $\mathbf{a}, \ldots, \mathbf{d}$; (2) the parameters of the distributions of the noise signals $\mathbf{w}(t)$ and $\mathbf{v}(t)$ and the column vectors of the weight matrices; (3) the hyperparameters describing the distributions of biases and the parameters in group (2). The priors for these parameters are Gaussian (see [35, Eqs. (16)–(28)]).

In Bayesian learning of generative models, the model is presented such that one could generate observations from it by first drawing parameters from their prior distributions, and then drawing hidden states and observations from the model equations given the parameters. There are infinitely many values for the parameters and hidden states that can explain any given data. Bayesian treatment uses all the possible explanations weighted by their posterior probability. The posterior probability $p(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y}, \mathbf{u})$ of the states and the parameters after observing the data, contains all the relevant information about them. Variational Bayesian learning is a way to approximate the intractable posterior density by a tractable parametric distribution $q(\mathbf{x}, \boldsymbol{\theta})$. The misfit is measured by a cost function based on Kullback–Leibler divergence:

$$C_{\mathrm{KL}} = \int q(\mathbf{x}, \boldsymbol{\theta}) \log \frac{q(\mathbf{x}, \boldsymbol{\theta})}{p(\mathbf{y}, \mathbf{u}, \mathbf{x}, \boldsymbol{\theta})}\, d\boldsymbol{\theta}\, d\mathbf{x}. \tag{5}$$

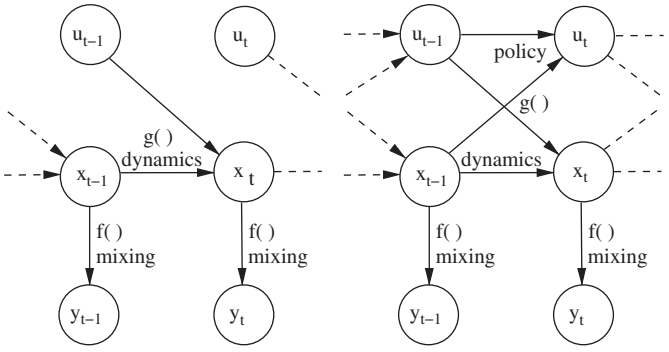

Fig. 1. Traditional model (left) and task-oriented identification (right). Traditionally, the control signals $\mathbf{u}(t)$ are coming from outside the model, but in task-oriented identification they are within the model.

The closer q is to the true Bayesian posterior, the smaller the cost function.

The joint probability density defined by the model can be factorised to a product

$$p(\mathbf{y},\mathbf{u},\mathbf{x},\boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{u},\mathbf{x},\boldsymbol{\theta})p(\mathbf{u},\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{6}$$

$$p(\mathbf{u},\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{u}(1),\mathbf{x}(1)|\boldsymbol{\theta}) \prod_{t=2}^{T} p(\mathbf{u}(t),\mathbf{x}(t)|\mathbf{u}(t-1),\mathbf{x}(t-1),\boldsymbol{\theta}), \tag{7}$$

$$p(\mathbf{y}|\mathbf{u},\mathbf{x},\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\mathbf{y}(t)|\mathbf{u}(t),\mathbf{x}(t),\boldsymbol{\theta}), \tag{8}$$

where individual factors come directly from the model equations (3) and (4), and the priors).

The approximation q needs to be simple for mathematical tractability and computational efficiency. Variables are assumed to depend on each other in the following way:

$$q(\mathbf{x},\boldsymbol{\theta}) = \prod_{t=1}^{T}\prod_{i=1}^{m} q(x_i(t)|x_i(t-1)) \prod_j q(\theta_j), \tag{9}$$

where m is the dimensionality of the state space $\mathbf{x}$. Furthermore, q is assumed to be Gaussian. To summarise, the distribution q is parametrised by the means and the variances of the unknown states and model parameters, and covariances of consecutive state components. They are called *variational parameters*. Now that both p and q are written in a factorised form, the cost function (5) becomes a sum of relatively simple terms, which can be computed. The same applies to the derivatives of the cost function w.r.t. variational parameters.

The only difficult part of evaluating the cost function is about handling the terms $p(\mathbf{u}(t),\mathbf{x}(t)|\mathbf{u}(t-1),\mathbf{x}(t-1),\boldsymbol{\theta})$ and $p(\mathbf{y}(t)|\mathbf{u}(t),\mathbf{x}(t),\boldsymbol{\theta})$. They require the propagation of Gaussian distributions through the nonlinear mappings $\mathbf{g}$ and $\mathbf{f}$, which cannot be done analytically. We used the approximation [13] where the linear parts of the MLP networks are treated accurately, and an unscented transformation is used for the tanh nonlinearity.

Inference (or state estimation) happens by adjusting variational parameters of the hidden states in q such that the cost function C_{KL} is minimised. Learning (or system identification) happens by adjusting the variational parameters of both the hidden states and the model parameters in q minimising C_{KL}. The NDFA package contains an algorithm based on conjugate gradient descent for that.

A proper initialisation of the states is important for the model to learn the dynamic mapping of the system and minimises the risk that the model gets stuck in a local minimum during early training. In the early phase of learning, the data vectors are embedded with delayed versions of itself using time delays of 1, 2, 4, 8, and 16. This promotes the state vector and mapping $\mathbf{f}$ finding a representation of the dynamics even when the dynamic mapping $\mathbf{g}$ has not been learned. The states are initialised to the principal components of the embedded data.

The cost function C_{KL} can also be used for model comparison, for instance to choose among models learned from different initialisations or with different dimensionalities of the state space.

3. Control schemes

In this section, three different control schemes are presented. First we describe direct control (DC), then nonlinear model-predictive control (NMPC) in variational Bayesian setting, and finally the optimistic inference control (OIC) scheme. A summary is given in Table 1. Different strategies for generating initialisations for model-predictive control are also explored.

Table 1
Control scheme summary.

Scheme	Based on	Data	Complexity
DC	Internal MLP	Task-oriented	Low
OIC	Probabilistic inference	General	High
NMPC	Cost minimisation	General	High

3.1. Direct control (DC)

In direct control schemes, the neural network itself acts as the controller. Many such schemes exists, including direct inverse control, optimal control, and feedforward control [22]. Direct control can only mimic the control done in the data that has been used for learning. It therefore requires examples of correct control aiming at the same goal. Direct control works by fitting a function approximator to estimate the mapping from the state to control. It has been used successfully for very complex systems [30].

Eq. (1) provides a prediction of the control signal $\mathbf{u}(t_0)$ based on the previous control signal $\mathbf{u}(t_0-1)$ and the previous estimate of the hidden state $\mathbf{x}(t_0-1)$. The prediction mapping is called the policy in Fig. 1. A control method that we simply call direct control (DC), chooses the control signal by collapsing the inferred probability distribution $q(\mathbf{u}(t_0))$ to its expected value. When the control signal $\mathbf{u}(t_0)$ is selected and the observation $\mathbf{y}(t_0)$ is made, the two probability distributions collapse and these changes affect the estimates of the states $\mathbf{x}(t)$ that are then re-inferred. This works as the error feedback mechanism.

DC in a nutshell:
Given observations $\ldots,\mathbf{y}(t_0-2),\mathbf{y}(t_0-1)$ and control signals $\ldots,\mathbf{u}(t_0-2),\mathbf{u}(t_0-1)$
1: Infer $\mathbf{x}(t_0-1)$
2: Compute $\left[\begin{smallmatrix}\mathbf{u}(t_0)\\ \mathbf{x}(t_0)\end{smallmatrix}\right]$ from Eq. (1)
3: Use the mean of $\mathbf{u}(t_0)$ as the control signal.
4: Observe $\mathbf{y}(t_0)$ and infer $\mathbf{x}(t_0)$
5: Increase t_0 by 1 and loop from 2

This approach is very similar to the one used in [30], where a system was learning the control tasks of devil-sticking, pole balancing, and inverse dynamics of humanoid robots. Instead of MLP-networks, locally linear function approximators were used, and instead of variational Bayes, the learning was based on cross-validation. They used a number of different versions for function approximation, starting from nearest neighbour methods evolving towards a more neural network type approximator. Their implementation has some nice properties that are still missing here, like automatically increasing the model complexity during learning and efficient search of the state-action space for good new commands. Application of [30] to robotics is given in [24]. These approaches assume that the state vector is observed (or $\mathbf{x}(t)=\mathbf{y}(t)$ in our notation), and it is discussed that a better representations of the state would be valuable.

3.2. Nonlinear model predictive control (NMPC)

Nonlinear model predictive control (NMPC) [18] is based on choosing control signal $\mathbf{u}(t_0),\ldots,\mathbf{u}(t_0+T_c-1)$ such that they minimise a cost function J defined over a future window of fixed

length T_c. For example, the quadratic difference between the predicted future observations $\mathbf{y}$ and a known reference signal $\mathbf{r}(t)$ can be used

$$J(\mathbf{y}(t_0), \mathbf{u}(t_0), \ldots, \mathbf{u}(t_0 + T_c - 1)) = \sum_{\tau=1}^{T_c} |\mathbf{y}(t_0 + \tau) - \mathbf{r}(t)|^2. \quad (10)$$

Then J is minimised w.r.t. the control signals $\mathbf{u}$ and the first one $\mathbf{u}(t_0)$ is executed. Direct analogy to decision theory is revealed when the control cost J is interpreted as negative utility.

Here, the states and observations (but not control signals) are modelled probabilistically so we minimise the expected cost $E_q\{J\}$ [7]. The current guess $\mathbf{u}(t_0), \ldots, \mathbf{u}(t_0 + T_c - 1)$ defines a probability distribution over future states and observations. This inference can be done with a single forward pass, when ignoring the internal forward model, that is, the dependency of the state on future control signals. In this case, it makes sense to ignore the forward model anyway, since the future control signals do not have to follow the learned policy.

Minimisation of $E_q\{J\}$ is done here with a quasi-Newton algorithm [21]. For that, the partial derivatives $Y^{t_2} = \partial \mathbf{y}(t_2)/\partial \mathbf{u}(t_1)$ for all $t_0 \le t_1 < t_2 \le t_0 + T_c$ must be computed. For a single input system we can simply apply the chain rule to arrive at the Jacobian

$$Y^{t_2} = F^{t_2} G^{t_2 - 1} \cdots G^{t_1 + 1} G^{t_1}, \quad (11)$$

where F^t and G^t are the Jacobians of the mappings f and g at the the time instant t. Dynamic programming can be used to efficiently compute these partial derivatives for multiple values of t_1 and t_2 in linear time using a backward pass. The extension of this result to multi-input systems is also relatively straightforward.

NMPC in a nutshell:
Given observations $\ldots, \mathbf{y}(t_0 - 2), \mathbf{y}(t_0 - 1)$ and control signals $\ldots, \mathbf{u}(t_0 - 2), \mathbf{u}(t_0 - 1)$
1: Fix the control signals $\{\mathbf{u}(t)\}_{t=t_0}^{t_0+T_c-1}$
2: Infer the distribution $q(\mathbf{x}(t), \mathbf{y}(t))$ for all t (forward pass)
3: Update $\{\mathbf{u}(t)\}_{t=t_0}^{t_0+T_c-1}$ such that the cost J decreases (backward pass)
4: Loop from 2 a few times
5: Use $\mathbf{u}(t_0)$ as the control signal
6: Observe $\mathbf{y}(t_0)$
7: Increase t_0 by 1 and loop from 1

The use of a cost function J makes NMPC very versatile. Costs for control signals and observations can be set for instance to restrict values within bounds. The reference signal $\mathbf{r}(t)$ can either simply be a constant at the desired goal, or depend on time, that is, to follow a given trajectory. There can be a cost for the control signal $\mathbf{u}$ as well, perhaps to minimise the use of energy.

Expectations over a quadratic cost (Eq. (10)) are easy to evaluate because

$$E_q\{|\mathbf{y}(t) - \mathbf{r}(t)|^2\} = |E_q\{\mathbf{y}(t)\} - \mathbf{r}(t)|^2 + \sum_{i=1}^{n} \mathrm{Var}_q\{y_i(t)\},$$

where n is the dimensionality of the observation space $\mathbf{y}$ and $\mathrm{Var}_q\{\cdot\}$ is the variance over the distribution q. The two terms are called the nominal and stochastic part of the cost function. There is a direct analogy with dual control [3] which means balancing between good control and small estimation errors. The usefulness of the decomposition is discussed in [7].

3.3. Optimistic inference control

Optimistic inference control (OIC) as described by us in [25,6] independently, works as follows. Assume that after a fixed delay T_c, the desired goal is reached. That is, (some components of) the observations $\mathbf{x}$ are at the desired level $\mathbf{r}$. Given this optimistic assumption and the observations and control signals so far, infer what happens in between. Then choose the expectation of $q(\mathbf{u}(t_0))$. An example situation is illustrated in Fig. 2.

OIC propagates information in two directions, forwards from the current state and, additionally, the evidence backwards from the desired future. The inference is conceptually simple, but algorithmically difficult. The information from the future needs to flow through tens of nonlinear mappings $\mathbf{g}$ before it affects $\mathbf{u}(t_0)$. The OIC algorithm as presented in [25] only propagates information one step forward and backward in time for each iteration. To speed up this process, total derivatives described in [26] are used to replace the partial derivatives, which leads to much faster propagation of information based on a forward and a backward pass. Another alternative for fast inference is the extended Kalman smoother [1], which corresponds to belief propagation or the E-step of the EM algorithm applied to the linearised model. The linearisation is done locally around the current state estimate for each time point t and it is iteratively updated after each forward–backward update. Extended Kalman smoother unfortunately suffers from stability issues and it is therefore only used to initialise the OIC algorithm.

OIC in a nutshell:
Given observations $\ldots, \mathbf{y}(t_0 - 2), \mathbf{y}(t_0 - 1)$ and control signals $\ldots, \mathbf{u}(t_0 - 2), \mathbf{u}(t_0 - 1)$
1: Fix future $\mathbf{y}(t_0 + T_c) = \mathbf{r}(t_0 + T_c)$, $\mathbf{y}(t_0 + T_c + 1) = \mathbf{r}(t_0 + T_c + 1), \ldots$
2: Infer the distribution $q(\mathbf{u}(t), \mathbf{x}(t), \mathbf{y}(t))$ for all t (a few forward and backward passes)
3: Select the mean of $q(\mathbf{u}(t_0))$ as the control signal
4: Observe $\mathbf{y}(t_0)$ and release $\mathbf{y}(t_0 + T_c)$
5: Increase t_0 and loop from 1

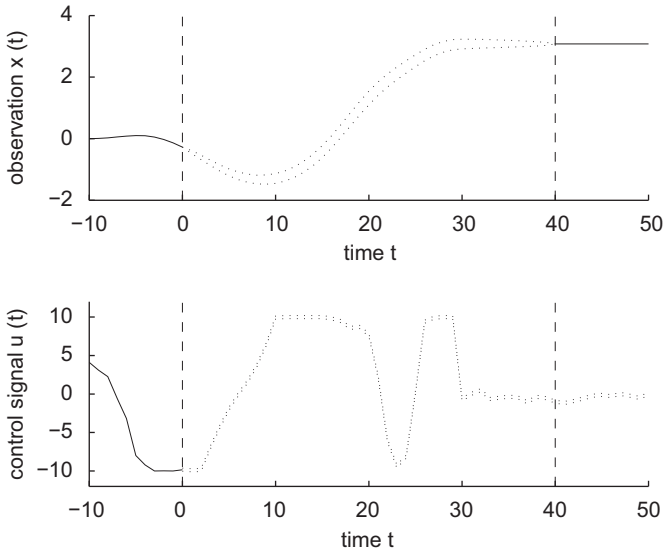

Fig. 2. Optimistic inference control (see Section 3.3). The inferred observations and control signals are plotted with confidence intervals. The current time is $t_0 = 0$ and after time $t_0 + T_c = 40$, the observation $\mathbf{x}(t)$ is assumed to be at the desired level $\mathbf{r}(t)$.

The differences between NMPC and OIC are as follows. In NMPC, the forward pass is based on the probabilistic inference, and the backward pass propagates the control cost J. OIC does not have a separate control cost, but the control objective is included as evidence (virtual future observations) to the probabilistic inference method.

In case there are constraints for control signals or observations in OIC, they are forced after every inference iteration. If the horizon is set too short or the goal is otherwise overoptimistic, the method becomes unreliable. Even with a realistic goal, it is not in general guaranteed that the iteration will converge to the optimal control signal, as the iteration may get stuck in an undesired local minimum of the cost function (5). The inferred control signals could be validated by releasing the optimistic future and re-inferring. If the future changes a lot, the control is unreliable.

The idea of using probabilistic inference for choosing actions originates from [9], where the likelihood function in probabilistic inference is replaced by the reward. (Perhaps it would be better to use exp(reward) which makes the approach invariant to adding a constant to all rewards.) With binary likelihood functions, this approach is equivalent to optimistic inference control. The first application to sequential decision problems [6] emphasised the fact that this methodology links inference and decision making such that algorithmic improvements in one can be used for the other.

Toussaint et al. [34,33] also use Bayesian inference for control such that motions with high control loss is treated as small posterior probability. The dynamics of the system are assumed to be known and belief propagation in factor graphs is used for the inference. Hoffman et al. [12] used OIC with MCMC for learning the policy.

3.4. Initialisation for control

The minimisation problems involved in our model are typically not convex. For nonlinear control tasks it is important that the initial estimate for the control signals $\{\mathbf{u}(t)\}_{t=t_0}^{t_0+T_c-1}$ is good, otherwise the optimisation algorithm may get stuck in a local minimum or fail to converge in reasonable time. In many cases, the control signal from the previous time step can be used with quite good results. However, when a new control task starts or the goal is changed, or there are unexpected changes in the system state, the previous control signal is often a poor choice.

A second option is to use random initialisations. If multiple different initialisations are used, this can be more robust than the first option. Unfortunately this is also much more time consuming because multiple control strategies must be computed for a single time step. If an internal forward model is available, a third option is also possible. The current system state can be propagated forward in time and the control signal from these predictions can be used as an initialisation.

4. Experiments

Mechanical dynamical systems are easily understandable by people and thus illustrative as examples. We chose a simulated system to ease experimentation. To make the setting more challenging, the controllers do not have access to the simulation equations but have to identify an unknown system instead. Our aim is not to find a solution to the cart–pole system, but to demonstrate our general algorithm for learning to control a system with unknown, complex, and nonlinear dynamics.

4.1. Cart–pole swing-up task

The cart–pole system [15] is a classic benchmark for nonlinear control. The system consist of a pole (which acts as an inverted pendulum) attached to a cart (Fig. 3). The force applied to the cart can be controlled, and the goal is to swing the pole to an upward position and stabilise it. This must be accomplished without the cart crashing into the walls of the track. It should be noted that this problem cannot be solved with a linear controller, and that the dynamics of the system are nonlinear, since for instance, the force with which the cart supports the pole is a nonlinear function of the system state.

The observed variables of the system are the position of the cart s, angle of the pole measured from the upward position ϕ, and their time derivatives s' and ϕ'. Control input is the force F applied to the cart. The detailed dynamics and constraints for the simulated cart–pole system can be found in [15]. $l = 0.5$ m is half the length of the pole, $g = 9.8\,\mathrm{m/s^2}$ is the acceleration of gravity, and $\mu_c = 0.05$ and $\mu_p = 0.01$ are the coefficients of the friction of the pole and the cart respectively.

A discrete system was simulated with a time step of $\Delta t = 0.05$ s. The possible force was constrained between -10 and 10 N, and the position between -3 and 3 m. The system was initialised to a random state taken from the uniform distributions $s = [-1, 1]$, $s' = [-2, 2]$, $\phi = [\pi - 1, \pi + 1]$, $\phi' = [-3, 3]$.

The state of the art [27] is that performing the swing-up can be learned efficiently even when assuming the system dynamics unknown, but under a very modest amount of observation noise.

4.2. Simulation

For all the simulations and the training data set, additive Gaussian observation noise with $\sigma = 0.001$ and Gaussian process noise with $\sigma = 0.001$ were used. For the performance comparison between NMPC and OIC, the length of the control horizon was set to 40 time steps corresponding to 2 s of system's time. The simulations were run for 70 time steps corresponding to 3.5 s of system's time to ensure that the controller was able to stabilise the pole.

NMPC was also tested with two variations. In the first variation, only the location of the cart s and the angle of the pole ϕ are observed. Even though this is a somewhat artificial modification to the problem, it highlights the benefits of hidden state-space model over directly learning the dynamics of the system based on observations. The second way of varying the problem setting is used to study the benefits of using a hidden state space in modelling the dynamics of an unknown system. A comparison model was built which used identity mapping $\mathbf{I}$ instead of an MLP $\mathbf{f}$ for the observation mapping. In practice this means replacing Eq. (2) with

$$\mathbf{y}(t) = \mathbf{x}(t) + \mathbf{w}(t). \tag{12}$$

This resembles a more traditional approach to NMPC and is thus a good comparison.

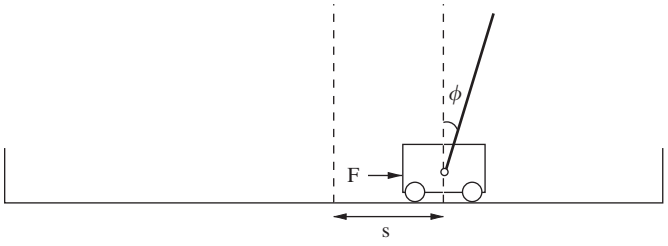

Fig. 3. The cart–pole system.

4.3. Implementation

The NDFA package [35] version 1.0.0, the scripts for running the experiments, and the used training data are publicly available.[1]

The model for NMPC and OIC control was identified in a separate training phase with 2500 samples. This can be compared to experiments in [37], where different reinforcement learning algorithms require from 9000 up to 2 500 000 samples to learn to control the cart. Most of the training data consisted of a sequence generated with semi-random control where the only goal was to ensure that the cart does not crash into the boundaries. Training data also contained some examples of hand-generated sections to better model the whole range of the observation and the dynamic mapping. The model was trained for 10 000 iterations, which translates to several hours of computation time. Six-dimensional ($m=6$) state space $\mathbf{x}(t)$ was used because it resulted in a model with the lowest cost function (Eq. (5)).

The direct control method requires a separate data set where the system is controlled successfully. This data consisted of 30 examples of successful swing-ups with 100 samples each. They were generated using the NMPC method with a horizon length of 40 time steps. Four-dimensional ($m=4$) state space proved to be the best here, and the model was trained for 100 000 iterations.

The state $\mathbf{x}(t)$ was estimated using the iterated extended Kalman smoother [1]. A history of five observations and control signals seemed to suffice to give a reliable estimate. The reference signal $\mathbf{r}(t)$ was $\phi=0$ and $\phi'=0$ at the end of the horizon and for five observations beyond that.

To take care of the constraints in the system with NMPC, a slightly modified version of the cost function (10) was used. Out-of-bounds values of the location of the cart and the force incurred a quadratic penalty, and the full cost function is

$$J_1(t_0,\mathbf{u}) = J(t_0,\mathbf{u}) \sum_{\tau=1}^{T_c} (\max(10, |u(t_0+\tau)|) - 10)^2 + \sum_{\tau=1}^{T_c} (\max(3, |y_s(t_0+\tau)|) - 3)^2, \quad (13)$$

where $y_s(t)$ refers to the location component s of the observation vector $\mathbf{y}(t)$.

Alternatively, a constrained optimisation algorithm could have been used. The performance of the non-constrained optimisation was found to be satisfactory, however, so this increase in complexity was deemed unnecessary. It is worth noting as well, that for any approximative model the actual plant is not guaranteed to satisfy the constraints even if predictions made by the optimisation algorithm indicate so. As there is typically no training data beyond the region within constraints, the predictions can become quite unreliable near the boundaries, so this is a real problem.

For OIC, the same reference signal was used as in NMPC. Means of the states were approximated with an extended Kalman smoother and these were then used to initialise the NDFA alongside with variances from the previous iteration. The performance was quite similar to the NMPC scheme. Position of the cart and the force were in no way constrained, however, which lead to some collisions with the track boundaries and unsuccessful swing-ups.

[1] http://www.cis.hut.fi/projects/bayes/software/

4.4. Simulation results

For all the control schemes, the cart–pole simulation was run for 100 times and the number of successful swing-ups was collected. As in [15], a swing-up is considered successful if the final angle is between -0.133π and 0.133π, final angular velocity between -2 and $2\,\mathrm{rad/s}$, and the cart has not crashed into the boundaries of the area during swing-up.

The results of all the simulations are collected in Table 2. For each simulation type, the number of successful swing-ups and the number of partial successes are listed. The partial successes include all the simulation runs that at some point reached the desired state, but possibly still failed either because the pole was not stabilised or the cart crashed into a wall.

The direct control could perform the swing-up part of the task quite well, but there were problems with stabilising the pole. Further testing is still needed to verify if the performance of the method can be improved by extra training with pole stabilising data.

Even though there was some modelling error left in the model used with indirect control schemes, both methods performed extremely well under low noise conditions. Even with added noise, the performance was pretty satisfactory. Examples of successful swing-ups can be found in Fig. 4. With high noise, some of the smoothness of the signal was lost. This is likely to be caused by the much larger error in the state estimates, which leads to varying control as well.

To measure the usefulness of the hidden state space, we compared the approach where the state space was the observation space, more accurately by restriction $\mathbf{f}=\mathbf{I}$. Working in the observation space directly, the performance was still perfect when the noise level was low. However, with a high noise level, the original model performed clearly better than the modified model. This result shows that the proposed method was able to find a state representation that made the system easier to control.

Even though most of the information on the speed y' and the angular velocity ϕ' can still be inferred taking into account past observations, in practice the problem of learning the dynamics of the system becomes harder and relying on past observations increases the reaction time. The model with hidden state could still perform the swing-up with some success, but a model based directly on observations could not handle the swing-up at all. This result was to be expected, as the dynamic mapping (1) alone cannot adequately describe the modified system.

On average, the traditional NMPC method was about 10–20 times slower than real-time on modern hardware (2.2 GHz AMD Opteron). The computation times for OIC were more varied, but in most cases the performance was inferior to NMPC. It should be noted, however, that the current implementation of OIC is quite heavily penalised by the presence of constraints, as the optimisa-

Table 2
Results: number of successful and semi-successful (in brackets) swing-ups with low and high noise level σ.

Setting	$\sigma=0.001$		$\sigma=0.1$	
Direct control	14	(48)	4	(31)
Optimistic inference control	97	(100)	94	(98)
NMPC	100	(100)	94	(95)
NMPC (only y and ϕ observed)	14	(66)	1	(21)
NMPC ($\mathbf{f}=\mathbf{I}$)	100	(100)	70	(70)
NMPC ($\mathbf{f}=\mathbf{I}$, only y and ϕ)	0	(0)	0	(0)

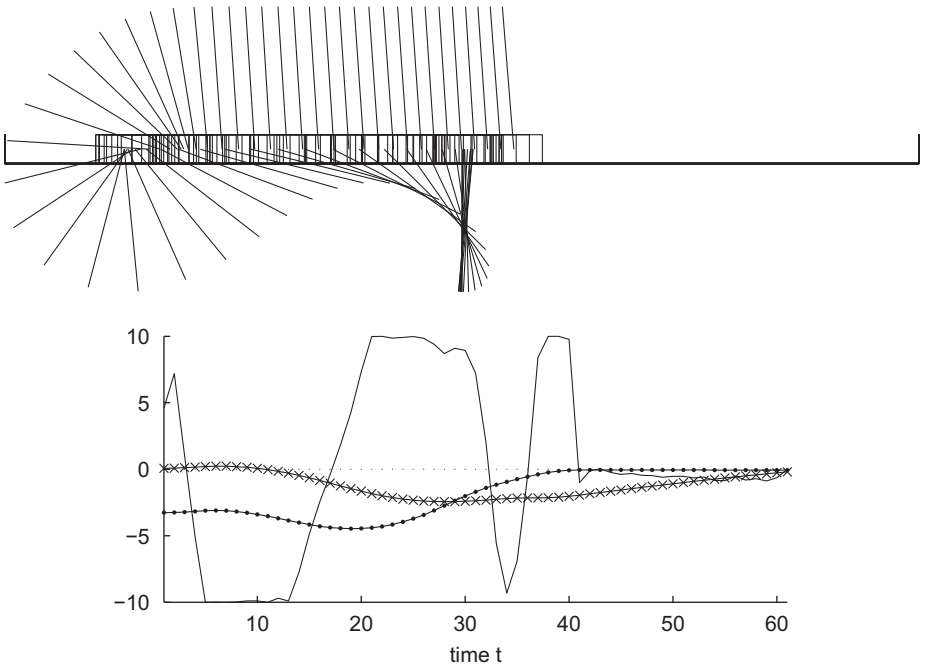

Fig. 4. Example of a successful swing-up with NMPC and low noise. The cart starts from the middle with the pole hanging down, and goes left to swing the pole up.

tion algorithm used cannot properly take their effects into account. In general, it is clear that further optimisations to the algorithms or improvements in hardware are required, before complex systems with fast dynamics can be controlled.

Comparison of the performance of NMPC and OIC can be seen in Fig. 5. With enough iterations, both methods reached a very high success rate. The few failed swing-ups were typically caused by difficult initial state of the cart–pole system resulting in an infeasible control strategy caused by limited horizon length. Example of a successful swing-up can be found in Fig. 4.

The importance of the horizon length to the performance of the NMPC can be seen in Fig. 6. All horizon lengths between 30 and 45 time steps had similar performance. Horizon lengths between 25 and 30 had problems with the cart crashing to the walls. Horizons shorter than 25 time steps could not reliably perform the swing-up task because the reference signal became too unrealistic.

Very long horizons are also problematic. First of all, they increase the computational burden of the algorithm. The increase in the number of the parameters often also leads into increase in the number of local minima, which makes the optimisation problem more involved. In addition, because only an approximative model of the system is available, longer predictions to the future are also more unreliable. This can lead the algorithm to choose an optimisation strategy which is not feasible in practice.

Different initialisations for the NMPC control signal show that local minima are the chief problem with long horizons (Fig. 6). It was observed that in most failed swing-ups the controller made a large prediction error, and in the following time instant was unable to recover from the local minimum where both the force penalty and the reference signal tracking penalty both suddenly became large. A more reasonable way to generate new initialisations in such situations is to either use random initialisations or to use the internal forward model to generate a new control signal.

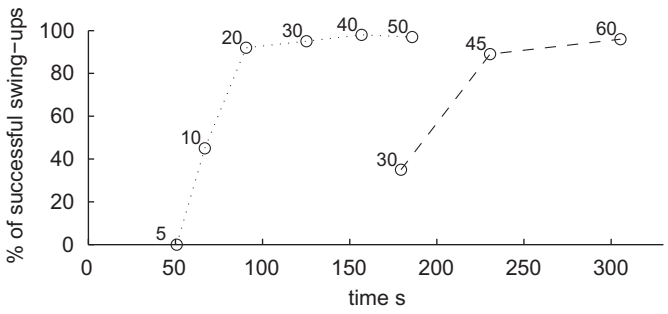

Fig. 5. Performance of the algorithms versus total computation time (in seconds). Dotted line is NMPC, dashed line is OIC. Numbers next to data points indicate number of iterations used. Control horizon length was 40 for all experiments.

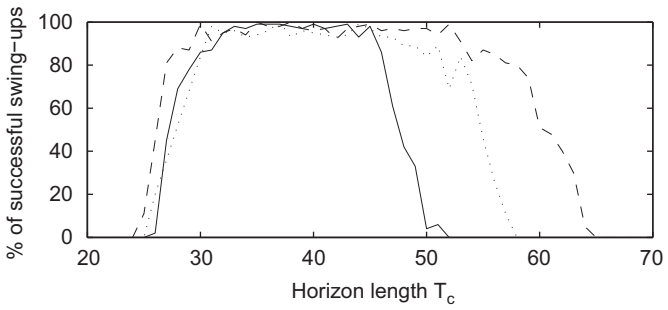

Fig. 6. The percentage of successful swing-ups as a function of the horizon length T_c in NMPC. Solid line is using old predictions as initialisation, dotted line is using initialisations based on the internal forward model and dashed line is using the best out of 10 random initialisations. Fifty NMPC iterations were used for all experiments.

5. Discussion and conclusion

While linear state-space models have long been used in system identification, the use of nonlinear hidden state-space models for control is a promising new direction [29]. In this paper, a state-of-the-art probabilistic method for identification of nonlinear hidden state-space models [35] is extended to controlled systems. The benefits of the model in [35] include the explicit modelling of both the uncertainty in the current state and in the modelled dynamics.

Three different control schemes were studied in the framework of variational Bayesian learning of nonlinear hidden state-space models. Fairly little data are required for learning a model useful for control. Direct control is fast to use, but requires the learning of a policy mapping, which is hard to do well. The second control scheme is stochastic nonlinear model predictive control, which is based on optimising control signals based on a cost function. The third scheme is optimistic inference control, which is based on fixing the desired observations at some point in the future and inferring the state and control signals between the future and the current state.

The nonlinear hidden state-space model and the variational Bayesian framework underlying it, are both rather complex. It will therefore require open-mindedness for people to actually take it into use. Also, it is very hard to guarantee that they work in all cases. Especially, it will not work well in situations that are completely different from the training data.

Variational Bayesian EM-algorithm cannot be easily used in this problem because neither the E-step (inference) nor the M-step (parameter update) can be solved analytically. Learning, inference, and NMPC use iterative algorithms based on gradient minimisation of the cost function instead. The indirect control schemes are computationally heavy which might restrict their use in time-critical applications.

Very recently, Rasmussen and Deisenroth [27] proposed a framework somewhat similar to ours. The common points are that they also were looking for a general algorithm for effective learning instead of a solution to a particular problem (they also used the cart–pole as an example), and that they also modelled the uncertainty of both the state and the model dynamics. Both approaches used a Gaussian approximation for the states. They only use direct control, but teach it not by example, but by policy optimisation. Instead of a hidden state, they used observations as the state directly, and instead of modelling dynamics and policy with MLPs, they used Gaussian processes. They only consider the case of modest amount of observation noise. It would be an interesting future direction to combine our best properties with theirs, that is, perhaps our hidden state representation and their policy optimisation.

5.1. Future work

A controller might be able to carry out active information gathering or probing [7]. It means that in an unknown state, one should first decrease the uncertainty and then take action based on what has been revealed. Probing requires the controller to be able to plan to react to future observations. Optimistic inference control does this automatically in theory, but in practice it would require an even more sophisticated model for the posterior distribution than Eq. (9). An interesting continuation is to use Monte Carlo methods, where one could also plan to give the control to the direct controller after a short horizon.

On a larger scale, to reduce the uncertainty of the model parameters, the controller should balance exploration and exploitation. In practice, this requires such long horizons that some heuristics need to be used. A good starting point for taking exploration into account is in [32].

For direct control, the model was learned using examples of control with a single goal in mind. It is straightforward to generalise this into a situation with a selection of different goals. The dynamics of the system stays the same regardless of the goal and only the policy mapping (see Fig. 1) needs to be goal-dependent.

Model-predictive control schemes are computationally very demanding. One possible way to speed up the NMPC algorithm would be to parallelise it. The MLP networks used in this work are not particularly well suited for parallel computation, but many parts of the computation can still be divided to parts. The novel control scheme, OIC, provides a link between Bayesian inference and model-predictive control, but does not currently compete in efficiency.

The direct and indirect control methods can be used together. Direct control can be used for initialising the control signals. The data produced by indirect control can be used for learning the direct controller. This can be done even offline, that is, simulating the estimated model and sampling observations from their predicted distributions. The enhancement of the task-oriented identification (policy mapping) in turn should help the indirect control, too. This idea is comparable to temporal difference learning [31] where the difference of temporally successive predictions is used for adjusting the earlier one. One should be careful, though. If the examples given for learning are fluent all the time, the robustness of the model might start to decrease.

Applications in which the proposed methodology would be useful could have unknown nonlinear dynamics and fairly high dimensional measurements, for instance in controlling a large process as a whole. Another example is robotics. In [24], a robot arm control is studied. The starting point is operational space control, where given the measurements, a reference trajectory, and the dynamics of the system, the control signal with minimum cost is solved. However, the mechanical parts have nonlinear effects that cannot be modelled in practice. Instead, they propose to learn a mapping from the state and the reference signal to the control signal from sampled data using a function approximator. This function approximator has the same role as the policy mapping in Fig. 1 in our approach.

5.2. Conclusions

Selecting actions based on a hidden state-space model instead of based on the observation directly has many benefits: firstly, it is more resistant to noise because it implicitly involves filtering. Secondly, the observations (without history) do not always carry enough information about the system state. Adding observation history explicitly increases the number of parameters in the model a lot. Thirdly, when nonlinear dynamics are modelled by a function approximator such as a multilayer perceptron network, a hidden state-space model can find such a representation of the state that it is more suitable for the approximation and thus more predictable. The benefit was empirically shown to be significant in the case of high noise.

When task-oriented identification is used, the state representation becomes such that also the control signals become easier to predict, that is, control becomes easier. The learned policy mapping can also be straightforwardly used for direct control. We think that task-oriented identification should also help indirect control methods but this is yet to be experimentally confirmed.

Nonlinear state-space models seem promising for complex control tasks, where the observations about the system state are incomplete or the dynamics of the system is not well known. The experiments with a simple control task indicated the benefits of the proposed approach. There is still work left in combating high computational complexity and in giving some guarantees or proofs on performance especially in unexpected situations or near boundaries.

Acknowledgements

We would like to thank Janne Paanajärvi, Heikki Hyötyniemi, Kai Zenger, Sampsa Laine, Harri Valpola, and Antti Honkela for fruitful discussions. This work was funded in part by the Academy of Finland project ‘Unsupervised machine learning in latent variable models’, and the IST Program of the European Community, under the PASCAL2 Network of Excellence. This publication only reflects the authors’ views.

References

[1] B. Anderson, J. Moore, Optimal Filtering, Prentice-Hall, Englewood Cliffs, NJ, 1979.
[2] K. Åström, P. Albertos, M. Blamke, A. Isidori, W. Schaufelberger, R. Sanz, Control of complex systems, Springer, Berlin, 2001.
[3] K. Åström, B. Wittenmark, Adaptive Control, second ed., Addison-Wesley, Reading, MA, 1995.
[4] C.G. Atkeson, Using local trajectory optimizers to speed up global optimization in dynamic programming, in: J.D. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann Publishers Inc., Los Altos, CA, 1994.
[5] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning for control, Artificial Intelligence Review 11 (1–5) (1997) 75–113.
[6] H. Attias, Planning by probabilistic inference, in: Proceedings of the AI-Stats 2003, 2003.
[7] Y. Bar-Shalom, Stochastic dynamic programming: caution and probing, IEEE Transactions on Automatic Control 26 (5) (1981) 1184–1195.
[8] A. Bemporad, G. Ferrari-Trecate, M. Morari, Observability and controllability of piecewise affine and hybrid systems, IEEE Transactions on Automatic Control 45 (10) (2000) 1864–1876.
[9] P. Dayan, G.E. Hinton, Using expectation-maximization for reinforcement learning, Neural Computation 9 (2) (1997) 271–278.
[10] J.A. Farrell, M.M. Polycarpou, Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches, Wiley, New York, 2006.
[11] S. Haykin, Neural Networks—A Comprehensive Foundation, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1999.
[12] M. Hoffman, A. Doucet, N. de Freitas, A. Jasra, Bayesian policy learning with trans-dimensional MCMC, in: Advances in Neural Information Processing Systems (NIPS), 2008.
[13] A. Honkela, H. Valpola, Unsupervised variational Bayesian learning of nonlinear models, in: L. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems 17, MIT Press, Cambridge, MA, USA, 2005, pp. 593–600.
[14] H. Kappen, A linear theory for control of non-linear stochastic systems, Physical Review Letters 95 (20) (2005).
[15] H. Kimura, S. Kobayashi, Efficient non-linear control by combining Q-learning with local linear controllers, in: Proceedings of the ICML, 1999.
[16] J. Kocijan, R. Murray-Smith, C. Rasmussen, A. Girard, Gaussian process model based predictive control, in: American Control Conference, Boston, MA, 2004.
[17] J. Kocijan, R. Murray-Smith, C.E. Rasmussen, B. Likar, Predictive control with Gaussian process models, in: Proceedings of IEEE Region 8 Eurocon 2003: Computer as a Tool, 2003.
[18] D. Mayne, J. Rawlings, C. Rao, P. Scokaert, Constrained model predictive control: stability and optimality, Automatica 36 (2000) 789–814.
[19] A. Moore, Acquisition of dynamic control knowledge for a robotic manipulator, in: M.B. Morgan (Ed.), Proceedings of the 7th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA 94104, 1990.
[20] R. Murray, K.J. Åström, S.P. Boyd, R.W. Brockett, G. Stein, Future directions in control in an information-rich world, IEEE Control Systems Magazine 23 (2) (2003) 20–33.
[21] J. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.
[22] M. Nørgaard, O. Ravn, N.K. Poulsen, L.K. Hansen, Neural Networks for Modelling and Control of Dynamic Systems, Springer-Verlag London Limited, 2001.
[23] J. Peters, S. Schaal, Policy learning for motor skills, in: Proceedings of 14th International Conference on Neural Information Processing (ICONIP 2007), Springer, Berlin, 2007.
[24] J. Peters, S. Schaal, Learning to control in operational space, The International Journal of Robotics Research 27 (2) (2008) 197–212.
[25] T. Raiko, M. Tornio, Learning nonlinear state-space models for control, in: Proceedings of the IJCNN'05, Montreal, Canada, 2005.
[26] T. Raiko, M. Tornio, A. Honkela, J. Karhunen, State inference in variational Bayesian nonlinear state-space models, in: Proceedings of the ICA 2006, Charleston, South Carolina, USA, 2006.
[27] C.E. Rasmussen, M.P. Deisenroth, Probabilistic inference for fast learning in control, in: Recent Advances in Reinforcement Learning, Lecture Notes in Computer Science, vol. 5323, Springer, Berlin, 2008, pp. 229–242.
[28] S. Richter, D. Aberdeen, J. Yu, Natural actor-critic for road traffic optimisation, in: Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2007.
[29] F. Rosenqvist, A. Karlström, Realisation and estimation of piecewise-linear output-error models, Automatica 41 (3) (2005) 545–551.
[30] S. Schaal, C.G. Atkeson, S. Vijayakumar, Scalable techniques from nonparameteric statistics for real-time robot learning, Applied Intelligence 1 (2002) 49–60.
[31] R. Sutton, Learning to predict by the methods of temporal differences, Machine Learning 3 (1988) 9–44.
[32] S.B. Thrun, The role of exploration in learning control, in: Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches, Van Nostrand Reinhold, New York, 1992, pp. 527–559.
[33] M. Toussaint, Bayesian inference for motion control and planning, Technical Report 22, Technische Universität Berlin, 2007.
[34] M. Toussaint, A. Storkey, Probabilistic inference for solving discrete and continuous state Markov decision processes, in: ICML '06: Proceedings of the 23rd International Conference on Machine Learning, ACM, New York, NY, USA, 2006.
[35] H. Valpola, J. Karhunen, An unsupervised ensemble learning method for nonlinear dynamic state-space models, Neural Computation 14 (11) (2002) 2647–2692.
[36] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, S. Schaal, statistical learning for humanoid robots, Autonomous Robots 1 (2002) 59–72.
[37] P. Wawrzynski, A. Pacut, Model-free off-policy reinforcement learning in continuous environment, in: Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, 2004.

Tapani Raiko received his DSc degree in Computer Science in 2006 from Helsinki University of Technology (TKK). He works as a post-doc researcher in the Adaptive Informatics Research Centre (AIRC) in TKK.

Matti Tornio worked as a research assistant in AIRC from 2004 to 2007

.