PILCO: A Model-Based and Data-Efficient Approach to Policy Search.

Conference Paper · January 2011

Source: DBLP

2 authors, including:

Marc Peter Deisenroth
University College London
120 PUBLICATIONS **11,608** CITATIONS

PILCO: A Model-Based and Data-Efficient Approach to Policy Search

Marc Peter Deisenroth MARC@CS.WASHINGTON.EDU
Department of Computer Science & Engineering, University of Washington, USA

Carl Edward Rasmussen CER54@CAM.AC.UK
Department of Engineering, University of Cambridge, UK

Abstract

In this paper, we introduce PILCO, a practical, data-efficient model-based policy search method. PILCO reduces model bias, one of the key problems of model-based reinforcement learning, in a principled way. By learning a probabilistic dynamics model and explicitly incorporating model uncertainty into long-term planning, PILCO can cope with very little data and facilitates learning from scratch in only a few trials. Policy evaluation is performed in closed form using state-of-the-art approximate inference. Furthermore, policy gradients are computed analytically for policy improvement. We report unprecedented learning efficiency on challenging and high-dimensional control tasks.

1. Introduction and Related Work

To date, reinforcement learning (RL) often suffers from being data inefficient, i.e., RL requires too many trials to learn a particular task. For example, learning one of the simplest RL tasks, the mountain-car, often requires tens if not hundreds or thousands of trials—independent of whether policy iteration, value iteration, or policy search methods are used. Hence, RL methods are often largely inapplicable to mechanical systems that quickly wear out, e.g., low-cost robots.

Increasing data efficiency requires either having informative prior knowledge or extracting more information from available data. In this paper, we do *not* assume that any expert knowledge is available (e.g., in terms of demonstrations or differential equations for the dynamics). Instead, we elicit a general policy-search framework for data-efficient learning from scratch.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

Generally, model-based methods, i.e., methods that learn a dynamics model of the environment, are more promising to efficiently extract valuable information from available data than model-free methods such as Q-learning or TD-learning. One reason why model-based methods are not widely used in learning from scratch is that they suffer from *model bias*, i.e., they inherently assume that the learned dynamics model sufficiently accurately resembles the real environment, see, e.g., (Schneider, 1997; Schaal, 1997; Atkeson & Santamaría, 1997). Model bias is especially an issue when only a few samples and no informative prior knowledge about the task to be learned are available.

Fig. 1 illustrates how model bias affects learning. Given a small data set of observed deterministic transitions (left), multiple transition functions plausibly could have generated the data (center). Choosing a single one causes severe consequences: When long-term predictions (or sampling trajectories from this model) leave the training data, the predictions of the function approximator are essentially arbitrary, but they are claimed with full confidence! By contrast, a probabilistic function approximator places a posterior distribution over the transition function (right) and expresses the level of uncertainty about the model.

Hence, for learning from scratch, we first require a *probabilistic* dynamics model to express model uncertainty. We employ non-parametric probabilistic Gaussian processes (GPs) for this purpose. Second, model uncertainty *must* be incorporated into planning and policy evaluation. Deterministic approximate inference techniques for policy evaluation allows us to apply policy search based on *analytic* policy gradients. An explicit value function model is not required. Based on these ideas, we propose a model-based policy search method, which we call PILCO (probabilistic inference for learning control). PILCO achieves unprecedented data efficiency in continuous state-action domains and is directly applicable to physical systems, e.g., robots.

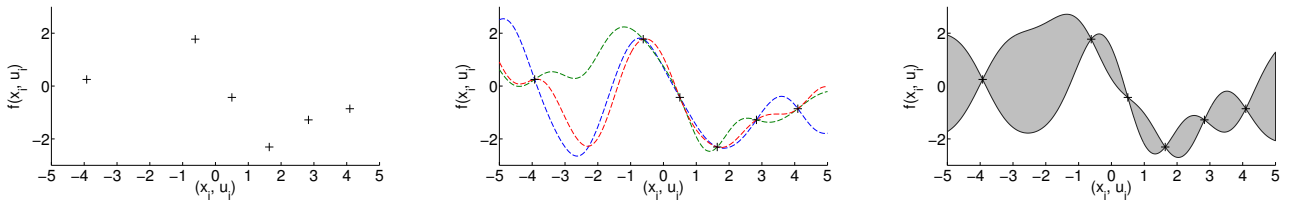


Figure 1. Small data set of observed transitions (left), multiple plausible deterministic function approximators (center), probabilistic function approximator (right). The probabilistic approximator models uncertainty about the latent function.

A common approach in designing adaptive controllers, which takes uncertainty of the model parameters into account, is to add an extra term in the cost function of a minimum-variance controller (Fabri & Kadirkamanathan, 1998). Here, the uncertainty of the model parameters is penalized to improve the model-parameter estimation. Abbeel et al. (2006) proposed further other successful heuristics to deal with inaccurate models. Based on good-guess parametric dynamics models, locally optimal controllers, and temporal bias terms to account for model discrepancies, very impressive results were obtained. Schneider (1997) and Bagnell & Schneider (2001) proposed to account for model bias by explicitly modeling and averaging over model uncertainty. PILCO builds upon the successful approach by Schneider (1997), where model uncertainty is treated as temporally uncorrelated noise. However, PILCO neither requires sampling methods for planning, nor is it restricted to a finite number of plausible models.

Algorithms with GP dynamics models in RL were presented by Rasmussen & Kuss (2004), Ko et al. (2007), and Deisenroth et al. (2009). Shortcomings of these approaches are that the dynamics models are either learned by motor babbling, which is data inefficient, or value function models have to be maintained, which does not scale well to high dimensions. The approaches by Engel et al. (2003) and Wilson et al. (2010) are based GP value function models and, thus, suffer from the same problems. As an indirect policy search method, PILCO does not require an explicit value function model.

An extension of PILCO to deal with planning and control under consideration of task-space constraints in a robotic manipulation task is presented in (Deisenroth et al., 2011).

Throughout this paper, we consider dynamic systems

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{1}$$

with continuous-valued states $\mathbf{x} \in \mathbb{R}^D$ and controls $\mathbf{u} \in \mathbb{R}^F$ and unknown transition dynamics f. The objective is to find a deterministic *policy/controller* $\pi : \mathbf{x} \mapsto \pi(\mathbf{x}) = \mathbf{u}$ that minimizes the *expected return*

$$J^\pi(\theta) = \sum_{t=0}^{T} \mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)]\,, \quad \mathbf{x}_0 \sim \mathcal{N}(\mu_0, \mathbf{\Sigma}_0)\,, \tag{2}$$

of following π for T steps, where $c(\mathbf{x}_t)$ is the cost (negative reward) of being in state $\mathbf{x}$ at time t. We assume that π is a function parametrized by θ and that c encodes some information about a target state $\mathbf{x}_{\text{target}}$.

2. Model-based Indirect Policy Search

In the following, we detail the key components of the PILCO policy-search framework: the dynamics model, analytic approximate policy evaluation, and gradient-based policy improvement.

2.1. Dynamics Model Learning

PILCO's probabilistic dynamics model is implemented as a GP, where we use tuples $(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \in \mathbb{R}^{D+F}$ as training inputs and differences $\Delta_t = \mathbf{x}_t - \mathbf{x}_{t-1} + \varepsilon \in \mathbb{R}^D$, $\varepsilon \sim \mathcal{N}(0, \mathbf{\Sigma}_\varepsilon)$, $\mathbf{\Sigma}_\varepsilon = \text{diag}([\sigma_{\varepsilon_1}, \ldots, \sigma_{\varepsilon_D}])$, as training targets. The GP yields *one-step* predictions

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \mathcal{N}\big(\mathbf{x}_t \,|\, \mu_t, \mathbf{\Sigma}_t\big)\,, \tag{3}$$

$$\mu_t = \mathbf{x}_{t-1} + \mathbb{E}_f[\Delta_t]\,, \tag{4}$$

$$\mathbf{\Sigma}_t = \text{var}_f[\Delta_t]\,. \tag{5}$$

Throughout this paper, we consider a prior mean function $m \equiv 0$ and the squared exponential (SE) kernel k with automatic relevance determination. The SE covariance function is defined as

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \alpha^2 \exp\big(-\tfrac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')^\top \mathbf{\Lambda}^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')\big) \tag{6}$$

with $\tilde{\mathbf{x}} \coloneqq [\mathbf{x}^\top \mathbf{u}^\top]^\top$. Here, we define α^2 as the variance of the latent function f and $\mathbf{\Lambda} \coloneqq \text{diag}([\ell_1^2, \ldots, \ell_D^2])$, which depends on the characteristic length-scales ℓ_i. Given n training inputs $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n]$ and corresponding training targets $\mathbf{y} = [\Delta_1, \ldots, \Delta_n]^\top$, the posterior GP hyper-parameters (length-scales ℓ_i, signal variance α^2, noise variances $\mathbf{\Sigma}_\varepsilon$) are learned by evidence maximization (Rasmussen & Williams, 2006).

The posterior predictive distribution $p(\Delta_*|\tilde{\mathbf{x}}_*)$ for an arbitrary, but known, test input $\tilde{\mathbf{x}}_*$ is Gaussian with mean and variance

$$m_f(\tilde{\mathbf{x}}_*) = \mathbb{E}_f[\Delta_*] = \mathbf{k}_*^\top(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y} = \mathbf{k}_*^\top\beta\,, \quad (7)$$

$$\sigma_f^2(\Delta_*) = \mathrm{var}_f[\Delta_*] = k_{**} - \mathbf{k}_*^\top(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{k}_*\,, \quad (8)$$

respectively, where $\mathbf{k}_* \coloneqq k(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_*)$, $k_{**} \coloneqq k(\tilde{\mathbf{x}}_*, \tilde{\mathbf{x}}_*)$, $\beta \coloneqq (\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y}$, and $\mathbf{K}$ being the Gram matrix with entries $K_{ij} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$.

For multivariate targets, we train conditionally independent GPs for each target dimension, i.e., the GPs are independent for deterministically given test inputs. For *uncertain* inputs, the target dimensions covary.

2.2. Policy Evaluation

Minimizing and evaluating J^π in Eq. (2) requires long-term predictions of the state evolution. To obtain the state distributions $p(\mathbf{x}_1), \ldots, p(\mathbf{x}_T)$, we cascade one-step predictions, see Eqs. (3)–(5). Doing this properly requires mapping uncertain test inputs through the GP dynamics model. In the following, we assume that these test inputs are Gaussian distributed and extend the results from Quiñonero-Candela et al. (2003) to the multivariate case and the incorporation of controls.

For predicting $\mathbf{x}_t$ from $p(\mathbf{x}_{t-1})$, we require a joint distribution $p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. As the control $\mathbf{u}_{t-1} = \pi(\mathbf{x}_{t-1}, \theta)$ is a function of the state, we compute the desired joint as follows: First, we compute the mean μ_u and the covariance $\mathbf{\Sigma}_u$ of the predictive control distribution $p(\mathbf{u}_{t-1})$ by integrating out the state. Subsequently, the cross-covariance $\mathrm{cov}[\mathbf{x}_{t-1}, \mathbf{u}_{t-1}]$ is computed. Finally, we approximate the joint state-control distribution $p(\tilde{\mathbf{x}}_{t-1}) = p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ by a Gaussian with the correct mean and covariance. These computations depend on the parametrization of the policy π. For many interesting controller parametrizations, the required computations can be performed analytically, although often neither $p(\mathbf{u}_{t-1})$ nor $p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ are exactly Gaussian (Deisenroth, 2010).

From now on, we assume a joint Gaussian distribution $p(\tilde{\mathbf{x}}_{t-1}) = \mathcal{N}\big(\tilde{\mathbf{x}}_{t-1} \,|\, \tilde{\mu}_{t-1}, \tilde{\mathbf{\Sigma}}_{t-1}\big)$ at time $t-1$. When predicting the distribution

$$p(\Delta_t) = \int p(f(\tilde{\mathbf{x}}_{t-1})|\tilde{\mathbf{x}}_{t-1}) p(\tilde{\mathbf{x}}_{t-1})\,\mathrm{d}\tilde{\mathbf{x}}_{t-1}\,, \quad (9)$$

we integrate out the random variable $\tilde{\mathbf{x}}_{t-1}$. Note that the transition probability $p(f(\tilde{\mathbf{x}}_{t-1})|\tilde{\mathbf{x}}_{t-1})$ is obtained from the posterior GP distribution. Computing the exact predictive distribution in Eq. (9) is analytically intractable. Therefore, we approximate $p(\Delta_t)$ by a Gaussian using exact moment matching, see Fig. 2.

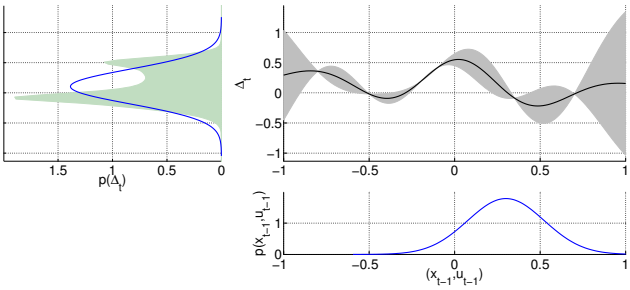


Figure 2. GP prediction at an uncertain input. The input distribution $p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ is assumed Gaussian (lower right panel). When propagating it through the GP model (upper right panel), we obtain the shaded distribution $p(\Delta_t)$, upper left panel. We approximate $p(\Delta_t)$ by a Gaussian with the exact mean and variance (upper left panel).

For the time being, assume the mean μ_Δ and the covariance $\mathbf{\Sigma}_\Delta$ of the predictive distribution $p(\Delta_t)$ are known. Then, a Gaussian approximation to the desired distribution $p(\mathbf{x}_t)$ is given as $\mathcal{N}\big(\mathbf{x}_t \,|\, \mu_t, \mathbf{\Sigma}_t\big)$ with

$$\mu_t = \mu_{t-1} + \mu_\Delta \quad (10)$$

$$\mathbf{\Sigma}_t = \mathbf{\Sigma}_{t-1} + \mathbf{\Sigma}_\Delta + \mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t] + \mathrm{cov}[\Delta_t, \mathbf{x}_{t-1}] \quad (11)$$

$$\mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t] = \mathrm{cov}[\mathbf{x}_{t-1}, \mathbf{u}_{t-1}]\mathbf{\Sigma}_u^{-1}\mathrm{cov}[\mathbf{u}_{t-1}, \Delta_t] \quad (12)$$

where the computation of the cross-covariances in Eq. (12) depends on the policy parametrization, but can often be computed analytically. The computation of the cross-covariance $\mathrm{cov}[\mathbf{x}_{t-1}, \Delta_t]$ in Eq. (11) is detailed by Deisenroth (2010).

In the following, we compute the mean μ_Δ and the variance $\mathbf{\Sigma}_\Delta$ of the predictive distribution, see Eq. (9).

2.2.1. Mean Prediction

Following the law of iterated expectations, for target dimensions $a = 1, \ldots, D$, we obtain

$$\mu_\Delta^a = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}[\mathbb{E}_f[f(\tilde{\mathbf{x}}_{t-1})|\tilde{\mathbf{x}}_{t-1}]] = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}[m_f(\tilde{\mathbf{x}}_{t-1})]$$

$$= \int m_f(\tilde{\mathbf{x}}_{t-1})\mathcal{N}\big(\tilde{\mathbf{x}}_{t-1} \,|\, \tilde{\mu}_{t-1}, \tilde{\mathbf{\Sigma}}_{t-1}\big)\,\mathrm{d}\tilde{\mathbf{x}}_{t-1} \quad (13)$$

$$= \beta_a^\top \mathbf{q}_a \quad (14)$$

with $\beta_a = (\mathbf{K}_a + \sigma_{\varepsilon_a}^2)^{-1}\mathbf{y}_a$ and $\mathbf{q}_a = [q_{a_1}, \ldots, q_{a_n}]^\top$. With m_f given in Eq. (7), the entries of $\mathbf{q}_a \in \mathbb{R}^n$ are

$$q_{a_i} = \int k_a(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_{t-1})\mathcal{N}\big(\tilde{\mathbf{x}}_{t-1} \,|\, \tilde{\mu}_{t-1}, \tilde{\mathbf{\Sigma}}_{t-1}\big)\,\mathrm{d}\tilde{\mathbf{x}}_{t-1}$$

$$= \frac{\alpha_a^2}{\sqrt{|\tilde{\mathbf{\Sigma}}_{t-1}\mathbf{\Lambda}_a^{-1}+\mathbf{I}|}} \exp\big(-\tfrac{1}{2}\nu_i^\top(\tilde{\mathbf{\Sigma}}_{t-1}+\mathbf{\Lambda}_a)^{-1}\nu_i\big)\,, \quad (15)$$

$$\nu_i \coloneqq (\tilde{\mathbf{x}}_i - \tilde{\mu}_{t-1})\,. \quad (16)$$

Here, ν_i in Eq. (16) is the difference between the training input $\tilde{\mathbf{x}}_i$ and the mean of the "test" input distribution $p(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$.

2.2.2. Covariance Matrix of the Prediction

To compute the predictive covariance matrix $\boldsymbol{\Sigma}_\Delta \in \mathbb{R}^{D\times D}$ we distinguish between diagonal elements and off-diagonal elements: Using the law of iterated variances, we obtain for target dimensions $a, b = 1, \ldots, D$

$$\sigma_{aa}^2 = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\big[\mathrm{var}_f[\Delta_a|\tilde{\mathbf{x}}_{t-1}]\big] + \mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a^2] - (\mu_\Delta^a)^2 \quad (17)$$

$$\sigma_{ab}^2 = \mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a\Delta_b] - \mu_\Delta^a\mu_\Delta^b\,, \quad a \neq b\,, \quad (18)$$

respectively, where μ_Δ^a is known from Eq. (14). The off-diagonal terms do not contain the additional term $\mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}[\mathrm{cov}_f[\Delta_a, \Delta_b|\tilde{\mathbf{x}}_{t-1}]]$ because of the conditional independence assumption of the GP models: Different target dimensions do not covary for given $\tilde{\mathbf{x}}_{t-1}$.

First we compute the terms that are common to both the diagonal and off-diagonal entries: With the Gaussian approximation $\mathcal{N}\big(\tilde{\mathbf{x}}_{t-1}\,|\,\tilde{\mu}_{t-1}, \tilde{\boldsymbol{\Sigma}}_{t-1}\big)$ of $p(\tilde{\mathbf{x}}_{t-1})$ and the law of iterated expectations, we obtain

$$\begin{aligned}\mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a\Delta_b] &= \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\big[\mathbb{E}_f[\Delta_a|\tilde{\mathbf{x}}_{t-1}]\,\mathbb{E}_f[\Delta_b|\tilde{\mathbf{x}}_{t-1}]\big]\\ &\overset{(7)}{=} \int m_f^a(\tilde{\mathbf{x}}_{t-1}) m_f^b(\tilde{\mathbf{x}}_{t-1}) p(\tilde{\mathbf{x}}_{t-1})\,\mathrm{d}\tilde{\mathbf{x}}_{t-1} \quad (19)\end{aligned}$$

due to the conditional independence of Δ_a and Δ_b given $\tilde{\mathbf{x}}_{t-1}$. Using now the definition of the mean function m_f in Eq. (7), we obtain

$$\mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a\Delta_b] = \beta_a^\top \mathbf{Q}\beta_b\,, \quad (20)$$

$$\mathbf{Q} \coloneqq \int k_a(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_{t-1})\, k_b(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_{t-1})^\top p(\tilde{\mathbf{x}}_{t-1})\,\mathrm{d}\tilde{\mathbf{x}}_{t-1}\,. \quad (21)$$

Using standard results from Gaussian multiplications and integration, we obtain the entries Q_{ij} of $\mathbf{Q} \in \mathbb{R}^{n\times n}$

$$Q_{ij} = \frac{k_a(\tilde{\mathbf{x}}_i, \tilde{\mu}_{t-1}) k_b(\tilde{\mathbf{x}}_j, \tilde{\mu}_{t-1})}{\sqrt{|\mathbf{R}|}} \exp\big(\tfrac{1}{2}\mathbf{z}_{ij}^\top \mathbf{R}^{-1}\tilde{\boldsymbol{\Sigma}}_{t-1}\mathbf{z}_{ij}\big) \quad (22)$$

where we defined $\mathbf{R} \coloneqq \tilde{\boldsymbol{\Sigma}}_{t-1}(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}) + \mathbf{I}$ and $\mathbf{z}_{ij} \coloneqq \boldsymbol{\Lambda}_a^{-1}\nu_i + \boldsymbol{\Lambda}_b^{-1}\nu_j$ with ν_i taken from Eq. (16). Hence, the *off-diagonal* entries of $\boldsymbol{\Sigma}_\Delta$ are fully determined by Eqs. (14)–(16), (18), and (20)–(22).

From Eq. (17), we see that the *diagonal* entries of $\boldsymbol{\Sigma}_\Delta$ contain an additional term

$$\mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\big[\mathrm{var}_f[\Delta_a|\tilde{\mathbf{x}}_{t-1}]\big] = \alpha_a^2 - \mathrm{tr}\big((\mathbf{K}_a + \sigma_{\varepsilon_a}^2\mathbf{I})^{-1}\mathbf{Q}\big) \quad (23)$$

with $\mathbf{Q}$ given in Eq. (22). This term is the expected variance of the latent function (see Eq. (8)) under the distribution of $\tilde{\mathbf{x}}_{t-1}$.

With the Gaussian approximation $\mathcal{N}\big(\Delta_t\,|\,\mu_\Delta, \boldsymbol{\Sigma}_\Delta\big)$ of $p(\Delta_t)$, we obtain a Gaussian approximation $\mathcal{N}\big(\mathbf{x}_t\,|\,\mu_t, \boldsymbol{\Sigma}_t\big)$ of $p(\mathbf{x}_t)$ through Eqs. (10)–(12).

To evaluate the expected return J^π in Eq. (2), it remains to compute the expected values

$$\mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)] = \int c(\mathbf{x}_t)\mathcal{N}\big(\mathbf{x}_t\,|\,\mu_t, \boldsymbol{\Sigma}_t\big)\,\mathrm{d}\mathbf{x}_t\,, \quad (24)$$

$t = 0, \ldots, T$, of the cost c with respect to the predictive state distributions. We assume that the cost c is chosen so that Eq. (24) can be solved analytically, e.g., polynomials. In this paper, we use

$$c(\mathbf{x}) = 1 - \exp(-\|\mathbf{x} - \mathbf{x}_{\mathrm{target}}\|^2/\sigma_c^2) \in [0, 1]\,, \quad (25)$$

which is a squared exponential subtracted from unity. In Eq. (25), $\mathbf{x}_{\mathrm{target}}$ is the target state and σ_c^2 controls the width of c. This unimodal cost can be considered a smooth approximation of a 0-1 cost of a target area.

2.3. Analytic Gradients for Policy Improvement

Both μ_t and $\boldsymbol{\Sigma}_t$ are functionally dependent on the mean μ_u and the covariance $\boldsymbol{\Sigma}_u$ of the control signal (and θ) through $\tilde{\mu}_{t-1}$ and $\tilde{\boldsymbol{\Sigma}}_{t-1}$, respectively, see Eqs. (15), (16), and (22), for instance. Hence, we can analytically compute the gradients of the expected return J^π with respect to the policy parameters θ, which we sketch in the following. We obtain the derivative $\mathrm{d}J^\pi/\,\mathrm{d}\theta$ by repeated application of the chain-rule: First, we swap the order of differentiating and summing in Eq. (2), and with $\mathcal{E}_t \coloneqq \mathbb{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)]$, we obtain

$$\frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}\theta} = \frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}p(\mathbf{x}_t)}\frac{\mathrm{d}p(\mathbf{x}_t)}{\mathrm{d}\theta} \coloneqq \frac{\partial\mathcal{E}_t}{\partial\mu_t}\frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} + \frac{\partial\mathcal{E}_t}{\partial\boldsymbol{\Sigma}_t}\frac{\mathrm{d}\boldsymbol{\Sigma}_t}{\mathrm{d}\theta}\,, \quad (26)$$

where we used the shorthand notation $\mathrm{d}\mathcal{E}_t/\,\mathrm{d}p(\mathbf{x}_t) \coloneqq \{\partial\mathcal{E}_t/\partial\mu_t, \partial\mathcal{E}_t/\partial\boldsymbol{\Sigma}_t\}$ for taking the derivative of $\mathcal{E}_t$ with respect to both the mean and covariance of $\mathbf{x}_t$. Second, from Sec. 2.2, we know that the predicted mean μ_t and the covariance $\boldsymbol{\Sigma}_t$ are functionally dependent on the moments of $p(\mathbf{x}_{t-1})$ and the controller parameters θ through $\mathbf{u}_{t-1}$. By applying the chain-rule to Eq. (26), we thus obtain

$$\frac{\mathrm{d}p(\mathbf{x}_t)}{\mathrm{d}\theta} = \frac{\partial p(\mathbf{x}_t)}{\partial p(\mathbf{x}_{t-1})}\frac{\mathrm{d}p(\mathbf{x}_{t-1})}{\mathrm{d}\theta} + \frac{\partial p(\mathbf{x}_t)}{\partial\theta}\,, \quad (27)$$

$$\frac{\partial p(\mathbf{x}_t)}{\partial p(\mathbf{x}_{t-1})} = \left\{\frac{\partial\mu_t}{\partial p(\mathbf{x}_{t-1})}, \frac{\partial\boldsymbol{\Sigma}_t}{\partial p(\mathbf{x}_{t-1})}\right\}. \quad (28)$$

From here onward, we focus on $\mathrm{d}\mu_t/\,\mathrm{d}\theta$, see Eq. (26), but computing $\mathrm{d}\boldsymbol{\Sigma}_t/\,\mathrm{d}\theta$ in Eq. (26) is similar. We get

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} = \frac{\partial\mu_t}{\partial\mu_{t-1}}\frac{\mathrm{d}\mu_{t-1}}{\mathrm{d}\theta} + \frac{\partial\mu_t}{\partial\boldsymbol{\Sigma}_{t-1}}\frac{\mathrm{d}\boldsymbol{\Sigma}_{t-1}}{\mathrm{d}\theta} + \frac{\partial\mu_t}{\partial\theta}\,. \quad (29)$$

Since $\mathrm{d}p(\mathbf{x}_{t-1})/\,\mathrm{d}\theta$ in Eq. (27) is known from time step $t-1$ and $\partial\mu_t/\partial p(\mathbf{x}_{t-1})$ is computed by applying the chain-rule to Eqs. (14)–(16), we conclude with

$$\frac{\partial\mu_t}{\partial\theta} = \frac{\partial\mu_\Delta}{\partial p(\mathbf{u}_{t-1})}\frac{\partial p(\mathbf{u}_{t-1})}{\partial\theta} = \frac{\partial\mu_\Delta}{\partial\mu_u}\frac{\partial\mu_u}{\partial\theta} + \frac{\partial\mu_\Delta}{\partial\boldsymbol{\Sigma}_u}\frac{\partial\boldsymbol{\Sigma}_u}{\partial\theta}\,. \quad (30)$$

The partial derivatives of $\partial\mu_u/\partial\theta$ and $\partial\boldsymbol{\Sigma}_u/\partial\theta$, see Eq. (30), depend on the policy parametrization θ. The

Algorithm 1 PILCO

init: Sample controller parameters $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Apply random control signals and record data.
repeat
  Learn probabilistic (GP) dynamics model, see Sec. 2.1, using all data.
  Model-based policy search, see Sec. 2.2–2.3.
  repeat
    Approximate inference for policy evaluation, see Sec. 2.2: get $J^\pi(\theta)$, Eqs. (10)–(12), (24).
    Gradient-based policy improvement, see Sec. 2.3: get $\mathrm{d}J^\pi(\theta)/\mathrm{d}\theta$, Eqs. (26)–(30).
    Update parameters θ (e.g., CG or L-BFGS).
  until convergence; **return** θ^*
  Set $\pi^* \leftarrow \pi(\theta^*)$.
  Apply π^* to system (single trial/episode) and record data.
until task learned

individual partial derivatives in Eqs. (26)–(30) can be computed *analytically* by repeated application of the chain-rule to Eqs. (10)–(12), (14)–(16), (20)–(23), and (26)–(30). We omit further lengthy details and refer to (Deisenroth, 2010) for more information.

Analytic derivatives allow for standard gradient-based non-convex optimization methods, e.g., CG or L-BFGS, which return optimized policy parameters θ^*. Analytic gradient computation of J^π is much more efficient than estimating policy gradients through sampling: For the latter, the variance in the gradient estimate grows quickly with the number of parameters (Peters & Schaal, 2006).

3. Experimental Results

In this section, we report PILCO's success in efficiently learning challenging control tasks, including both standard benchmark problems and high-dimensional control problems. In all cases, PILCO learns completely from scratch by following the steps detailed in Alg. 1. The results discussed in the following are *typical*, i.e., they do neither represent best nor worst cases. Videos and further information will be made available at `http://mlg.eng.cam.ac.uk/carl/pilco` and at `http://cs.uw.edu/homes/marc/pilco`.

3.1. Cart-Pole Swing-up

PILCO was applied to learning to control a *real* cart-pole system, see Fig. 3. The system consists of a cart with mass 0.7 kg running on a track and a freely swinging pendulum with mass 0.325 kg attached to the cart. The state of the system is the position of the cart, the velocity of the cart, the angle of the pendulum, and the angular velocity. A horizontal force $u \in [-10, 10]$ N could be applied to the cart. The objective was to learn a controller to swing the pendulum up and to balance it in the inverted position in the middle of the track. A linear controller is not capable of doing this (Raiko & Tornio, 2009). The learned state-feedback controller was a nonlinear RBF network, i.e.,

$$\pi(\mathbf{x}, \theta) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}) \,, \tag{31}$$

$$\phi_i(\mathbf{x}) = \exp(-\tfrac{1}{2}(\mathbf{x} - \mu_i)^\top \mathbf{\Lambda}^{-1}(\mathbf{x} - \mu_i)) \tag{32}$$

with $n = 50$ squared exponential basis functions centered at μ_i. In our experiment, $\theta = \{w_i, \mathbf{\Lambda}, \mu_i\} \in \mathbb{R}^{305}$.

PILCO successfully learned a sufficiently good dynamics model and a good controller for this standard benchmark problem fully automatically in only a handful of trials and a total experience of 17.5 s. Snapshots of a 20 s test trajectory are shown in Fig. 3.

3.2. Cart-Double-Pendulum Swing-up

In the following, we show the results for PILCO learning a dynamics model and a controller for the cart-double-pendulum swing-up. The cart-double pendulum system consists of a cart (mass 0.5 kg) running on a track and a freely swinging two-link pendulum (each link of mass 0.5 kg) attached to it. The state of the system is the position x_1 and the velocity $\dot{x}_1$ of the cart and the angles θ_2, θ_3 and the angular velocities of both attached pendulums. The control signals $|u| \leq 20$ N were horizontal forces to the cart. Initially, the system was expected to be in a state $\mathbf{x}_0$ at location x, where both pendulums hung down. The objective was to learn a policy π^* to swing the double pendulum up to the inverted position and to balance it with the cart being at the expected start location x. A linear controller is not capable of solving the this problem.

A standard control approach to solving the cart-double pendulum task is to design two separate controllers, one for the swing up and one linear controller for the balancing task, see for instance (Zhong & Röck, 2001), requiring prior knowledge about the task's solution. Unlike this engineered solution, PILCO fully automatically *learned* a dynamics model and a single nonlinear RBF controller, see Eq. (31), with $n = 200$ and $\theta \in \mathbb{R}^{1816}$ to jointly solve the swing-up and balancing. For this, PILCO required about 20–30 trials corresponding to an interaction time of about 60 s–90 s.

3.3. Unicycle Riding

We applied PILCO to riding a 5-DoF unicycle in a realistic simulation of the one shown in Fig. 4(a). The

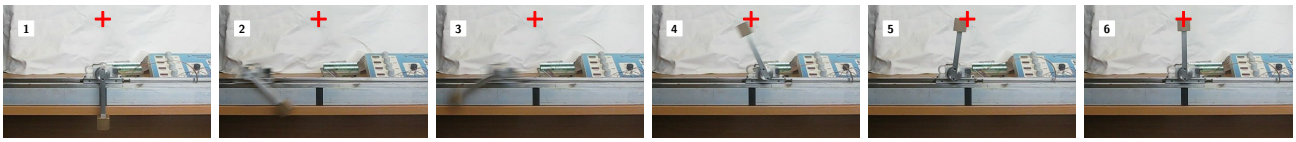
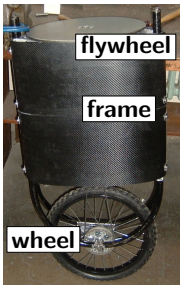
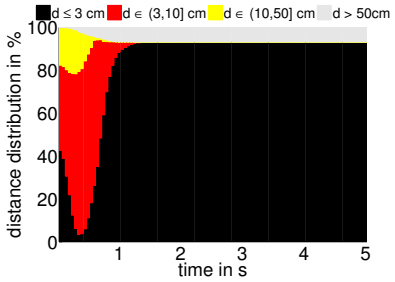


Figure 3. Real cart-pole system. Snapshots of a controlled trajectory of 20 s length after having learned the task. To solve the swing-up plus balancing, PILCO required only 17.5 s of interaction with the physical system.



(a) Robotic unicycle.



(b) Histogram (after 1,000 test runs) of the distances of the flywheel from being upright.

Figure 4. Robotic unicycle system and simulation results. The state space is $\mathbb{R}^{12}$, the control space $\mathbb{R}^2$.

unicycle is 0.76 m high and consists of a 1 kg wheel, a 23.5 kg frame, and a 10 kg flywheel mounted perpendicularly to the frame. Two torques could be applied to the unicycle: The first torque $|u_w| \leq 10$ Nm was applied directly on the wheel and mimics a human rider using pedals. The torque produced longitudinal and tilt accelerations. Lateral stability of the wheel could be maintained by steering the wheel toward the falling direction of the unicycle and by applying a torque $|u_t| \leq 50$ Nm to the flywheel. The dynamics of the robotic unicycle can be described by 12 coupled first-order ODEs, see (Forster, 2009).

The goal was to ride the unicycle, i.e., to prevent it from falling. To solve the balancing task, we used a linear controller $\pi(\mathbf{x}, \theta) = \mathbf{A}\mathbf{x} + \mathbf{b}$ with $\theta = \{\mathbf{A}, \mathbf{b}\} \in \mathbb{R}^{28}$. The covariance $\mathbf{\Sigma}_0$ of the initial state was $0.25^2\mathbf{I}$ allowing each angle to be off by about 30° (twice the standard deviation).

PILCO differs from conventional controllers in that it learns a single controller for all control dimensions *jointly*. Thus, PILCO takes the correlation of all control and state dimensions into account during planning and control. Learning separate controllers for each control variable is often unsuccessful (Naveh et al., 1999).

PILCO required about 20 trials (experience of about 30 s) to learn a dynamics model and a controller that keeps the unicycle upright. The interaction time is fairly short since a trial was aborted when the turntable hit the ground, which happened quickly during the five random trials used for initialization. Fig. 4(b) shows empirical results after 1,000 test runs with the learned policy: Differently-colored bars show the distance of the flywheel from a fully upright position. Depending on the initial configuration of the angles, the unicycle had a transient phase of about a second. After 1.2 s, either the unicycle had fallen or the learned controller had managed to balance it very closely to the desired upright position. The success rate was approximately 93%; bringing the unicycle upright from extreme initial configurations was sometimes impossible due to the torque constraints.

Table 1. PILCO's data efficiency scales to high dimensions.

	cart-pole	cart-double-pole	unicycle
state space	$\mathbb{R}^4$	$\mathbb{R}^6$	$\mathbb{R}^{12}$
# trials	≤ 10	20–30	≈ 20
experience	≈ 20 s	≈ 60 s–90 s	≈ 20 s–30 s
parameter space	$\mathbb{R}^{305}$	$\mathbb{R}^{1816}$	$\mathbb{R}^{28}$

3.4. Data Efficiency

Tab. 1 summarizes the results presented in this paper: For each task, the dimensionality of the state and parameter spaces is listed together with the required number of trials and the corresponding total interaction time. The table shows that PILCO can efficiently find good policies even in high dimensions. The generality of this statement depends on both the complexity of the dynamics model and the controller to be learned.

In the following, we compare PILCO's data efficiency (required interaction time) to other RL methods that learn previously discussed tasks from scratch, i.e., without informative prior knowledge. This excludes methods relying on known dynamics models or expert demonstrations.

Fig. 5 shows the interaction time with the *cart-pole* system required by PILCO and algorithms in the literature that solved this task from scratch (Kimura & Kobayashi, 1999), (Doya, 2000), (Coulom, 2002), (Wawrzynski & Pacut, 2004), (Riedmiller, 2005), (Raiko & Tornio, 2009), (van Hasselt, 2010). Dy-

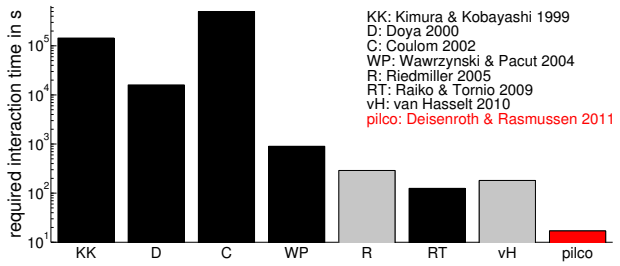


Figure 5. Data efficiency for learning the cart-pole task in the absence of expert knowledge. The horizontal axis chronologically orders the references according to their publication date. The vertical axis shows the required interaction time with the cart-pole system on a log-scale.

namics models were only learned by Doya (2000) and Raiko & Tornio (2009), using RBF networks and multi-layered perceptrons, respectively. Note that both the NFQ algorithm by Riedmiller (2005) and the (C)ACLA algorithms by van Hasselt (2010) were applied to balancing the pole *without* swing up. In all cases without state-space discretization, cost functions similar to ours were used. Fig. 5 demonstrates PILCO's data efficiency since PILCO outperforms any other algorithm by at least one order of magnitude.

We cannot present comparisons for the *cart-double pendulum* swing up or *unicycle* riding: To the best of our knowledge, fully autonomous learning has not yet succeeded in learning these tasks from scratch.

4. Discussion and Conclusion

Trial-and-error learning leads to some limitations in the discovered policy: PILCO is not an optimal control method; it merely finds *a* solution for the task. There are no guarantees of global optimality: Since the optimization problem for learning the policy parameters is not convex, the discovered solution is invariably only locally optimal. It is also conditional on the experience the learning system was exposed to. In particular, the learned dynamics models are only confident in areas of the state space previously observed.

PILCO exploits analytic gradients of an approximation to the expected return J^π for indirect policy search. Obtaining nonzero gradients depends on two factors: the state distributions $p(\mathbf{x}_1), \ldots, p(\mathbf{x}_T)$ along a predicted trajectory and the width σ_c of the immediate cost in Eq. (25). If the cost is very peaked, say, a 0-1 cost with 0 being exactly in the target and 1 otherwise, *and* the dynamics model is poor, i.e., the distributions $p(\mathbf{x}_1), \ldots, p(\mathbf{x}_T)$ *nowhere* cover the target region (implicitly defined through σ_c), PILCO obtains gradients with value zero and gets stuck in a local optimum.

Although PILCO is relatively robust against the choice of the width σ_c of the cost in Eq. (25), there is no guarantee that PILCO always learns with a 0-1 cost. However, we have evidence that PILCO *can* learn with this cost, e.g., PILCO could solve the cart-pole task with a cost width σ_c to 10^{-6} m. Hence, PILCO's unprecedented data efficiency cannot solely be attributed to any kind of reward shaping.

One of PILCO's key benefits is the reduction of model bias by explicitly incorporating model uncertainty into planning and control. PILCO, however, does not take temporal correlation into account. Instead, model uncertainty is treated similarly to uncorrelated noise. This can result in an under-estimation of model uncertainty (Schneider, 1997). On the other hand, the moment-matching approximation used for approximate inference is typically a conservative approximation. Simulation results suggest that the predictive distributions $p(\mathbf{x}_1), \ldots, p(\mathbf{x}_T)$ used for policy evaluation are usually not overconfident.

The probabilistic dynamics model was crucial to PILCO's learning success: We also applied the PILCO-framework with a *deterministic dynamics model* to a simulated cart-pole swing-up. For a fair comparison, we used the posterior mean function of a GP, i.e., only the model uncertainty was discarded. Learning from scratch with this deterministic model was unsuccessful because of the missing representation of model uncertainty: Since the initial training set for the dynamics model did not contain states close to the target state, the predictive model was overconfident during planning (see Fig. 1, center). When predictions left the regions close to the training set, the model's extrapolation eventually fell back to the uninformative prior mean function (with *zero* variance) yielding essentially useless predictions.

We introduced PILCO, a practical model-based policy search method using analytic gradients for policy improvement. PILCO advances state-of-the-art RL methods in terms of learning speed by at least an order of magnitude. Key to PILCO's success is a principled way of reducing model bias in model learning, long-term planning, and policy learning. PILCO does not rely on expert knowledge, such as demonstrations or task-specific prior knowledge. Nevertheless, PILCO allows for unprecedented data-efficient learning from scratch in continuous state and control domains. Demo code will be made publicly available at `http://mloss.org`.

The results in this paper suggest using probabilistic dynamics models for planning and policy learning to account for model uncertainties in the small-sample case—even if the underlying system is deterministic.

Acknowledgements

We are very grateful to Jan Peters and Drew Bagnell for valuable suggestions concerning the presentation of this work. M. Deisenroth has been supported by ONR MURI grant N00014-09-1-1052 and by Intel Labs.

References

Abbeel, P., Quigley, M., and Ng, A. Y. Using Inaccurate Models in Reinforcement Learning. In *Proceedings of the ICML*, pp. 1–8, 2006.

Atkeson, C. G. and Santamaría, J. C. A Comparison of Direct and Model-Based Reinforcement Learning. In *Proceedings of the ICRA*, 1997.

Bagnell, J. A. and Schneider, J. G. Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods. In *Proceedings of the ICRA*, pp. 1615–1620, 2001.

Coulom, R. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

Deisenroth, M. P. *Efficient Reinforcement Learning using Gaussian Processes*. KIT Scientific Publishing, 2010. ISBN 978-3-86644-569-7.

Deisenroth, M. P., Rasmussen, C. E., and Peters, J. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, 2009.

Deisenroth, M. P., Rasmussen, C. E., and Fox, D. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of R:SS*, 2011.

Doya, K. Reinforcement Learning in Continuous Time and Space. *Neural Computation*, 12(1):219–245, 2000. ISSN 0899-7667.

Engel, Y., Mannor, S., and Meir, R. Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning. In *Proceedings of the ICML*, pp. 154–161, 2003.

Fabri, S. and Kadirkamanathan, V. Dual Adaptive Control of Nonlinear Stochastic Systems using Neural Networks. *Automatica*, 34(2):245–253, 1998.

Forster, D. Robotic Unicycle. Report, Department of Engineering, University of Cambridge, UK, 2009.

Kimura, H. and Kobayashi, S. Efficient Non-Linear Control by Combining Q-learning with Local Linear Controllers. In *Proceedings of the ICML*, pp. 210–219, 1999.

Ko, J., Klein, D. J., Fox, D., and Haehnel, D. Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp. In *ICRA*, pp. 742–747, 2007.

Naveh, Y., Bar-Yoseph, P. Z., and Halevi, Y. Nonlinear Modeling and Control of a Unicycle. *Journal of Dynamics and Control*, 9(4):279–296, 1999.

Peters, J. and Schaal, S. Policy Gradient Methods for Robotics. In *Proceedings of the IROS*, pp. 2219–2225, 2006.

Quiñonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. E. Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. In *Proceedings of the ICASSP*, pp. 701–704, 2003.

Raiko, T. and Tornio, M. Variational Bayesian Learning of Nonlinear Hidden State-Space Models for Model Predictive Control. *Neurocomputing*, 72(16–18):3702–3712, 2009.

Rasmussen, C. E. and Kuss, M. Gaussian Processes in Reinforcement Learning. In *NIPS*, pp. 751–759. 2004.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Riedmiller, M. Neural Fitted Q Iteration—First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Proceedings of the ECML*, 2005.

Schaal, S. Learning From Demonstration. In *NIPS*, pp. 1040–1046. 1997.

Schneider, J. G. Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning. In *NIPS*, pp. 1047–1053. 1997.

van Hasselt, H. *Insights in Reinforcement Learning*. Wöhrmann Print Service, 2010. ISBN 978-90-39354964.

Wawrzynski, P. and Pacut, A. Model-free off-policy Reinforcement Learning in Continuous Environment. In *Proceedings of the IJCNN*, pp. 1091–1096, 2004.

Wilson, A., Fern, A., and Tadepalli, P. Incorporating Domain Models into Bayesian Optimization for RL. In *ECML-PKDD*, pp. 467–482, 2010.

Zhong, W. and Röck, H. Energy and Passivity Based Control of the Double Inverted Pendulum on a Cart. In *Proceedings of the CCA*, pp. 896–901, 2001.