MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Model-Based Reinforcement Learning Using Monte Carlo Gradient Estimation

Amadio, Fabio; Dalla Libera, Alberto; Carli, Ruggero; Romeres, Diego

TR2021-108 September 16, 2021

Abstract

We propose an MBRL algorithm named Monte Carlo Probabilistic Inference for Learning COntrol (MC-PILCO). MC-PILCO is a policy gradient algorithm, which uses GPs to model the system dynamics, but it overcomes PILCO's limitations by relying on a particle-based method to compute long-term predictions, instead of using moment matching.

Automatica.it 2021

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Model-Based Reinforcement Learning Using Monte Carlo Gradient Estimation

Fabio Amadio amadiofa@dei.unipd.it
Alberto Dalla Libera dallaliber@dei.unipd.it
Ruggero Carli carlirug@dei.unipd.it
Diego Romeres romeres@merl.com

I. Introduction

Reinforcement learning (RL) [1] has shown the potential to provide an automated framework for learning different control applications from scratch. However, model-free RL (MFRL) algorithms might require a massive amount of interactions with the environment to solve the assigned task. This data inefficiency puts a limit to RL's potential in real-world applications. A promising way to increase data-efficiency is model-based reinforcement learning (MBRL), which uses data to build a model of the environment and exploit it to plan control.

However, MBRL methods are effective only if their models resemble accurately the environment. Hence, the use of stochastic models becomes necessary to capture uncertainty. Gaussian Processes (GPs) [2] have been commonly used in RL precisely for their capacity to handle uncertainty [3]. PILCO (Probabilistic Inference for Learning COntrol) [4] is a successful MBRL algorithm that uses GPs and gradient-based policy search to solve different control problems [5].

In PILCO, long-term predictions are computed analytically, approximating the distribution of the next state with a Gaussian distribution by means of moment matching. This approximation allows computing the policy gradient in closed form. However, it also introduces two relevant limitations. (i) Moment matching can model only unimodal distributions. This fact, besides being a potentially incorrect assumption on the system dynamics, introduces relevant limitations related to initial conditions. (ii) The computation of the moments is shown to be tractable only when considering Squared Exponential (SE) kernels. The latter limitation might be very restrictive, as GPs with SE kernel show poor generalization in points that have not been seen during training [6].

We propose an MBRL algorithm named Monte Carlo Probabilistic Inference for Learning COntrol (MC-PILCO). MC-PILCO is a policy gradient algorithm, which uses GPs to model the system dynamics, but it overcomes PILCO's limitations by relying on a particle-based method to compute long-term predictions, instead of using moment matching.

II. Problem Formulation

Consider the discrete-time system described by the (possibly stochastic) unknown transition function $f(\cdot,\cdot)$,

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t),$$

where, at each time step t, $\boldsymbol{x}_t \in \mathbb{R}^{d_{\boldsymbol{x}}}$ and $\boldsymbol{u}_t \in \mathbb{R}^{d_{\boldsymbol{u}}}$ are, respectively, the state and the inputs of the system. The cost $c(\boldsymbol{x}_t)$ defines the penalty for being in state $\boldsymbol{x}_t$. Inputs are chosen according to a general-purpose policy function $\pi_{\boldsymbol{\theta}} : \boldsymbol{x} \mapsto \boldsymbol{u}$, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$. The objective is to find the set of parameters that minimizes the expected cumulative cost over a finite number of time steps T, i.e.,

$$J(\boldsymbol{\theta}) = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{x}_t}\left[c(\boldsymbol{x}_t)\right], \tag{1}$$

starting from an initial state distributed according to $p(\boldsymbol{x}_0)$. We assume to have access to (noisy) measures of the full state.

III. MC-PILCO

The proposed algorithm, MC-PILCO, consists of the succession of several trials; i.e. attempts to solve the desired task. Generally speaking, each trial involves the following phases:

- *Model Learning*: the data collected from all the previous interactions are used to train a model of the system dynamics (at the first iteration data are collected applying a sequence of exploratory actions, e.g. random controls);
- *Policy Optimization*: the policy is optimized in order to minimize an estimate of the cost $J(\boldsymbol{\theta})$. Model predictions are used to estimate gradient $\nabla_{\boldsymbol{\theta}} J$ and policy parameters $\boldsymbol{\theta}$ are updated following a gradient-descent approach.
- *Policy Execution*: the optimized policy is applied to the system and the data are stored for model improvement.

A. Gaussian Processes for Model Learning

Given a set of state-action pairs collected during the interactions with the system, it is possible to use GP regression to train a probabilistic model that approximates $f(\cdot)$. A common strategy is to model the evolution of each state dimension with a distinct zero mean GP. Let us indicate with $x^{(i)}$ the i-th component of the state, for $i \in \{1, \dots, d_{\boldsymbol{x}}\}$. The i-th GP takes $\tilde{\boldsymbol{x}}_t$ as input, and predicts $x^{(i)}_{t+1} - x^{(i)}_t$. Let us denote with $\mathcal{D}$ the data set of state-action pairs, the GPs provide a closed form expression of $\boldsymbol{p}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t, \mathcal{D})$, the posterior distribution of the estimated state at time $t+1$. The GPs are completely characterized by their kernel functions that represent our belief on the a priori covariance. With respect to this, PILCO is restricted to use only the SE kernel, due to the moment matching approximation it makes for computing the policy gradient. MC-PILCO does not need this kind of assumptions, hence we can adopt also more structured functions, as polynomial and semi-parametrical kernels.

Fabio Amadio, Alberto Dalla Libera and Ruggero Carli are with the Department of Information Engineering, University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy. Diego Romeres is with the Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139.

B. Particle-based Policy Optimization

The predictive model is now employed to optimize the policy parameters $\boldsymbol{\theta}$ following a particle-based policy gradient strategy. MC-PILCO computes $\hat{J}(\boldsymbol{\theta})$, an approximation of $J(\boldsymbol{\theta})$ in (1) exploiting the distribution $\boldsymbol{p}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t, \mathcal{D})$. After that, it updates $\boldsymbol{\theta}$ with a gradient-based procedure.

The computation of $\hat{J}(\boldsymbol{\theta})$ entails the simulation of the effects of $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ on M independent state particles by cascading the one-step-ahead stochastic predictions for T time steps. In particular, let $\boldsymbol{x}_t^{(m)}$, for $m = 1, \dots, M$, represent the position of the M state particles. They are all initialized by sampling from the distribution $p(\boldsymbol{x}_0)$. Then, at each time step t the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ selects the next control actions $\boldsymbol{u}_t^{(m)}$ for every particle. We adopted, as policy, a RBF network with saturated output, but one can use any kind of differentiable function. Finally, the state of the M particles at the next time step, $t+1$, are obtained by forward sampling from $\boldsymbol{p}(\boldsymbol{x}_{t+1}^{(m)}|\boldsymbol{x}_t^{(m)}, \boldsymbol{u}_t^{(m)}, \mathcal{D})$ with $m = 1 \dots M$. This procedure is iterated for T time steps, obtaining M different particle trajectories $\{\{\boldsymbol{x}_t^{(m)}\}_{m=1}^{M}\}_{t=0}^{T}$, that simulate the results of the policy. The sample mean of the costs incurred by the particles provides an estimate of the expected cumulative cost, namely

$$\hat{J}(\boldsymbol{\theta}) = \sum_{t=0}^{T} \left(\frac{1}{M} \sum_{m=1}^{M} c\left(\boldsymbol{x}_t^{(m)}\right) \right). \tag{2}$$

The computational graph resulting from (2) allows us to compute $\nabla_{\boldsymbol{\theta}} \hat{J}(\boldsymbol{\theta})$, i.e., the gradient of $\hat{J}(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$, by means of backpropagation, exploiting the reparametrization trick [7] to propagate the gradient through the stochastic operations. Finally, a stochastic gradient descent algorithm, e.g. Adam [8], can use the estimated gradient to update $\boldsymbol{\theta}$.

Note how this particle-based strategy for computing the gradient does not make any approximation regarding the long-term state distributions, as it does PILCO with moment matching. This is the reason why MC-PILCO can handle any kernel and can also model multimodal distributions.

IV. Experimental Results

Initially, we tested MC-PILCO on a simulated cart-pole system, where it outperformed two state-of-the-art GP-based MBRL algorithms, such as the aforementioned PILCO, and Black-DROPS [9]. Fig. 1 reports the result obtained in 50 different runs of the experiment, in terms of median cumulative cost per trial (with confidence intervals). MC-PILCO achieves the best data efficiency and consistency in finding the optimal policy.

After that, we also applied the algorithm to the two real systems depicted in Fig. 2. More precisely, we modified the original formulation of MC-PILCO to take into account the inevitable necessity of performing state estimation, during both the model learning and policy optimization phases. Our method managed to learn from scratch how to control both systems. Details about the experiments[1] can be found in [10].

[1]A video is available at https://youtu.be/--73hmZYaHA.

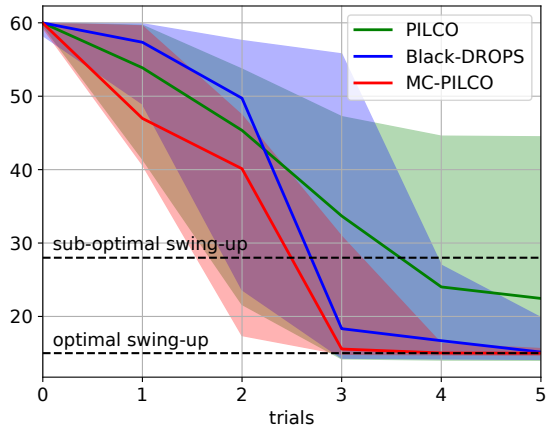

Fig. 1. Median and confidence interval of the cumulative cost per trial obtained with PILCO, Black-DROPS and MC-PILCO.

Fig. 2. Real setups: (left) Furuta pendulum, (right) ball-and-plate.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[3] Malte Kuss and Carl E Rasmussen. Gaussian processes in reinforcement learning. In *Advances in neural information processing systems*, pages 751–758, 2004.

[4] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[5] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics: Science and Systems VII*, pages 57–64, 2011.

[6] D. Nguyen-Tuong and J. Peters. Using model knowledge for learning inverse dynamics. In *2010 IEEE International Conference on Robotics and Automation*, pages 2677–2682, 2010.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51–58. IEEE, 2017.

[10] Fabio Amadio, Alberto Dalla Libera, Riccardo Antonello, Daniel Nikovski, Ruggero Carli, and Diego Romeres. Model-based policy search using monte carlo gradient estimation with real systems application. *arXiv preprint:2101.12115*, 2021.