Fleet Reinforcement Learning using Dependent Gaussian Processes

Timothy Verstraeten
Vrije Universiteit Brussel
tiverstr@vub.ac.be

Peter Vrancx
Vrije Universiteit Brussel
pvrancx@vub.ac.be

Ann Nowé
Vrije Universiteit Brussel
anowe@vub.ac.be

Abstract

Physical systems are progressively moving forward from local controllers towards smarter cloud-based architectures. This allows similar inter-connected reinforcement learning agents to share information in order to obtain a more global perspective on the control task at hand. However, the local context and inherent properties of these agents are in practice not identical, making the approach of naively combining gathered information unsuitable. We propose to detect correlations between the observed dynamics of similar agents through dependent Gaussian processes, allowing us to effectively share information between these agents. We validate our approach in a pendulum swing-up and cart-pole setting. Our approach significantly outperforms the naive method of combining all samples into one model, by quickly and accurately estimating dependencies. In future work, we expect to improve our results by measuring correlations between rewards.

1 Introduction

Reinforcement learning (RL) [11] allows goal-oriented agents that are able to sense and act based on their current state to learn an optimal control policy by interacting with the environment. However, those interactions are quite expensive in many realistic control systems, such as low cost robot arms [4] and hard to maintain wind farms [5]. Moreover, the local context and inherent properties (e.g., hardware) of a control system causes poor estimations of global events due to noisy measurements.

When a group of similar devices is executing the same control task, aggregating information allows one to obtain a wider view of the problem, and achieve more effective and robust control. This is the idea of *fleet control*. Agents have to collaborate in order to satisfy the control task requirements and share noisy measurements about events in the environment that are potentially not observable by other agents. However, naively combining all information is unsuitable when agents are not identical. For example, due to the aging of wind turbine blades, friction parameters change over time. This causes discrepancies in the outcome of similar control actions taken by different turbines in the wind farm. Therefore, there is a need for turbine-specific control policies, even though the underlying dynamics are identical for each turbine.

As a first step towards a fleet learning framework, similarities between agents have to be inferred in order to effectively share relevant information. We propose a Bayesian model-based fleet RL method using dependent Gaussian processes (GP). GPs are well known for their flexibility and are able to find complex patterns using a limited set of training data. We introduce a correlation parameter for each pair of agents in the covariance kernel of the processes [2], such that similar agents share an informative prior over the environment dynamics. We focus on the PILCO method for inferring a controller from the estimated dynamical GP [3].

Our work is closely related to Bayesian multi-task RL [6, 12]. The authors define a hierarchical Bayesian model that generates an arbitrary number of informative priors for classes of similar MDPs or value functions. However, these MDPs and functions must inherently have a low-dimensional parametric form to be clustered effectively and are confined to use the prior associated with their class. We are able to define informative priors between each pair of agents. Moreover, our approach fits nicely into the GP framework, requiring no additional model or inference algorithm for clustering.

We give background information on RL and PILCO in respectively Sections 2 and 3. Our dependent GP RL method is introduced in Section 4. We validated our approach on a pendulum swing-up and cart-pole setting and explain the results in Section 5. We conclude with discussing future work in Section 6.

2 Reinforcement Learning

In reinforcement learning, an agent learns a policy that maximizes the cumulative reward for achieving a certain goal over a sequence of state transitions. The problem is formulated as a Markov decision process (MDP) $M = (S, A, T, \gamma, R)$ [8], where S, A are the state and action spaces, $T : S \times A \times S \rightarrow \mathbb{R}$ is a probabilistic transition function, γ is the discount factor determining the importance of future rewards and $R : S \times A \times S \rightarrow \mathbb{R}$ is the immediate reward function.

An agent behaves according to a policy $\pi : S \rightarrow A$. Optimizing π using policy iteration consists of two parts: policy evaluation and policy improvement. The former refers to assessing the quality of a policy by estimating the expected discounted long-term reward from a given starting state s_0:

$$V^{\pi}(s) = \mathbb{E}_{T,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s = s_0\right] \tag{1}$$

where r_{t+1} is the reward obtained upon taking action a_t at state s_t.

Policy improvement refers to choosing a new policy of higher quality. Model-based methods use an additional model of the environment to perform long-term planning in order to find the best action [11] and are preferred when real-world experience is costly. However, model bias is a recurring issue in these methods, as the prediction error grows when simulating one-step state transitions in an estimated model of the actual environment [1]. A probabilistic dynamics model reduces this bias immensely, since the uncertainty in the predictions is fully accounted for [10].

3 Gaussian Process Dynamics Model

We adopt the model-based method PILCO [3], which models the environment dynamics as a Gaussian process (GP) [9]. A GP is an infinite collection of random variables, of which each finite subset is jointly normally distributed. It is defined by a mean function $m(x)$ and covariance kernel $k(x, x')$. Intuitively, a process $f(x)$ describes probabilities over a set of functions, where $m(x)$ is the most likely function of the set and $k(x, x')$ specifies the covariances between the random function values on inputs x and x'. This relationship between inputs defines the properties of the functions considered by the process, such as continuity, periodicity and length scale.

Similar to multi-variate normal distributions, a posterior process $f(x) \mid \mathcal{D}$ given observations $\mathcal{D} = (X, y)$ can be inferred. When used as a regression technique, the mean of the posterior process gives the best linear regularized predictor on a set of basis functions $k(x, x_i)$. Thus, it is equivalent to solving the following system of N linear equations

$$y_j = \sum_{i=1}^{N} w_i(k(x_j, x_i) + \sigma_n^2 \delta(x_j, x_i)), 1 \leq j \leq N$$

where $\{(x_j, y_j)\}_{j=1}^{N}$ is the training data and σ_n^2 is white Gaussian noise. Note that the complexity of such a model grows with the number of observed samples, allowing it to capture increasingly complex nonlinear patterns. Computing the posterior mean of a random variable f^* at input x^* is done as follows:

$$\mathbb{E}\left[f^* \mid x^*, \mathcal{D}\right] = K(x^*, X)(K(X, X)^{-1} + \sigma_n^2 I)^{-1} y$$

where $K(X, X)$ and $K(x^*, X)$ are the resulting matrices when applying kernel k to the respective inputs.

PILCO uses a zero-mean GP with the squared exponential (SE) kernel k_{SE} to model the random dynamical process $f(s,a) \sim \mathcal{GP}(0, k_{SE}([s,a], [s',a']))$ The SE kernel is widely used, as it models continuous, stationary and infinitely differentiable processes. It is defined as follows:

$$k^{\theta}_{SE}(x,x') = \alpha^2 \exp(-0.5(x-x')^T \Lambda^{-1}(x-x'))$$

where α^2 is the magnitude of the output and Λ is a diagonal matrix of length scales per input dimension. We denote the combination of these hyperparameters as θ, which can be optimized by maximizing the likelihood of the data $p(y \mid X, \theta)$.

Given such a dynamics model, PILCO estimates the expected return $V^{\pi}(s_0)$ for a given policy π by iteratively propagating Gaussian approximations of state distributions through the dynamics model until the horizon T is met. This gives an analytic expression for the expected return that can be derived and optimized w.r.t. the policy parameters using gradient descent techniques.

4 Correlations between Agents

Fleet control deals with the operation of similar inter-connected devices. Our goal is to take advantage of this fleet aspect by aggregating information over similar agents, such that less real-world experience is necessary per agent in order to learn a good representative model of its environment.

However, in practice, the underlying properties of each agent are not identical. An agent might have small technical discrepancies or a limited view on global events due to its location. These aspects cause the agent to have its own unique view on the environment dynamics. Therefore, naively combining the samples from all agents into one model might capture an inaccurate representation of the environment dynamics.

We propose to extend the PILCO method to capture the dynamics observed by multiple agents i as a set of dependent GPs f_i. Each data set is annotated with a label i, referring to the corresponding process f_i. Inter-process covariances are captured by a matrix C, such that entry $C_{i,i'}$ specifies the relationship between agents i and i'. We obtain the following covariance kernel [2]:

$$k^{\theta}([x,i],[x',i']) = k^{\theta_x}_{SE}(x,x') C^{\theta_i}_{i,i'}$$

where input points x and x' are respectively observed by agents i and i'. Thus, for similar agents, the off-diagonal entries in C should be high, such that the posterior process $f_i \mid \mathcal{D}_i, \mathcal{D}_{i'}$ yields a better representation of the dynamics than $f_i \mid \mathcal{D}_i$. When agents are extremely different, the entries of C should be close to zero, rendering the processes independent.

It is important that C is positive semi-definite in order to maintain a valid covariance kernel. We use the spherical parametrization θ_i of C [7], which allows us to transform any vector of unconstrained parameters into a unique covariance matrix. We consider C as part of the hyperparameters $\theta = [\theta_x, \theta_i]$. As the spherical transformation is differentiable w.r.t. each parameter, we can use gradient descent techniques to maximize the likelihood $p(y \mid X, \theta)$ of the data (X, y) w.r.t. each element of θ.

Policy iteration in a fleet setting can be done by learning processes f_i and their dependencies through the proposed covariance kernel k on batches of training data collected for each agent i. Similar to the PILCO method, we are then able to propagate one-step predictions through a process f_i and derive an expression for the expected return $V^{\pi_i}(s_0)$, which can be optimized to improve the policy π_i.

5 Experiments

We validate our approach on the continuous pendulum swing-up and cart-pole domains. In the former domain, the objective is to swing up and balance a pole. A state consists of the angle and angular velocity of the pole and an action is a force of maximum $2.5N$ applied to either side of the pole.

The cart-pole setting is similar, but requires a pole to be balanced on top of a horizontally moving cart, while the cart has to stay as close as possible to its starting position, making it a more complex task to solve. In addition to the state of the pendulum, the position and velocity of the cart are also observed. An action is a force of maximum $10N$ applied to either side of the cart.

For both domains, we consider the following reward function:

$$R(s) = \exp(-||s - s_{target}||^2/0.5^2)$$

where s_{target} is the pole at its equilibrium and the cart at its starting position. We start in state s_0 in which the pole is downwards, and we maximize the expected undiscounted cumulative reward $V^\pi(s_0)$ over an horizon of 4 seconds. The sampling frequency is 0.1 seconds, giving us 40 samples per episode.

After each episode, we train a dynamics model using all previous samples, and estimate the policy parameters θ_p using line search. A policy π^{θ_p} is represented by a regularized non-linear RBF network using 10 Gaussian basis functions

$$\pi^{\theta_p}(s) = \sum_{i=1}^{10} w_i \exp(-0.5(s-\mu_i)^T \Lambda^{-1}(s-\mu_i))$$

where $\theta_p = [w, \mu, \Lambda]$ are the policy parameters.

5.1 Pendulum Swing-Up

We consider two agents learning to swing up and balance their own pendulum by consecutively modeling the environment dynamics through two dependent processes and deriving their own policy. Each agent has the same unknown dynamics parameters, except for the pole mass, which we put to 1.0 kg for the first agent and 1.5 kg for the second. We focus on the former agent and report its performance.

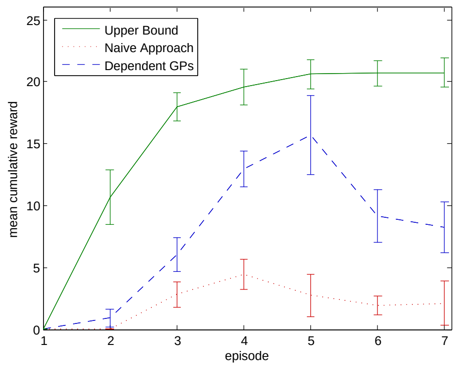

Figure 1: The cumulative reward over 7 episodes when inferring one process for two identical agents ('Upper Bound'), one process for two different agents ('Naive Approach') and two processes for two different agents ('Dependent GPs'). The return is averaged over 10 trials and the 95% confidence interval is given.

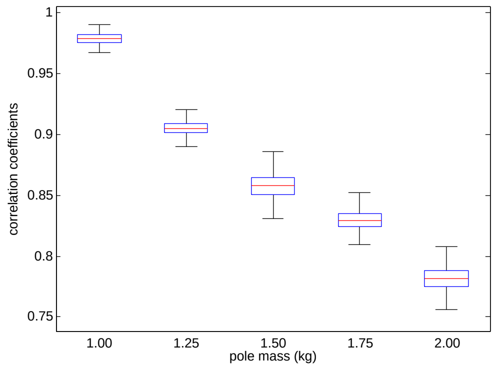

Figure 2: The correlation coefficients learned in the covariance matrix C between an agent with a pole mass of 1 kg and agents with variable pole masses. The correlations are averaged over 10 trials and both the means and standard deviations are given.

Figure 1 shows the return, averaged over 10 trials, for the agent with a pole mass of 1.0 kg during 7 episodes, which is enough learning time for PILCO to balance the pole. We define an upper bound on the performance that we can reach, by inferring a single process for two *identical* agents. As expected, the naive approach of maintaining one process for two different agents does not work, as the agents are incapable of learning a single policy that works for both of them. Our dependent GPs are able to extract the correlations between the two different dynamics and infer two separate policies, achieving a significantly better performance than the naive approach.

We noticed in the individual trials that the return is slightly oscillating when the agents are different, which explains the increase in variance for our approach. This is due to the fact that PILCO notices a decrease in performance for one agent when optimizing for another. This causes it to switch focus to the other agent, creating this oscillating effect.

Even though our method significantly outperforms the naive approach, we expected the performance to be more comparable to upper bound. In order to investigate this, we examine the correlations inferred during learning (see Figure 2). We keep the pole mass fixed at 1.0 kg for one agent and vary it for the other.

Although the correlation coefficients are decreasing, we notice that the dynamics stay highly correlated. This is expected, as changing the pole mass in the pendulum setting has a fairly linear impact on the dynamics. However, we speculate that such high correlations do not sufficiently separate the two processes in order to obtain the two required policies, and we expect that capturing correlations between received rewards will improve results. Nevertheless, the correlation coefficients are inferred accurately by the dependent GPs, which allows our agents to only learn from the most similar agents. We test this setting in the cart-pole domain.

5.2 Cart-Pole

Even though the resulting policies of different agents are still too similar when derived through our method, the correlations between the dynamics are determined quickly and accurately. Therefore, we will focus on integrating a new agent into the fleet by allowing it to share samples with only the most similar agent. We expect that learning correlations is faster than the optimization of the policy itself, such that using samples from a similar agent increases the learning speed of the target agent.

We test this hypothesis in the more complex cart-pole domain. We assume a group of 4 agents which already learned a control policy to balance the pole on top of a cart, and a target agent that has to learn this from scratch. At the start of each episode, the target agent computes the dependent processes between itself and all other agents in order to obtain the correlations between them. Then it augments its data set with the samples of the most similar agent at that moment. This means that the resulting target policy is derived from the dynamical process inferred from both the samples of the target agent and the most similar agent at that moment. Each agent has a different set of parameters, which are unknown during learning (see Table 1). Note that agent 1 is similar to our target and thus the best candidate to share samples with.

agents	**pole length (m)**	**pole mass (kg)**	**cart mass (kg)**
target	0.50	0.50	0.50
1	0.55	0.50	0.50
2	0.70	0.50	0.30
3	0.30	0.30	0.30
4	0.50	0.50	0.80

Table 1: The unknown parameters of the target and each agent in the fleet. The mass and length of the pole, as well as the mass of the cart vary over all agents. Agent 1 is the most similar to the target.

We compared our method to both the cases in which the target agent learns from everyone and on its own. We measure the performance of these approaches by averaging the return of our target agent over 10 trials and plot the return over 7 episodes in Figure 3.

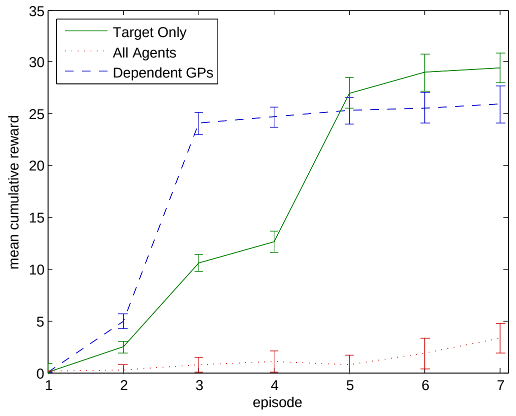

Figure 3: The cumulative reward over 7 episodes when inferring a process using the samples of only the target agent (‘Target Only’), the samples over all agents (‘All Agents’), and the samples of both the target and the most similar agent (‘Dependent GPs’). The return is averaged over 10 trials and the 95% confidence interval is given.

We can see that aggregating the samples obtained by all agents slows down learning immensely for the target. Moreover, it seems that the target is barely able to lift up the pole. We think this is caused specifically by incorporating the samples of agent 3 (see Table 1). The target expects to have a lightweight cart and pole, such that not a lot of force is necessary for a swing-up, while the actual cart and pole are quite heavy.

Using dependent GPs, the target is able to detect the most similar agent 79% of the time over all episodes and trials. Sharing samples with only this agent yields a significant improvement in terms of learning speed, maintaining a final performance comparable to the case in which the target agent learns independently.

6 Conclusion

We introduced a method to extract correlations between state-transitions observed by different agents using dependent Gaussian processes. We first evaluated our approach on a pendulum swing-up, showing a significant improvement over the naive approach and the ability to extract correlations between similar dynamics. However, we expect that better performance can be achieved by investigating the returns observed by the agents. We also evaluated our method on a cart-pole setting, in which a target agent had the opportunity to share samples with a single agent from an already existing fleet. Our approach significantly improved the learning speed, indicating the ability of detecting correlations between agents quickly and accurately. The tests also show that combining samples over similar, but nonidentical, agents can drastically decrease the overall performance of all agents.

In future work, we will extend our approach to also find correlations between the rewards and returns observed by different agents. We expect this will help us identify larger impacts on control decisions, increasing the performance of our method. Additionally, we will investigate the multi-agent interactions that arise in more realistic fleet systems.

References

[1] C. G. Atkeson and J. C. Santamaría. A comparison of direct and model-based reinforcement learning. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, 1997.

[2] E. V. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-Task Gaussian Process Prediction. *Advances in Neural Information Processing Systems*, 20:153–160, 2008.

[3] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proc. of the International Conference on Machine Learning*. 2011.

[4] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science & Systems*, volume 7. 2011.

[5] J. Helsen, G. L. De Sitter, and P. J. Jordaens. Long-term monitoring of wind farms using big data approach. In *Proc. of the 2nd IEEE Big Data Computing Service and Applications conference*, pages 265–268. 2016.

[6] A. Lazaric and M. Ghavamzadeh. Bayesian multi-task reinforcement learning. In *Proc. of the 27th International Conference on Machine Learning*, pages 599–606. Omnipress, 2010.

[7] J. C. Pinheiro and D. M. Bates. Unconstrained Parameterizations for Variance-Covariance Matrices. *Statistics and Computing*, 6:289–296, 1996.

[8] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[9] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA, 2006.

[10] J. G. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. In M. I. Jordan and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 1047–1053. MIT Press, 1997.

[11] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press Cambridge, 1998.

[12] A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proc. of the 24th International Conference on Machine Learning*, pages 1015–1022, 2007.