Qualitative Transfer for Reinforcement Learning with Continuous State and Action Spaces

Esteban O. Garcia, Enrique Munoz de Cote, and Eduardo F. Morales

Instituto Nacional de Astrófisica, Óptica y Electrónica,
Luis Enrique Erro # 1. Tonantzintla, Puebla, México
{eomargr,jemc,emorales}@inaoep.mx
http://ccc.inaoep.mx

Abstract. In this work we present a novel approach to transfer knowledge between reinforcement learning tasks with continuous states and actions, where the transition and policy functions are approximated by Gaussian Processes (GPs). The novelty in the proposed approach lies in the idea of transferring qualitative knowledge between tasks, we do so by using the GPs' hyper-parameters used to represent the state transition function in the source task, which represents qualitative knowledge about the type of transition function that the target task might have. We show that the proposed technique constrains the search space, which accelerates the learning process. We performed experiments varying the relevance of transferring the hyper-parameters from the source task into the target task and show, in general, a clear improvement in the overall performance of the system when compared to a state of the art reinforcement learning algorithm for continuous state and action spaces without transfer.

Keywords: Transfer learning, Reinforcement learning, Gaussian Processes, Hyper-parameters.

1 Introduction

The objective in reinforcement learning (RL) is to find a sequence of actions that maximizes a long-term cumulative reward. An RL algorithm achieves such an objective by exploring the world and collecting information about it in order to determine such sequence of actions [16]. RL algorithms provide mechanisms to learn solutions without the need of human experience. However, when these are applied to real world problems, two major problems arise: (i) a large number of samples or interaction time with the environment is needed to learn an optimal solution, and (ii) after an agent has learned to solve a task, if it is required to solve a different (although similar) task, the learning process must be restarted.

Typically, RL is used on discrete state-action spaces, despite the fact that, most real-world problems involve continuous domains and discretizations of the domain variables may lead to very large discrete state-action spaces or imprecise policy functions that may harm the learning process [7,3]. Several approaches

J. Ruiz-Shulcloper and G. Sanniti di Baja (Eds.): CIARP 2013, Part I, LNCS 8258, pp. 198–205, 2013.

have been proposed to deal with continuous domains, (e.g., [9,8,11,7,2]), most of them use function approximators. In particular, Gaussian Processes (GPs) have been used to represent value functions [6,5,14,1], and more recently, to represent transition function models with very promising results [12,13,4,3,2]. In this paper, we use GPs to represent policy and state transition functions.

A common approach to lessen the problem of learning a new, although similar task is use transfer learning (TL). Several approaches have been proposed where the source and target tasks may have different transition functions, state spaces, start or goal states, reward functions or action sets [18]. In this paper, we assume that there is only one source task and that the source and target tasks have the same variables.

Most of the TL methods for RL that use the same assumptions as us, focus on discrete tasks and model-free learning methods. In [10] and [17] model-based learning methods are proposed to transfer samples (tuples or instances of the form $< s, a, r, s' >$) from the source task to the target task. Contrary to previous approaches, in this paper, we are interested in transferring information about the transition function. In particular, we propose a *batch learning* method which transfers information from the GP hyper-parameters of the state transition function to represent prior distributions of functions over the state transition function of the target task. We will show that by providing a family of functions as prior information about the underlying state transition function, significant reductions can be obtained in the convergence of the algorithm. Our proposal gradually incorporates the information from the target task producing a more stable process and faster convergence times. The proposed methodology also uses the source task policy function to initialize the policy in the target task. This creates more informative initial traces in the target task and a further boost to the convergence of the algorithm. The main contribution of this paper is a relatively simple, yet very effective approach for transfer learning in continuous state and action spaces, based on two intuitive ideas: (i) Within similar domains, you can expect similar properties on the state transition functions. This is implemented with a gradual transit between the hyper-parameters of the source task to those of the target task. (ii) Without any prior knowledge, your best initial trial is obtained using the policy learned in the source task. We performed experiments on the inverted pendulum under different conditions and show a significant improvement in the learning process.

2 Background

RL problems are typically formalized as MDPs, defined by $\langle S, A, P, R \rangle$, where S is the set of states, A is the set of possible actions that the agent may execute, $P : S \times A \times S \to [0, 1]$ is the state transition probability function, describing the task dynamics, $R : S \times A \to \mathbb{R}$ is the reward function measuring the performance of the agent. A policy $\pi : S \to A$ is defined as a probability distribution over state action pairs. In the case of continuous tasks, S and A are continuous spaces and functions approximators have to be used to represent the functions P and π, in this work we use GPs.

A Gaussian Process is a generalization of the Gaussian probability distribution. Given a set of input vectors $\mathbf{x_i}$ arranged as a matrix $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]$ and a vector of training observations $\mathbf{y} = [y_1, \ldots, y_n]^\top$, Gaussian Process methods for regression problems assume that the observations are generated as $y_i = h(\mathbf{x_i}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Thus the objective is to infer a model of the function h that generates the data. Similar to a Gaussian distribution, which is fully specified by a mean vector and a covariance matrix, a GP is specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$, also called a *kernel.*

Given a GP model of the *latent function* $h \sim \mathcal{GP}(m, k)$, it is possible to predict function values for an arbitrary input $\mathbf{x}_*$.

The covariance function k commonly used is the squared exponential kernel:

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \alpha^2 \exp(-\frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')^\top \mathbf{\Lambda}^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}')) + \delta_{\tilde{x}\tilde{x}'}\sigma_\epsilon^2 \tag{1}$$

where $\tilde{\mathbf{x}} = [\mathbf{x}^\top \mathbf{u}^\top]^\top$, α^2 is the variance of the transition function f, $\Lambda = \text{diag}([\ell_1^2, \ldots, \ell_D^2])$, which depends on length-scales ℓ_i, and $\delta_{\tilde{x}\tilde{x}'}$ denotes the Kronecker delta.

The hyper-parameters α^2, ℓ, σ_ϵ^2 describe the shape of the functions in the prior distribution (e.g., smoothness, noise tolerance).

These hyper-parameters are often optimized by evidence maximization (see [15] for further details).

The unknown transition function P can be described as $\mathbf{x}_t = f\left(\mathbf{x}_{t-1}, \mathbf{a}_{t-1}\right)$, $f \sim \mathcal{GP}(m, k)$, where $\mathbf{x}_t \in S$ is the state of the agent at time t, with continuous-valued states $\mathbf{x} \in \mathbb{R}^D$ and actions $\mathbf{a} \in A$, $A = \mathbb{R}^F$. Following [2], the transition model f is distributed as a Gaussian Process with mean function m and covariance function k, with sample tuples of the form $(\mathbf{x}_{t-1}, \mathbf{a}_{t-1}) \in \mathbb{R}^{D+F}$ as inputs and corresponding $\Delta_t = \mathbf{x}_t - \mathbf{x}_{t-1} + \epsilon \in \mathbb{R}^D$, $\epsilon \sim \mathcal{N}\left(0, \Sigma_\epsilon\right)$, as training targets.

The objective in RL is to find a policy π: $\mathbb{R}^D \mapsto \mathbb{R}^F$ that minimizes the expected accumulative cost given as:

$$V^\pi(\mathbf{x}_0) = \sum_{t=0}^{T} \mathbb{E}\left[c(\mathbf{x}_t)\right], \mathbf{x}_0 \sim \mathcal{N}\left(\mu_0, \Sigma_0\right) \tag{2}$$

which is the sum of expected cost $c(\mathbf{x}_t)$ of a trace $(\mathbf{x}_0, \ldots, \mathbf{x}_T)$, T steps ahead, where π is a continuous function approximated by $\tilde{\pi}$, using some set of parameters ψ.

The state transition function can be learned as a GP, using available data, going from a prior distribution of transition functions to a posterior one [15]. The learned transition model can then be used to simulate the system and speculate about the long-term behavior without the need of interaction (batch learning). The policy is then optimized according to these simulations and then used to get more tuples (state, action, successor state).

3 Qualitative Transfer Learning

The problem that we study is one where the source and target tasks have the same state variables and are variants of the same task. For instance, the source task could be to learn how to drive a car while the target task could be to learn how to drive a small truck. We expect, that at least "qualitatively", the behavior of both task should be the same. Following these ideas, we transfer information from the hyper-parameters of the transition function of the source task to the target task. With the samples from the source task, we learn the state transition function using GPs with a squared exponential kernel k as defined in the previous section.

In GP learning however, when no expert knowledge is available about the function properties, kernel hyper-parameters are often adjusted taking data into account and optimizing the log *marginal likelihood* (see [15] for more detail). That is the case of PILCO [2], where hyper-parameters are adjusted each time new data is added. Hyper-parameters are learned given the tuples $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n]$ and their corresponding $\mathbf{y} = [\Delta_1, \ldots, \Delta_n]$ acquired during the interaction with the environment.

In our approach, we do not let the evidence maximization process to take control of the hyper-parameters values in the target task, instead, we adjust the hyper-parameters using a forgetting factor. Let $\theta = [\alpha^2, \ell, \sigma_\epsilon^2]^\top$ denote the vector of hyper-parameters. Let $\theta^{(s)}$ denote the hyper-parameters transferred from the source task, θ_i the hyper-parameters used in the kernel for the target task at episode i, $\theta_i^{(p)}$ the hyper-parameters learned by evidence maximization in target task at episode i. We calculate the values of the hyper-parameters in the target task as follows:

$$\theta_0 = \theta^{(s)} \tag{3}$$

$$\theta_i = \gamma\theta_{i-1} + (1-\gamma)\theta_i^{(p)}, i > 0 \tag{4}$$

where $\gamma \in [0,1]$ is the ratio in which previous episode hyper-parameters are being incorporated into the kernel function.

Interaction with the environment when it is completely unknown requires an exploration phase where actions are chosen randomly. However, it is reasonable to believe that more accurate action heuristics exist when one has already learned a policy in a related task. For this reason, we also transfer the learned policy function from the source task and use it only in the first interaction with the environment for the target task.

4 Experiments

In this section we show experimental results in the well known inverted-pendulum task, commonly used as benchmark to compare reinforcement learning algorithms. We compare the performance of the proposed learning approach, QTL-PILCO, against PILCO [2] under different conditions.

Algorithm 1. Qualitative Transfer Learning

```
Require: $\theta^{(s)}$, $\psi^{(s)}$
$\tilde{\pi} \leftarrow \pi(\psi^{(s)})$
$\theta \leftarrow \theta^{(s)}$
Interact with environment, apply $\tilde{\pi}$ to obtain tuples.
repeat
    Infer transition function distribution $f$ from tuples and hyper-parameters $\theta$.
    repeat
        Evaluate policy $\tilde{\pi}$ over $f$. Get $V^{\tilde{\pi}}$
        Improve $\tilde{\pi}$                    ▷ Updating parameters $\psi$
    until convergence
    $\tilde{\pi} \leftarrow \pi(\psi)$
    Interact with environment, apply $\tilde{\pi}$ to obtain more tuples.
    Learn $\theta^{(p)}$ from all tuples.
    $\theta \leftarrow \gamma\theta + (1-\gamma)\theta^{(p)}$
until task learned
```

In the experiments, an inverted pendulum has to be swung up and then balanced. The pendulum is attached to a cart that moves along one axis when an external force is applied (action). The inverted pendulum problem involves applying actions that temporarily move the pendulum away from the target state, and the agent has to apply two different control criteria, one to swing the pendulum up and the other to balance it, thus it is non trivial to solve.

In the continuous scenario, a state $\mathbf{x}$ is represented by the position x of the cart, its velocity $\dot{x}$, the angle θ of the pendulum, and its angular velocity $\dot{\theta}$. The cost function is expressed as $c(\mathbf{x}) = 1 - \mathbf{exp}(-\frac{1}{2}a \times d^2)$, where a is a scale constant of the cost function (set to 0.25 in the experiments) and d is the Euclidean distance between the current and desired states, expressed as $d\left(\mathbf{x}, \mathbf{x}_{target}\right)^2 = x^2 + 2xl\sin\theta + 2l^2 + 2l^2\cos\theta$. In the current setup, the reward remains close to zero if the distance of the pendulum tip to the target is greater than $l = 0.6m$.

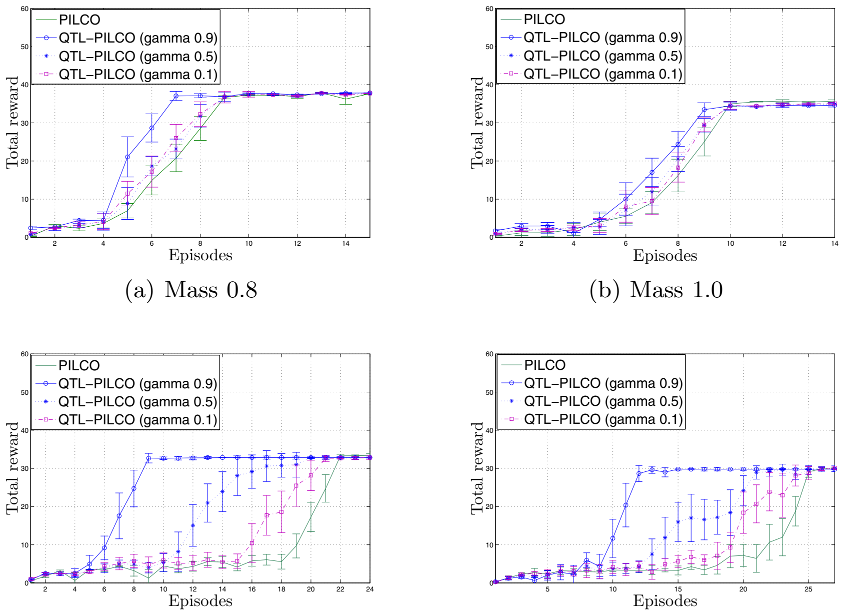
The source task consists of swinging a pendulum of mass 0.5 Kg. while in the target tasks the pendulums weights are changed to 0.8, 1.0, 1.5, and 2.0 Kg., respectively. Even when tasks have the same state and action spaces, their dynamics vary significantly and transferring the learned policy from the source task does not improve over learning from scratch and may even lead to negative transfer.

In our experiments, the source task was learned using PILCO. From that learning process, we transferred the hyper-parameters of the transition function and used the policy function for the first trial of the target task.

We repeated the procedure 5 times, randomly drawing the initial state $\mathbf{x} \sim \mathcal{N}(\mu_{s0}, \Sigma_0)$, the learning curves were averaged and plotted with their corresponding standard deviation. For PILCO, the Kernel hyper-parameters in the source task were initialized with heuristic values, as proposed in [2]. The initial training set for the transition function was generated by applying actions drawn uniformly from $[-\mathbf{a}_{\max}, \mathbf{a}_{\max}]$. For policy transfer, the whole policy learned in the source task was used as initial policy in the target task instead of a random policy to obtain initial samples, followed by QTL-PILCO (see Algorithm 1) to refine the policy.

In our proposed methodology, 8 hyper-parameters for each of the kernels $\mathbf{K}_i$, are taken from the source task, so 32 free variables are considered (considering the four variables for this domain). Those hyper-parameters are used as initial ones in the target tasks, and after the first episode, they are updated via evidence maximization from the samples and a weighted history of the original values, as described in Eq. 4. We performed experiments with different values for γ, from $\gamma = 0$, which is equivalent to learning with PILCO, to $\gamma = 0.9$ which provides more "inertia" to the hyper-parameters found in the source task.

A comparison of the learning curves for target tasks is showed in Figure 1, where we plot PILCO and QTL-PILCO with different values of γ. The horizontal axis shows the number of episodes (interactions with the environment) while the vertical axis shows the total reward, which is computed as the cumulative count of $1 - c(\mathbf{x})$ at every step.



(a) Mass 0.8

(b) Mass 1.0

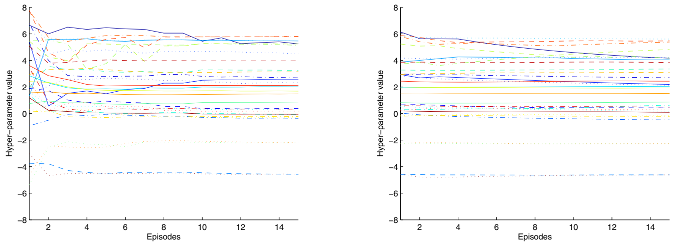
(c) Mass 1.5

(d) Mass 2.0

Fig. 1. Learning curves for target tasks 0.8Kg, 1.0Kg, 1.5Kg and 2.0Kg learned from 0.5Kg source task. Error bars represent ±1 standard deviation.

As can be appreciated from the figures, the proposed transfer learning approach can significantly reduce the learning process. When the target task is quite similar to the source task (in this case, with a similar mass), QTL-PILCO shows a clear improvement over learning without transfer. When the target task is less similar (larger mass) the improvement is much more noticeable.

The values of the hyper-parameters learned by evidence maximization can change drastically during the first iterations of the learning process due to poor samples. This is illustrated in the top graph of Figure 2(a). On the other hand, it can be seen in the lower graph of this figure, that with QTL-PILCO the values of the hyper-parameters are more stable and help to learn faster an adequate policy.



(a) PILCO

(b) QTL-PILCO

Fig. 2. Hyper-parameters convergence for the 0.8 Kg. task. PILCO oscillates more while QTL-PILCO leads to steadier values.

Our weighting technique provides a more suitable way to approximate target's hyper-parameters, provided that their values are expected to be not too different from those of the source task. This significantly reduces the expected oscillations that their values take with the initial trials and focuses the learning process in finding a policy with good priors on the expected values of the hyper-parameters of the transition function.

5 Conclusions

In this paper we have presented a transfer learning approach for reinforcement learning with continuous state and action spaces. The proposed approach is simple, yet very effective for transferring knowledge between related tasks. It is based on two general ideas for transfer learning. The first one is based on the idea that if you are going to transfer knowledge between similar tasks you can expect them to have similar general behaviors. We implemented this idea by starting with the hyper-parameters learned in the source task and gradually incorporating information from the learned hyper-parameters of the target task. The second idea is based on starting the new task with your "best guess". In this case, we used as starting policy the policy learned in the source task.

As future work, we would like to know the limits of our approach as the source and target tasks become less similar. We would also like to explore how to transfer knowledge from several source tasks. Finally, we would like to perform experiments in other more challenging domains.

Acknowledgements. Work supported by CONACyT grant No. 51415.

References

1. Deisenroth, M.P., Peters, J., Rasmussen, C.E.: Approximate dynamic programming with Gaussian processes. In: American Control Conference, pp. 4480–4485 (2008)
2. Deisenroth, M.P., Rasmussen, C.E.: PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In: ICML 2011, pp. 465–472 (2011)
3. Deisenroth, M.P., Rasmussen, C.E., Fox, D.: Learning to control a low-cost manipulator using data-efficient reinforcement learning. In: Proceedings of Robotics: Science and Systems, Los Angeles, CA, USA (2011)
4. Deisenroth, M.P., Rasmussen, C.E., Peters, J.: Model-based reinforcement learning with continuous states and actions. In: 16th European Symposium on Artificial Neural Networks, pp. 19–24 (April 2008)
5. Engel, Y., Mannor, S., Meir, R.: Bayes meets Bellman: The Gaussian process approach to temporal difference learning. ICML 20(1), 154 (2003)
6. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with Gaussian processes. In: ICML 2005, pp. 201–208 (2005)
7. Hasselt, H.V.: Insights in Reinforcement Learning Formal analysis and empirical evaluation of temporal-difference learning algorithms (2011)
8. Hasselt, H.V.: Reinforcement Learning in Continuous State and Action Spaces. In: Reinforcement Learning: State of the Art (2011)
9. Lazaric, A., Restelli, M., Bonarini, A.: Reinforcement learning in continuous action spaces through sequential monte carlo methods. In: Advances in Neural Information Processing Systems (2007)
10. Lazaric, A., Restelli, M., Bonarini, A.: Transfer of samples in batch reinforcement learning. In: Proceedings of the 25th International Conference on Machine Learning, ICML 2008, pp. 544–551 (2008)
11. Martín, J.A., de Lope, H.J., Maravall, D.: Robust high performance reinforcement learning through weighted k-nearest neighbors. Neurocomputing 74(8), 1251–1259 (2011)
12. Murray-Smith, R., Sbarbaro, D.: Nonlinear adaptive control using non-parametric Gaussian process prior models. In: 15TH IFAC, pp. 21–26 (July 2002)
13. Rasmussen, C.E., Deisenroth, M.P.: Probabilistic inference for fast learning in control. In: Girgin, S., Loth, M., Munos, R., Preux, P., Ryabko, D. (eds.) EWRL 2008. LNCS (LNAI), vol. 5323, pp. 229–242. Springer, Heidelberg (2008)
14. Rasmussen, C.E., Kuss, M.: Gaussian Processes in Reinforcement Learning. Advances in Neural Information Processing Systems 16, 16 (2004)
15. Rasmussen, C.E., Williams, C.: Gaussian Processes for Machine Learning. International Journal of Neural Systems 14(2), 69–106 (2006)
16. Sutton, R., Barto, A.G.: Introduction to Reinforcement Learning. MIT Press (1998)
17. Taylor, M.E., Jong, N.K., Stone, P.: Transferring Instances for Model-Based Reinforcement Learning. Machine Learning (September 2008)
18. Taylor, M.E., Stone, P.: Transfer Learning for Reinforcement Learning Domains: A Survey. Journal of Machine Learning Research 10, 1633–1685 (2009)