Active exploration in parameterized reinforcement learning

Mehdi Khamassi   MEHDI.KHAMASSI@UPMC.FR
Institute of Intelligent Systems and Robotics
Sorbonne Universités, UPMC Univ Paris 06, CNRS
F-75005 Paris, France

Costas Tzafestas   KTZAF@CS.NTUA.GR
School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece

Abstract

Online model-free reinforcement learning (RL) methods with continuous actions are playing a prominent role when dealing with real-world applications such as Robotics. However, when confronted to non-stationary environments, these methods crucially rely on an exploration-exploitation trade-off which is rarely dynamically and automatically adjusted to changes in the environment. Here we propose an active exploration algorithm for RL in structured (parameterized) continuous action space. This framework deals with a set of discrete actions, each of which is parameterized with continuous variables. Discrete exploration is controlled through a Boltzmann softmax function with an inverse temperature β parameter. In parallel, a Gaussian exploration is applied to the continuous action parameters. We apply a meta-learning algorithm based on the comparison between variations of short-term and long-term reward running averages to simultaneously tune β and the width of the Gaussian distribution from which continuous action parameters are drawn. When applied to a simple virtual human-robot interaction task, we show that this algorithm outperforms continuous parameterized RL both without active exploration and with active exploration based on uncertainty variations measured by a Kalman-Q-learning algorithm.

Keywords: Reinforcement Learning, Exploration/Exploitation, Multi-Armed Bandits, Meta-Learning, Active Exploration, Parameterized/Structured Reinforcement Learning.

1. Introduction

Important progresses have been made in recent years in reinforcement learning (RL) with continuous action spaces, permitting successful real-world applications such as Robotics applications (Kober and Peters, 2011; Stulp and Sigaud, 2013). Nevertheless, a recent review on reinforcement learning applied to Robotics (Kober et al., 2013) highlighted, among other points, that (i) a variety of algorithms have been developed, each being appropriate to specific tasks: model-based versus model-free, function approximation versus policy search, continuous versus discrete action spaces; (ii) important human knowledge is injected concerning the search in the parameter space, either by reducing it through learning from demonstration, or by pre-adjusting parameters such as the exploration rate based on the prior determination of the total number of episodes in the experiment. In particular, the balance between exploration and exploitation is often pre-determined with human prior knowledge and does not extend well to tasks with non-stationary reward functions.

To address the first issue, the recent proposal of RL algorithms in structured Parameterized Action Space Markov Decision Processes (PAMDP) (Masson and Konidaris, 2016; Hausknecht and Stone, 2016) seems to open a promising line of research. It combines a

set of discrete actions $A_d = \{a_1, a_2, ..., a_k\}$, each action $a \in A_d$ featuring m_a continuous parameters $\{\theta_1^a, ..., \theta_{m_a}^a\} \in \mathbb{R}^{m_a}$. Actions are thus represented by tuples $(a, \theta_1^a, ..., \theta_{m_a}^a)$ and the overall action space is defined as $A = \cup_{a \in A_d}(a, \theta_1^a, ..., \theta_{m_a}^a)$. This framework has been successfully applied to simulations of a Robocup 2D soccer task where agents have to learn to timely select between discrete actions such as running, turning or kicking the ball, and should learn at the same time with which speed to run, which angle to turn or which strength to kick. To ensure algorithm convergence, Masson and Konidaris (2016) alternate between learning phases: (i) given a fixed policy for parameter selection, they use Q-Learning to optimize the policy discrete action selection; (ii) Next, they fix the policy for discrete action selection and use a policy search method to optimize the parameter selection. In contrast, Hausknecht and Stone (2016) learn both in parallel by employing a parameterized actor that learns both discrete actions and parameters, and a parameterized critic that learns only the action-value function. Instead of relying on an external policy search procedure, they are thus able to directly query the critic for gradients.

Nevertheless, the exploration-exploitation trade-off is fixed in these methods, thus falling into the second issue raised by Kober et al. (2013)'s review. Exploration in continuous action spaces being different from discrete spaces, Hausknecht and Stone (2016) adapt ϵ-greedy exploration to parameterized action space by picking a random discrete action $a \in A_d$ with probability ϵ and sampling the action's parameters θ_i^a from a uniform random distribution. ϵ is arbitrarily annealed from 1.0 to 0.1 over the first 10,000 updates, thus requiring human prior knowledge about the duration of the task to appropriately tune exploration.

Here we use the Gaussian exploration for continuous action parameters proposed by van Hasselt and Wiering (2007), which in the original formulation uses a fixed Gaussian width σ. We then apply a noiseless version of the meta-learning algorithm of Schweighofer and Doya (2003), which tracks online variations of the agent's performance measured by short-term and long-term reward running averages. At each timestep, we use the difference between the two averages to simultaneously tune the inverse temperature β_t used for selecting between discrete actions a_j, and the width σ_t of the Gaussian distribution from which each continuous action parameter θ_i^a is sampled around its current value. We test our algorithm on a simple simulated human-robot interaction, where the algorithm tries to maximise reward computed as the virtual engagement of the human in the task, this engagement representing the attention that the human pays to the robot. We show that the proposed algorithm outperforms both continuous parameterized RL without active exploration and with active exploration based on uncertainty variations measured by a Kalman-RL algorithm (Geist and Pietquin, 2010).

2. Active exploration algorithm

The algorithm is summarized in Algorithm 1. It first employs Q-Learning (Watkins and Dayan, 1992) to learn the value of discrete action $a_t \in A_d$ selected at timestep t in state s_t:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha_Q[r_t + \gamma \max_a(Q_t(s_{t+1}, a)) - Q_t(s_t, a_t)] \qquad (1)$$

where α_Q is a learning rate and γ is a discount factor. The probability of executing discrete action a_j at timestep t is given by a Boltzmann softmax equation:

$$P(a_j|s_t, \beta_t) = \frac{exp\left(\beta_t Q_t(s_t, a_j)\right)}{\sum_a exp\left(\beta_t Q_t(s_t, a)\right)} \tag{2}$$

where β_t is a dynamic inverse temperature meta-parameter which will be tuned through meta-learning (see below).

In parallel, continuous $\widetilde{\theta}_{i,t}^{a_j}$ parameters with which action a_j is executed at timestep t are selected from a Gaussian exploration function centered on the current values $\theta_{i,t}^{a_j}(s_t)$ in state s_t of the parameters of this action (van Hasselt and Wiering, 2007):

$$P(\widetilde{\theta}_{i,t}^{a_j}|s_t, a_j, \sigma_t) = \frac{1}{\sqrt{2\pi}\sigma_t}exp\left(-(\widetilde{\theta}_{i,t}^{a_j} - \theta_{i,t}^{a_j}(s_t))^2/(2\sigma_t^2)\right) \tag{3}$$

where the width σ_t of the Gaussian is a meta-parameter which will be tuned through meta-learning (see below) and action parameters $\theta_{i,t}^{a}(s_t)$ are learned with a continuous actor-critic algorithm (van Hasselt and Wiering, 2007). A reward prediction error is computed from the critic: $\delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$ and is used to update the parameter vectors ω_t^C and ω_t^A of the neural network function approximations in the critic and the actor:

$$\omega_{i,t+1}^C = \omega_{i,t}^C + \alpha_C \delta_t \frac{\delta V_t(s_t)}{\delta \omega_{i,t}^C} \text{ and } \omega_{i,t+1}^A = \omega_{i,t}^A + \alpha_A \delta_t(\widetilde{\theta}_{i,t}^{a} - \theta_{i,t}^{a}(s_t))\frac{\delta \theta_{i,t}^{a}(s_t)}{\delta \omega_{i,t}^A} \tag{4}$$

where α_C and α_A are learning rates. In contrast to the original version where ω_t^A updates are performed only when $\delta_t > 0$ (van Hasselt and Wiering, 2007), here we update them all the time and proportionally to δ_t as in (Caluwaerts et al., 2012).

Finally, in order to perform active exploration on β_t and σ_t, we implement a noiseless version of the meta-learning algorithm proposed by (Schweighofer and Doya, 2003). We compute short- and long-term reward running averages:

$$\Delta \bar{r}(t) = (r(t) - \bar{r}(t))/\tau_1 \text{ and } \Delta \bar{\bar{r}}(t) = (\bar{r}(t) - \bar{\bar{r}}(t))/\tau_2 \tag{5}$$

where τ_1 and τ_2 are two time constants. We then update β_t and σ_t with:

$$\beta_t = F(\mu(\bar{r}(t) - \bar{\bar{r}}(t))) \text{ and } \sigma_t = G(\mu(\bar{r}(t) - \bar{\bar{r}}(t))) \tag{6}$$

where μ is a learning rate, $F(x) > 0$ is affine, and $0 < G(x) < 20$ is a sigmoid.

We also compared this meta-learning algorithm with the Kalman Q-Learning proposed by (Geist and Pietquin, 2010). However, in contrast to the original formulation which proposes a purely exploratory agent by replacing Q-values in Equation 2 by the action-specific diagonal terms of the covariance matrix – these terms representing the current variance/uncertainty about an action's Q-value –, here we multiply these terms by a weight η and add them as exploration bonuses b_t^a to Q-values in Equation 2. We also use the covariance terms to tune action-specific σ_t^a with function $G(x)$.

Algorithm 1 Active exploration with meta-learning

1: Initialize $\omega_{i,0}^A$, $\omega_{i,0}^C$, $Q_{i,0}$, β_0 and σ_0
2: **for** $t = 0, 1, 2, ...$ **do**
3:      Select discrete action a_t with $softmax(s_t, \beta_t)$ (Eq. 2)
4:      Select action parameters $\widetilde{\theta}_{i,t}^a$ with $GaussianExploration(s_t, a_t, \theta_{i,t}^a, \sigma_t)$ (Eq. 3)
5:      Observe new state and reward $\{s_{t+1}, r_{t+1}\} \leftarrow Transition(s_t, a_t, \widetilde{\theta}_{i,t}^a)$
6:      Update $Q_{t+1}(s_t, a_t)$ in the discrete Q-Learning (Eq. 1)
7:      Update function approx. $\omega_{i,t+1}^C$ and $\omega_{i,t+1}^A$ in continuous actor-critic (Eq. 4)
8:      **if** meta-learning **then**
9:          Update reward running averages $\bar{r}(t)$ and $\bar{\bar{r}}(t)$ (Eq. 5)
10:        Update β_t and σ_t (Eq. 6)
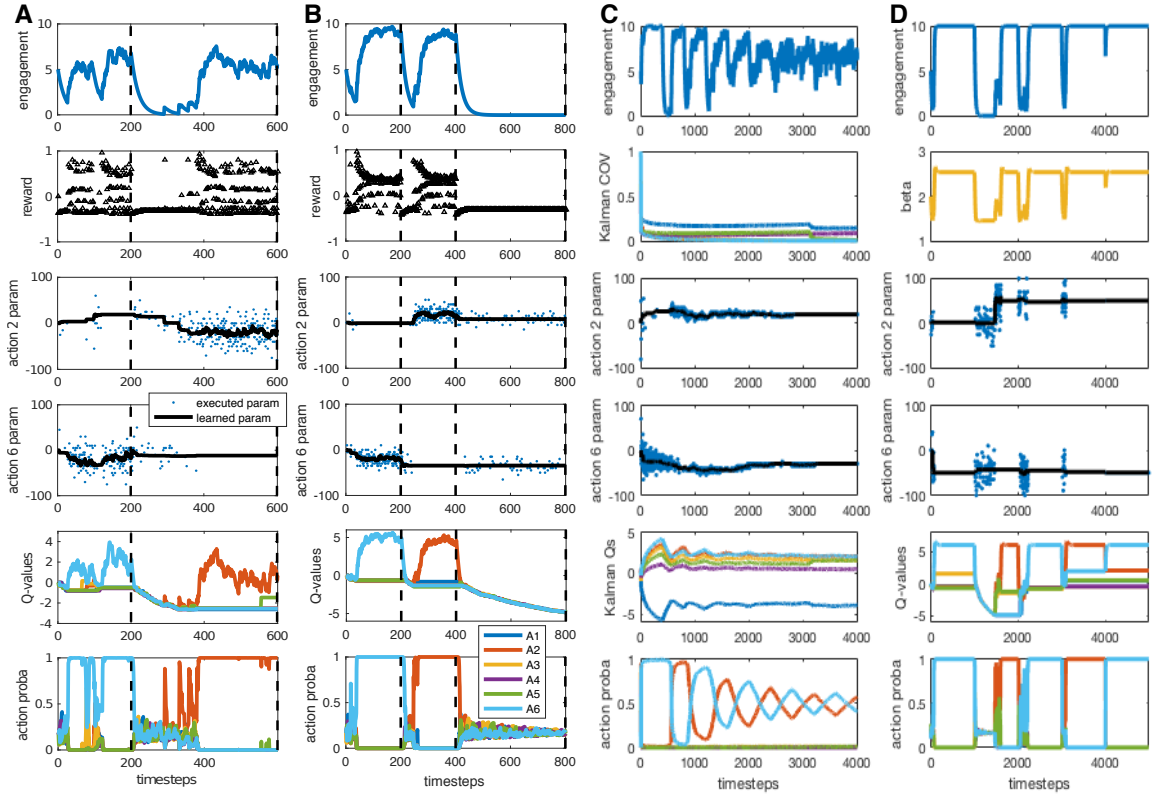11:   **end if**
12: **end for**

Figure 1: Simulations of the parameterized reinforcement learning (RL) algorithm with (A) fixed $\sigma = 20$ and $\beta = 4$ (no active exploration), (B) fixed $\sigma = 10$ and $\beta = 4$ (no active exploration), (C) σ_t^a and b_t^a tuned by Kalman-RL (active exploration) or (D) σ_t and β_t tuned by meta-learning (active exploration).

3. Experiments

We test the algorithm described in Section 2 in a simple simulated human-robot interaction involving a single state, 6 discrete actions, and continuous action parameters between -100 and 100. The task is similar to a non-stationary stochastic multi-armed bandit task except that rather than associating a fixed probability of reward to each discrete action, an action will yield reward only when its continuous parameters are chosen within a Gaussian distribution around the current optimal action parameter $\mu^\star$ with variance $\sigma^\star$ (which are unknown to the robot). Every n timesteps, $\mu^\star$ changes so that the task is non-stationary and requires constant re-exploration and learning by the robot.

The reward is given by a dynamical system which is based on the virtual engagement $e(t)$ of the human in the task. This engagement is supposed to represent the attention that the human pays to the robot. It starts at 5, increases up to a maximum $e_{max} = 10$ when the robot performs the appropriate actions with the appropriate parameters, and decreases down to a minimum $e_{min} = 0$ otherwise:

$$e(t+1) = \begin{cases} e(t) + \eta_1(e_{max} - e(t))H(\theta_t^a), & \text{if } a(t) = a^\star \ \& \ H(\theta_t^a) \geq 0 \\ e(t) - \eta_2(e_{min} - e(t))H(\theta_t^a), & \text{if } a(t) = a^\star \ \& \ H(\theta_t^a) < 0 \\ e(t) + \eta_2(e_{min} - e(t)), & \text{otherwise} \end{cases}$$

where $\eta_1 = 0.1$ is the increasing rate, $\eta_2 = 0.05$ is the decreasing rate, and $H(x)$ is the reengagement function given by $H(x) = 2\left(exp\left(-\frac{(x-\mu^\star)^2}{2\sigma^{\star 2}}\right) - 0.5\right)$ where $a^\star$, $\mu^\star$ and $\sigma^\star$ are respectively the optimal action, action parameter and variance around $a^\star$.

The reward function is then computed as $r(t+1) = (1-\lambda)e(t+1) + \lambda\Delta e(t+1)$ where $\lambda = 0.7$ is a weight. This reward function ensures that the algorithm gets rewarded in cases where the engagement $e(t+1)$ is low but nevertheless has just been increased by the action tuple $(a(t), \theta^a(t))$ performed by the robot.

We first simulated the algorithm without active exploration (thus with a fixed $\sigma = 20$) in a task where the optimal action tuple $(a^\star, \mu^\star)$ is $(a_6, -20)$ during 200 timesteps ($\sigma^\star = 10$ in all the experiments presented here), then switches to $(a_2, -20)$ until timestep 600. Figure 1A shows that the algorithm first learns the appropriate action tuple $(a_6, -20)$, then takes some time to learn the second tuple, making the engagement drop between timesteps 200 and 400 and eventually finds the second optimal tuple. Nevertheless, $\sigma = 20$ makes the robot select action parameters $\widetilde{\theta}_t^a$ with a large variance (illustrated by the clouds of blue dots around the learned action parameters θ_t^2 and θ_t^6 plotted as black curves). As a consequence, the engagement is not optimized and always remains below 7.5. In contrast, the same algorithm with a smaller fixed variance $\sigma = 10$ can make the engagement reach the optimum of 10 when the optimal action tuple is learned (Figure 1B before timestep 400), but results in too little exploration which prevents the robot from finding a new action parameter which is too far away from the previously learned one (after timestep 400, the new optimal action tuple is $(a_6, 20)$). These two examples illustrate the need to actively vary the variance σ_t as a function of changes in the robot's performance.

We next tested active exploration with the Kalman Q-Learning algorithm in a task alternating between optimal tuples $(a_2, -20)$ and $(a_6, 20)$ every 400 timesteps (Figure 1C). The diagonal terms of the covariance matrix COV in the Kalman filter nearly monotonically
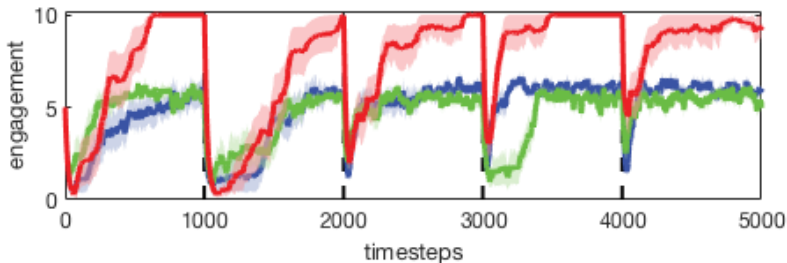
Figure 2: Comparison of engagement in 10 simulations of the meta-learning model (red), the model without active exploration (blue), and the Kalman-QL (green).

decrease, resulting in a large variance σ_t when action a_6 is executed until about timestep 600, and progressively decreasing the variance until the end of the experiment. Nevertheless, the algorithm quickly finds the appropriate action parameters and rapidly shifts between actions a_2 and a_6 after each change in the task condition. In the long run, the model progressively averages the statistics of the two conditions and learns to perform both actions with 50/50 probabilities (bottom part of Figure 1C) which decreases the simulated engagement (top).

We then tested active exploration with the meta-learning algorithm in a slightly more difficult task where the optimal action tuple alternate between $(a_2, -50)$ and $(a_6, 50)$ every 1000 timesteps (Figure 1D). Transient drops in the engagement result in transient decreases in the exploration parameter β_t as well as transient increases in the variance σ_t. This enables the algorithm to go through quick transient but wide exploration phases and to rapidly reconverge to exploitation, thus maximizing the simulated engagement.

Finally, we performed 10 simulations of each model on the difficult version of the task and plotted the average and standard deviation of the simulated engagement (Figure 2). The blue curve shows the performance of the algorithm without active exploration (i.e. fixed $\sigma = 19$ obtained through parameter optimization), which adapts to each new condition but never exceeds a plateau of about 6. The green curve shows the active exploration with Kalman, which adapts faster at the beginning but progressively decreases its maximal engagement. The red curve shows the active exploration with meta-learning which initially takes more time to adapt but then only performs short transient explorations and reaches the optimum engagement of 10.

4. Conclusion

In this work, we have shown that a meta-learning algorithm based on online variations of reward running averages can be used to adaptively tune two exploration parameters simultaneously used to select between both discrete actions and continuous action parameters in a parameterized action space. While we had previously successfully used the Kalman Q-Learning proposed by Geist and Pietquin (2010) to coordinate model-based and model-free reinforcement learning in a stationary task (Viejo et al., 2015), it was not appropriate for the current non-stationary task. The proposed active exploration scheme could be a promising solution for Robotics applications of parameterized reinforcement learning.

Acknowledgments

References

K. Caluwaerts, M. Staffa, N'Guyen. S., C. Grand, L. Dollé, A. Favre-Félix, B. Girard, and M. Khamassi. A biologically inspired meta-control navigation system for the psikharpax rat robot. *Bioinspiration and Biomimetics*, 7(2):025009, 2012.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of artificial intelligence research*, 39:483–532, 2010.

M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. In *International Conference on Learning Representations (ICLR 2016)*. 2016.

J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84:171–203, 2011.

J. Kober, J.A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, pages 1238–1274, 2013. doi: 10.1177/0278364913495721.

W. Masson and G. Konidaris. Reinforcement learning with parameterized actions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. 2016.

N. Schweighofer and K. Doya. Meta-learning in reinforcement learning. *Neural Networks*, 16(1):5–9, 2003.

F. Stulp and O. Sigaud. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn Journal of Behavioral Robotics*, 4(1):49–61, 2013.

H. van Hasselt and M. Wiering. Reinforcement learning in continuous action spaces. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279. 2007.

G. Viejo, M. Khamassi, A. Brovelli, and B. Girard. Modeling choice and reaction time during arbitrary visuomotor learning through the coordination of adaptive working memory and reinforcement learning. *Frontiers in behavioral neuroscience*, 9, 2015.

C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.