Learnings Options End-to-End for Continuous Action Tasks

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, Doina Precup
Reasoning and Learning Lab,
McGill University
{mklissa,pbacon,jharb,dprecup}@cs.mcgill.ca

Abstract

We present new results on learning temporally extended actions for continuous tasks, using the options framework (Sutton *et al.* [1999b], Precup [2000]). In order to achieve this goal we work with the option-critic architecture (Bacon *et al.* [2017]) using a deliberation cost and train it with proximal policy optimization (Schulman *et al.* [2017]) instead of vanilla policy gradient. Results on Mujoco domains are promising, but lead to interesting questions about *when* a given option should be used, an issue directly connected to the use of initiation sets.

1 Introduction

The options framework (Sutton *et al.* [1999b], Precup [2000]) allows a reinforcement learning agent to represent, learn and plan with temporally extended actions. These temporally extended actions consist of a set of internal policies, termination conditions and sometimes initiation sets that allow controlling the number of choices available to an agent. Given a set of options, the agent will learn a policy over options, which is typically viewed as executing in a call-and-return fashion: once this policy chooses an option, the option will execute until it terminates, then the policy over options will make a new choice. Learning options is beneficial as it leads to specialization in the state space, and therefore to potentially reduced complexity in terms of the internal policies of the options. The option-critic architecture (Bacon *et al.* [2017]) provides an agent with an end-to-end algorithm to learn options in order to maximize the expected discounted return, by relying on ideas akin to actor-critic methods. In this work, we exploit the option-critic architecture by combining it to a recent algorithm, Proximal Policy Optimization (PPO) (Schulman *et al.* [2017]), which is very well suited for continuous control tasks and has shown better sample complexity in empirical comparisons. We present results of our approach on a set of environments from the Mujoco framework; our results are consistent with published evaluations which show that learning options provides increased performance, better interpretability and faster learning.

2 Background

A Markov Decision Process $\mathcal{M}$ is a tuple $\doteq(\mathcal{S}, \mathcal{A}, \gamma, r, P)$ with $\mathcal{S}$ the state set, $\mathcal{A}$ the action set and the scalar $\gamma \in [0, 1)$ the discount factor. The reward function maps states and actions to a scalar reward $r : \mathcal{S} \times \mathcal{A} \to Dist(\mathbb{R})$ and the transition matrix $P : \mathcal{S} \times \mathcal{A} \to Dist(\mathcal{S})$ specifies the environment's dynamics. A policy π is a set of probability distributions over actions conditioned on states π: $\mathcal{S} \to \mathcal{A}$. For a given policy, the value function $V_\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{t=0} \gamma^t r(S_t, A_t) | S_0 = s\right]$ defines the expected return obtained by following π. V_π satisfies the Bellman equations : $V_\pi(s) = \sum_a \pi(a|s) \left(r(s, a) + \gamma \sum_{s'} P(s'|s, a) V_\pi(s')\right)$.

The policy gradient theorem (Sutton *et al.* [1999a]) provides the gradient of a parametrized stochastic policy π_θ with respect to the expected discounted return from an initial state distribution $d_0 \in \text{dist}(\mathcal{S})$.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

For simplicity, we write the policy as π, making its parametrization (θ) implicit.

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_s d(s;\theta) \sum_a \frac{\partial \pi\left(a|s\right)}{\partial \theta} Q_\pi(s,a)$$

where $d(s;\theta) = \sum_{s_0} d(s_0) \sum_{t=0}^{\infty} \gamma^t P_\pi(S_t = s|S_0 = s_0)$ is a weighting of states along the trajectories generated by π and passing through s. Using the log-likelihood trick (Williams [1992]),

$$\frac{\partial L(\theta)}{\partial \theta} = \mathbb{E}\left[\frac{\partial \log \pi\left(A_t|S_t\right)}{\partial \theta} A^\pi(S_t, A_t)\right]$$

where $A^\pi(S_t, A_t) = Q_\pi(S_t, A_t) - V_\pi(S_t)$ is the advantage function. The term $V_\pi(s_t)$ acts as a *baseline* (Williams [1992]; Sutton *et al.* [1999a]) which reduces the variance of the resulting estimator.

2.1 Trust region methods and Proximal Policy Optimization (PPO)

Trust region methods, and in particular the TRPO algorithm (Schulman *et al.* [2015a]), are second-order methods that maximize a surrogate objective subject to a constraint. TRPO has proven useful for continuous control, but it can be computationally expensive and doesn't allow for parameter sharing.

Proximal Policy Optimiation (PPO) achieves the same level of reliability and performance as TRPO while being a first-order method. To do so, it uses an objective with clipped probability ratios, preventing an excessive shift in the probability distribution between updates. This clipping also allows for multiple epochs of minibatch updates on a single sampled trajectory. The clipped surrogate objective is:

$$\frac{\partial L(\theta)^{\text{PPO}}}{\partial \theta} = \mathbb{E}\left[\frac{\partial}{\partial \theta} \min(\rho_t(\theta) A^\pi(S_t, A_t), \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon) A^\pi(S_t, A_t))\right]$$

where $\rho_t(\theta) = \frac{\pi(A_t|S_t)}{\pi_{old}(A_t|S_t)}$ is the importance sampling ratio. The authors use the Generalized Advantage Estimation (Schulman *et al.* [2015b]) to calculate the advantage function $A^\pi(S_t, A_t)$.

2.2 Option-Critic

The option-critic architecture (Bacon *et al.* [2017]) is a gradient-based approach for learning intra-option policies as well termination conditions, assuming that all options are available at every state. Moreover, the parameters of the intra-option policies (θ_π) and the termination function (θ_β) are assumed to be independent. The intra-option policy gradient is as follows:

$$\frac{\partial L(\theta)}{\partial \theta_\pi} = \mathbb{E}\left[\frac{\partial \log \pi\left(A_t|S_t, O_t\right)}{\partial \theta_\pi} Q_\pi(S_t, O_t, A_t)\right]$$

where a baseline (i.e. the above state-option value function Q_π parametrized by θ_w) is generally added. However, if the options are learned to optimize returns, in the long run, they will tend to disappear since any MDP can be solved optimally using primitive actions. To avoid this problem, Harb *et al.* [2018] use the bounded rationality framework (Simon [1969]) and introduce a deliberation cost (η), interpreted as a margin of how much better an option should be than the current option in order to replace it. The termination gradient then takes the following form:

$$\frac{\partial L(\theta)}{\partial \theta_\beta} = \mathbb{E}\left[-\frac{\partial \beta(S_t, O_t)}{\partial \theta_\beta} (A^{\pi\beta}(S_t, O_t) + \eta)\right]$$

where $A^{\pi\beta}(S_t, O_t) = Q_\pi(S_t, O_t) - V_\pi(S_t)$ is the termination advantage function and stems directly from the derivation of the gradient.

3 Algorithm

We introduce the Proximal Policy Option-Critic (PPOC) algorithm which, just like PPO, works in two stages. In the first stage, the agent collects trajectories of different options and computes the advantage functions using Monte-Carlo returns. We then proceed to the optimization stage where, for K optimizer iterations, we choose M tuples and apply the gradients. We also chose to use a stochastic policy over options, parameterized by an independent vector θ_μ (as opposed to ϵ-greedy) which we learned under the same policy gradient approach.

Algorithm 1: Proximal Policy Option Critic (PPOC)

for *iteration=1,2,....* **do**
  $c_t \leftarrow 0$
  $s_t \leftarrow s_0$
  Choose o_t with a softmax policy over options $\mu(o_t|s_t)$
  repeat
    Choose a_t according to $\pi(a_t|s_t)$
    Take action a_t in s_t, observe s_{t+1}, r_t
    $\hat{r}_t = r_t - c_t$
    if β *terminates in* s_{t+1} **then**
      choose new o_{t+1} according to softmax $\mu(o_{t+1}|s_{t+1})$
      $c_t = \eta$
    else
      $c_t = 0$
    end
  until *T timesteps*
  Compute the advantage estimates for each timestep;
  for o=o_1,o_2,.... **do**
    $\theta_{old} \leftarrow \theta$
    for *K optimizer iterations with minibatches M* **do**
      $\theta_\pi \leftarrow \theta_\pi + \alpha_{\theta_\pi} \frac{\partial L_t(\theta)^{\text{PPO}}}{\partial \theta_\pi}$ intra option policy
      $\theta_\beta \leftarrow \theta_\beta - \alpha_{\theta_\beta} \frac{\partial \beta(s_t)}{\partial \theta_\beta} (A(s_t, o_t) + \eta)$ termination
      $\theta_\mu \leftarrow \theta_\mu + \alpha_{\theta_\mu} \frac{\partial \log \mu(o_t|s_t)}{\partial \theta_\mu} A(s_t, o_t)$ option policy
      $\theta_w \leftarrow \theta_w - \alpha_{\theta_w} \frac{\partial (G_t - Q_\pi(s_t, o_t))^2}{\partial \theta_w}$ option-value
    end
  end
end

4 Experiments

We performed experiments on locomotion tasks available on OpenAI's Gym (Brockman *et al.* [2016]) using the Mujoco simulator (Todorov *et al.* [2012]). We aim to assess the following: (1) whether the use of options can increase the speed of learning as well as the final performance, (2) the interpretability of the resulting options.

In our experiments, we used as input the vectors defining joint angles, joint velocities, and coordinates of the center of mass. We used two separate networks with 64 hidden units per layer, each containing two layers.[1] For all the layers we used tanh non-linearity, except for the output which was linear for of value functions and intra-option policies, sigmoid for the termination probability and softmax for the policy over options. The first network was used to output the policy over options $\mu(o|s)$ and the intra-option policies $\pi(a|s)$, while the second network was used to output the value functions $Q_\pi(s, o)$ and the termination probabilities $\beta(s)$. The log-standard deviations were parameterized by a vector independent of the input state. We used the exact same hyper-parameters as mentioned in Schulman *et al.* [2017], except for the optimizer mini-batch size which was divided by the number of options. We proceeded so in order to avoid training more samples per iteration with the options framework, thus enabling a fair comparison between options and primitive actions. In the case of options, we also divide the reward by 10 to reduce the scale of the value functions, and therefore the termination probability gradient, making it more stable. We didn't proceed to any hyper-parameters search to improve the results. Our experiments exclusively investigate the merits of using two options and compare the results to the case of primitive actions (no options).

[1] The code, as well as the values for the hyperparameters, are available here: `https://github.com/mklissa/PPOC`

In addition to the classic Mujoco environments, we ran agents in an environment called HopperIceBlock-v0.[2] This environment contained a more explicit compositionality than the original Mujoco environments available on OpenAI's Gym, which simply require to learn a gait that maximizes speed in a direction. We used Hopper-v1 as a starting point and added some obstacles in the agent's path: solid slippery blocks. The agent had to learn to pass them either by jumping completely over them or by sliding on their surface.

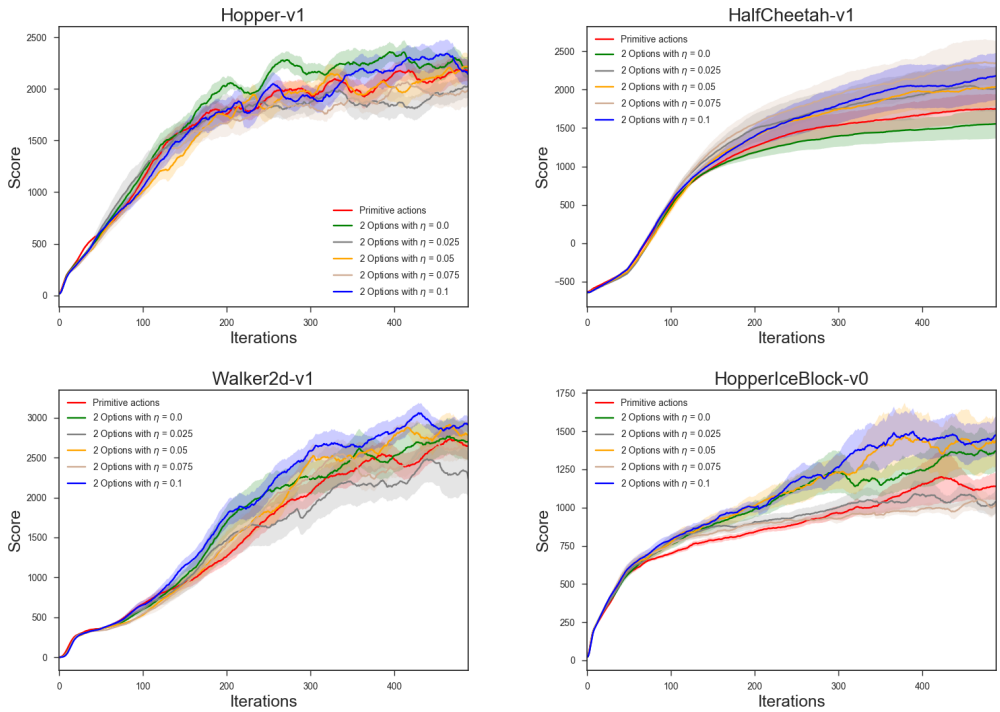

Figure 1: Results in Mujocco using 12 different random seeds for a total of 1 million steps (each iteration is 2000 steps)

The results are summarized in Fig.1. As expected, using options with a deliberation cost yields better results and faster learning on most environments. It is interesting to note that the increase in performance is not directly proportional to the value of η. This is due to the different scales of the average returns across environments, as well as during the course of learning. In the current formulation of the deliberation cost, its value is a hyperparameter that has to be set. It would be useful to explore the possibility of working with a learned value instead of a constant. This is left as future work.

The results that stand out the most are the one on the customized environment. More importantly, the success threshold for the environment is around 1200 points, under that level the agent actually doesn't learn to pass the iceblock and continue its gait. So, the agent using options is the only one solving this environment. This also led us to investigate how the options are used in this environment as opposed to the classic Mujoco environments.[3] In HopperIceBlock-v0, the interpretability of the options is obvious and greatly helps the performance: one option is used to hop when there is no iceblock nearby, but then when passing over the iceblock, both options are used to complete the specific task. In the case of the classic Mujoco environments, one option is used to gain momentum at the start of the episode and is never used thereafter. Even if the agents using options outperform the agents with primitive actions on classic environments, we can only truly see the benefits of a hierarchical framework when used in the appropriate environment.

[2]HopperIceBlock-v0 is based on Henderson *et al.* [2017] and is avaialable here: `https://github.com/mklissa/gym-extensions`

[3]Videos from the environments are available at `https://www.youtube.com/watch?v=XI_txkRnKjU`

5 Conclusion

Our experiments demonstrate that it is possible to learn options in an end-to-end manner using deep networks on continuous actions environments, and to the best of our knowledge this is the first work to do so. Our results also suggest that the increase in performance is not directly linked to the deliberation cost, which is problematic as it leaves us with the task of finding the right value. For the options framework to be truly end-to-end it would be necessary to learn a value of η. More importantly, we have seen that the increase in performance is related to the compositionality of the environment. In the classic Mujoco environments, using options is not as beneficial as using them in a customized environment with a more obvious division in the state-space. This leads to the following question: when should we be using options? This question also points to a fundamental problem in the current options framework: it is necessary for us to manually specify the number of available options. How should one decide on this number? As stated in Bacon *et al.* [2017], one way to answer this question would be to reintroduce the notion of initiation sets in the option-critic architecture.

References

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option : Learning options with a deliberation cost. In *AAAI*, pages 1726–1734, 2018.

P. Henderson, W.-D. Chang, F. Shkurti, J. Hansen, D. Meger, and G. Dudek. Benchmark environments for multitask learning in continuous domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*, 2017.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amhersts, 2000.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, first edition, 1969.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, 1999.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.