Asynchronous Advantage Option-Critic with Deliberation Cost

Jean Harb
McGill University
Montreal, Quebec, Canada
jharb@cs.mcgill.ca

Pierre-Luc Bacon
McGill University
Montreal, Quebec, Canada
pbacon@cs.mcgill.ca

Doina Precup
McGill University
Montreal, Quebec, Canada
dprecup@cs.mcgill.ca

Abstract

Learning temporally extended actions is a long-standing problem that has recently been tackled by different types of frameworks (Bacon *et al.* [2016], Vezhnevets *et al.* [2016], Kulkarni *et al.* [2016], Tessler *et al.* [2016]). The option-critic architecture (OC) is a framework that allows an agent to learn options in an end-to-end manner, while optimizing the expected return. In this work, we introduce an asynchronous variant of OC where it trains on multiple processes at once and accumulates the experience to train a single network, like in A3C (Mnih *et al.* [2016]).

Option-critic has the problem that options collapse to lasting only a single time-step. This problem arises from the fact that there's no reason for the options to be temporally extended. The agent sees termination only as giving it more choices, which cannot be worse than staying in the same option. In the worst case, it will simply choose to go back into the same option. In reality, there is a cost to terminating. Computation resources are limited, and when terminating, the agent must take time and computation to decide which option to execute. A deliberation cost would indicate that there's a cost to terminating an option. We introduce a deliberation cost which can be simply implemented into option-critic, which allows the agent to learn temporally extended options as it now sees termination associated with a negative reward.

We perform experiments on a few games of the Arcade Learning Environment (Atari 2600 games) and show the learning capacity of the asynchronous version of option-critic and the effects of different deliberation costs.

Keywords: Reinforcement Learning, Deep Learning

1 Introduction

Temporal abstraction is a fundamental aspect of human reasoning. It allows us to make long term predictions without being distracted by the noise that comes with short term predictions. For example, if one considers doing a Ph.D., it makes no sense to consider that as 5 years of muscle twitches, but instead to consider key and abstract points such as completing the comprehensive exam or even the general idea of learning a lot about a certain subject.

The options framework (Sutton *et al.* [1999], Precup [2000]) allows reinforcement learning agents to have temporally extended actions, where the agent can follow a certain policy for an amount of time before switching to a different policy later on. However, it proved difficult to learn options automatically.

Bacon *et al.* [2016] introduced the option-critic architecture (OC), which allows an agent to learn options in an end-to-end manner that maximize the expected return. They also introduced a DQN (Mnih *et al.* [2015]) variant of OC, in which they were capable of learning options in combination with neural networks. However, the architecture of the DQN algorithm was not compatible with OC. DQN is an off-policy algorithm which uses experience replay to train on past samples, whereas OC is an on-policy algorithm, which only trains on the last samples seen. Furthermore, training is done on sequential, and highly correlated states, which could destabilize the algorithm as it uses deep neural networks.

Mnih *et al.* [2016] introduced variant of actor-critic (AC) using deep networks for function approximation. This algorithm is also on-policy, using sequential states as training data. To stabilize the networks during training, they introduced an asynchronous version of AC (A3C) which uses parallel agents and pools the gradient updates from the different agents into a single network. This decorrelates the training data as the updates now come from a large variety of states. The properties of A3C are similar to those of OC, and as such, we introduce a variant of OC which is combined with A3C.

A deliberation cost is the idea that there is a cost to reasoning about what to do. When an option terminates, the agent must choose which option to select, and then return to executing an option. In the current option-critic architecture, there is no cost to terminating, which results in an agent which wants to terminate the option at every step and select a new option. However, if we have limited resources, we would want to limit the number of steps where options terminate. Bacon and Precup [2015] introduced the deliberation cost in the options framework, and in this work we derive a new variation of the option-critic architecture which incorporates the deliberation cost while learning the options. Recently, Léon and Denoyer [2016] also developed a framework to learn options while considering a cost for termination and extending the options.

2 Background

In this paper, we work with Markov Decision Processes (MDP), which are tuples $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ where we have a set of states $s \in \mathcal{S}$, a set of actions $a \in \mathcal{A}$, a reward function $r(s'|s,a)$ which outputs a possibly stochastic reward, a γ scalar which represents the discount factor and a set of transitions $p(s'|s,a) \in \mathcal{P}$ which represents the probability of transitioning from state s to s' when action a is taken.

A policy π is a behaviour function which takes a state s and action a, and outputs the probability of selecting the action.

The value of a state $V^\pi(s)$ is the expected sum of discounted rewards when following policy π, which can be given by $V^\pi(s) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r_t\big]$.

In option-critic, we have many policies (called options) as opposed to a single policy, and which we denote as $\pi_\omega(a|s)$, where ω represents the which option is selected. We also have a termination function $\beta_{\omega,\vartheta}(s)$, which outputs the probability that a option ω will terminate in state s. If the option terminates, then we select a new option to execute until the new one terminates. Finally, we have a policy over policies $\pi_\Omega(\omega|s)$ which outputs the probability of selecting an option whenever the previous one has terminated.

3 Asynchronous Advantage Option Critic (A2OC)

In A3C, every layer of the network up until the last is shared among the actor and the critic. Each has their own objective function, the critic being the squared TD and the actor being the policy gradient. In A2OC, we have a similar architecture, except instead of only having an actor and a critic, we have an actor, Q-value and termination function for each option. The Q-value is the same as the value function in A3C, except in our case we have a value for each policy, making it a Q-value instead. The termination function is a linear layer with a sigmoid activation function which represents the probability of terminating the current option in a given state.

We have two variants of the algorithm, where the policy over option can be either ϵ-greedy or an actor. If we use an actor, then we have a fourth component, a linear softmax layer, connected to the shared layer which is output from the convolutional neural network (CNN).

3.1 Deliberation Cost

As we know, the gradient of the termination function is the proportional to the advantage of the current option used. When the ϵ-greedy policy over options is used, the advantage function is less than or equal to 0. This is because we're comparing the value of the current option with the highest value of any option, as the policy is to select greedily. In this case, the termination function quickly learns to always output a near 100% of terminating, as it's always better to terminate and select a new option. In the worst case, the agent simply re-selects the same option. There's no reason to restrain itself to using the same option as the last step.

This however goes against our objective of having long-lasting options. In Bacon *et al.* [2016], a termination regularization was introduced, where a scalar was added to the advantage function. In essence, this is saying that as long as the option has a value within a certain range of the best option, it's fine to continue executing the current option.

Here, we introduce a deliberation cost, which is essentially a negative reward that we give to the agent whenever it terminates an option. To implement the deliberation cost into the OC algorithm, we need to perform 2 steps. First, whenever the option terminates during training, we subtract the deliberation cost from the reward, allowing the deliberation to be part of the Q-values of the options. Second, the termination gradient needs to be updated to incorporate the fact that terminating will lead to a negative reward. This can be done as follows.

We are maximizing the expected return of state s_1 but with the option ω_0 which was selected at the previous time-step.

$$U(\omega_0, s_1) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t-1} r_t \middle| \omega_0, s_1\right]$$

This can be rewritten as follows, with ϑ being the parameters of the termination function.

$$U(\omega, s') = (1 - \beta_{\omega,\vartheta}(s'))Q(s', \omega) + \beta_{\omega,\vartheta}(s')V_\Omega(s')$$

It's important to see that $U(\omega, s')$ is the value of s' before the decision of terminating has been made. As such, we can update this function to the following, where the deliberation cost c is received as a negative reward if the agent chooses to terminate.

$$U(\omega, s') = (1 - \beta_{\omega,\vartheta}(s'))Q(s', \omega) + \beta_{\omega,\vartheta}(s')(V_\Omega(s') - c)$$

We can then derive the gradient to get the following. See Bacon *et al.* [2016] for the exact derivation of the gradient.

$$\frac{\partial U(s_1, \omega_0)}{\partial \vartheta} = -\sum_{s',\omega} d_\Omega(s', \omega) \frac{\partial \beta_{\omega,\vartheta}(s')}{\partial \vartheta} (Q_\Omega(s', \omega) - V_\Omega(s') + c)$$

Surprisingly, this is the exact same function as the regularizer present in Bacon *et al.* [2016]. The only difference is that we must both change the termination gradient and also subtract the deliberation cost from the last reward, encoding its effect in Q and V. This small difference does have a significant effect however. It means that the termination function doesn't just consider whether the value of the current option is within a threshold of the best option, but also considers whether it expects to terminate in the future. Consider the case where the advantage is negative and very close in value to the deliberation cost. If the agent knows the advantage will be even worse in the next step and will have to terminate, it's not actually worth sticking with the current sub-optimal option for one more step and it will terminate.

In Bacon and Precup [2015], the cumulative discounted deliberation cost was a separate function $Q_c(s, \omega)$ as opposed to being merged into the $Q_\Omega(s, \omega)$. Using that point of view, we could see the termination regularizer as this separate $Q_c(s, \omega)$, but with a gamma of 0, whereas the deliberation cost would have a gamma equal to the one used with the reward.

As for the second variant of OC presented, where we use an actor as the policy over options, the need for a deliberation cost is less pressing. As opposed to being the maximum option value, the state value is now a weighting of the option values by the probability of each one being selected, as $V(s) = \sum_\omega \pi_\Omega(\omega|s)Q(s, \omega)$. Because of the stochasticity of the policy over options, if the option with the highest value is currently being executed, the advantage function will be positive. This will make the termination gradient lower the odds of terminating, stretching the option length.

4 Experiments and Results

The experiments were kept as consistent with the 1-day feed-forward A3C (Mnih *et al.* [2016]) as possible. That is, we use a three layer CNN which is shared among each function. We used a learning rate of 0.0007 which decays to 0 linearly through training. RMSprop parameters are shared across the processes, with the epsilon and rho hyper-parameters set to 0.01 and 0.99, respectively. The discount rate γ is set to 0.99. We use an entropy regularization of 0.01 for both the option

Figure 1: Comparison of average scores through training. All option critic models used ϵ-greedy policies over options, except for the last one mentioning "actor policy", meaning we use an actor as the policy over options and use policy gradient to learn it.

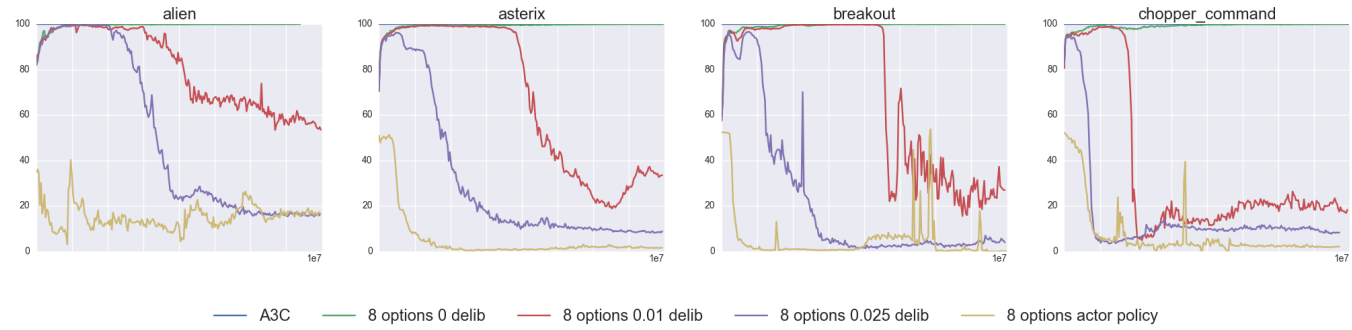

Figure 2: Comparison of average termination rates through training.

policies and the policy over options (if the actor is used). We use gradient norm clipping of 40. In our experiments we use 16 parallel threads to train, for 80M actions (or 320M frames).

When using ϵ-greedy as the policy over options, we use a fixed ϵ of 0.1. When using a deliberation cost, the cost linearly increases through half of training, where it reaches the wanted amount and then remains fixed for the second half of training. We found that starting with a high deliberation cost made options collapse to lasting the entire episode. We believe this is due to the fact that the values are initially random, and the option with the highest value will be pushed to last as long as possible without even learning the true value of the other options. By incrementing the cost in time, it allows the options to first be learned along with their values and the model learns to stretch them to appropriate lengths relative to the deliberation cost.

One notable difference is the number of steps used for the n-step return. In A3C, a training batch consists of a combination of a 1-step, 2-step, up to $t_{\max}$-step return, where $t_{\max}$ is chosen to be 5. In A2OC, we allow this parameter to vary depending on the length of the options, with a minimum of 5 steps. This means if the options are short, then $t_{\max}$ is the same, but if the options become longer, the training batch sees longer n-step returns. We also set a maximum n-step length of 30.

We test our algorithms on the Atari 2600 environment (Bellemare *et al.* [2013]), on 4 games: Alien, Asterix, Breakout and Chopper Command.

As we can see from figure 1, the curves for the A2OC with an actor policy over options are quite unstable. This is most likely due to the learning rate being to high. We kept the learning rates among all experiments fixed, without performing a hyper-parameter search. However, the A2OC with an actor as policy over options model has an extra objective, the policy gradient for the actor over options. As the gradients for all objectives are summed up and applied in one batch, this increases the scale of the update, which is most likely the source of instability.

Figure 2 shows how the average termination rates change through training. We can see that with no deliberation cost, A2OC's termination function learns quickly that it should always terminate. However, the models with deliberation costs raising to 0.01 and 0.025 both start with termination at 100%, but that it decreases through training. Furthermore, the 0.025 deliberation cost has much more of an effect than the 0.01 model. In the Alien environment, the higher deliberation

cost led to longer options, which in turn led to a better performance. We can also see that the increase in performance is aligned with the drop in termination rates in the 0.025 deliberation model.

The higher deliberation cost isn't always better, as we can see in the Asterix results. The model with 0.025 deliberation has longer options early on in training and performance is simply slowed down. On the other hand, the model with 0.01 deliberation keeps short options until halfway through training, at which point the options start becoming longer and it immediately starts outperforming other models.

We can see that the termination of the A2OC with actor policy always decreases to low amounts very early in training. As mentioned previously, having long options early in training has been detrimental to the overall performance as the model doesn't get the chance to learn different options. In future experiments, it would be worth trying to reduce the learning rate on the termination function when using the actor over options.

References

Pierre-Luc Bacon and Doina Precup. Learning with options: Just deliberate and relax. *NIPS Bounded Optimality and Rational Metareasoning Workshop*, 2015.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. *arXiv preprint arXiv:1609.05140*, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. 2013.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016.

Aurélia Léon and Ludovic Denoyer. Options discovery with budgeted reinforcement learning. *arXiv preprint arXiv:1611.06824*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Doina Precup. Temporal abstraction in reinforcement learning. 2000.

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

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. *arXiv preprint arXiv:1604.07255*, 2016.

Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Advances in Neural Information Processing Systems*, pages 3486–3494, 2016.