# Importance Sampled Option-Critic for More Sample Efficient Reinforcement Learning

**Karan Goel**
Machine Learning Department
Carnegie Mellon University
kgoel93@gmail.com

**Emma Brunskill**
Computer Science Department
Stanford University
ebrun@cs.stanford.edu

## Abstract

Recently, the Option-Critic architecture demonstrated the possibility of learning a hierarchical policy at roughly the same rate as flat policy deep reinforcement learning methods. In this work, we extend the Option-Critic architecture to improve its sample efficiency by leveraging off-policy information. We derive new gradient estimators that improve the incorporation of information from the agent's experience into all parts of the agent's policy. Preliminary results on a navigation task indicate that our proposed algorithm can speed up learning significantly.

## 1 Introduction and Related Work

Recent successes in reinforcement learning have made it possible to achieve super-human performance on Atari games [8], and the game of Go [12]. However, the sample efficiency of these approaches is often prohibitive, requiring large amounts of data and computation to learn a good policy. This makes it important for us to develop more sample-efficient algorithms that are applicable in practice.

Hierarchical reinforcement learning is an approach to reinforcement learning that decomposes the policy into several sub-policies that are active for different sub-tasks in the agent's environment. There are several classical approaches to hierarchical reinforcement learning such as the options framework [13], HAM [10], MAXQ [4]. This decomposition provides an avenue to learn policies more efficiently than primitive action policies, if the hierarchy is known ahead of time. Hierarchical policies can also be useful in transfer or multi-task learning [9, 1], where the ability to reuse sub-policies for different tasks becomes useful. However, an additional challenge is discovering the hierarchical structure in conjunction with learning the sub-policies. Option-Critic [2, 6] is a deep reinforcement learning approach in the options framework. Option-Critic learns at a rate comparable to popular deep reinforcement learning methods, an open question is whether these methods can be sped up to learn significantly faster.

There has been limited work on improving the sample-efficiency of hierarchical RL from a methodological perspective. Recent work [5, 7] uses a combination of hierarchical RL and intrinsic motivation to improve sample-efficiency on a variety of tasks. In this paper, we make a first attempt at addressing the question of how to improve the sample-efficiency of hierarchical reinforcement learning. We extend the results of Option-Critic [2] and derive new gradient estimators for end-to-end learning of the intra-option policies and termination functions. Our key observation is that we can use improtance sampling to leverage update all intra-option policies compatible with the selected action simultaneously, rather than just option that was used to select the chosen action. We describe our proposed approach in Section 3 and present preliminary experiments that show the promise of our approach in Section 4.

Hierarchical Reinforcement Learning Workshop at the 31st Conference on Neural Information Processing Systems (HRL@NIPS 2017), Long Beach, CA, USA.

## 2 Preliminaries

**Markov Decision Process (MDP).** An MDP is a 6 tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, p \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions (for ease of exposition, these are assumed to be discrete sets), $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in [0, 1)$ is a discount factor used to weigh the relative importance of future rewards and $p : \mathcal{S} \to [0, 1]$ is an initial state distribution. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines a conditional distribution over the actions given a state.

The value function of a policy $\pi$ is defined as the long-term expected reward that we expect to accumulate on following $\pi$: $V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 \sim p, s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)\right]$. Similarly, we define the action-value function of a policy $\pi$ as $Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} r_{t+1} | s_0 \sim p, a_0 = a, s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)\right]$.

**Options Framework.** An option is a temporally extended action defined by a closed-loop policy [13]. An option $o \in \Omega$ is a 3 tuple $\langle \mathcal{I}_o, \pi_o, \beta_o \rangle$ where $\mathcal{I}_o \subseteq \mathcal{S}$ is set of states in which $o$ can be initiated, $\pi_o : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is an intra-option policy and $\beta_o : \mathcal{S} \to [0, 1]$ is a termination function. As done in Bacon et al. [2], we assume that options are available everywhere, *i.e.* $\mathcal{I}_o = \mathcal{S}, \ \forall o \in \Omega$. A meta-policy $\pi_\Omega : \mathcal{S} \times \Omega \to [0, 1]$ decides how options should be picked.

An MDP with a set of options is equivalent to a Semi-Markov Decision Process (SMDP) which has a corresponding value-function over options $V_\Omega(s)$ and option-value function $Q_\Omega(s, o)$.

We employ the *call-and-return* option execution model: the agent first picks an option $o$ according to the meta-policy $\pi_\Omega$, then follows $\pi_o$ until a termination signal is sampled from $\beta_o$. The process then repeats. We will concern ourselves with the function-approximation setting, parameterizing the intra-option policies $\pi_{o,\theta}$ with parameter vector $\theta$ and the termination functions $\beta_{o,\vartheta}$ with parameter vector $\vartheta$.

Define $Q_\Omega(s, o) = \sum_a \pi_{o,\theta}(a|s) Q_U(s, o, a)$ as the state-option value function. $Q_U(s, o, a) = r(s, a) + \gamma \sum_{s'} \mathcal{P}(s'|s, a) U(o, s')$ is the state-option-action value function. $U(o, s') = (1 - \beta_{o,\vartheta}(s')) Q_\Omega(s', o) + \beta_{o,\vartheta}(s') V_\Omega(s')$ denotes the value of executing option $o$ on entering $s'$.

**Option-Critic Architecture**. Bacon et al. [2] derive two policy-gradient theorems for learning options where the goal is to maximize the expected discounted return $\rho(\Omega, \theta, \vartheta, s_0, o_0) = \mathbb{E}_{\Omega,\theta,o} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0, o_0\right]$, which depends on the parameters of the intra-option policies and the termination functions, as well as the meta-policy. The number of options $K$, is assumed to be fixed and known a-priori. The first result (Theorem 1, [2]) is an Intra-Option Policy Gradient Theorem,

$$\nabla_\theta Q_\Omega(s, o)|_{s_0,o_0} = \sum_{s,o} \mu_\Omega(s, o|s_0, o_0) \sum_a \nabla_\theta \pi_{o,\theta}(a|s) Q_U(s, o, a) \tag{1}$$

where $\mu_\Omega(s, o|s_0, o_0) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s, o_t = o|s_0, o_0)$ is a discounted weighing of state-option pairs starting from $s_o, o_0$. The second result (Theorem 2, [2]) is a Termination Gradient Theorem,

$$\nabla_\vartheta U(o, s')|_{o_0,s_1} = -\sum_{o,s'} \mu_\Omega(o, s'|o_0, s_1) \nabla_\vartheta \beta_{o,\vartheta}(s') A_\Omega(s', o) \tag{2}$$

where $A_\Omega(s', o) = Q_\Omega(s', o) - V_\Omega(s')$ is the advantage function over options. Equations 1 & 2 are used to perform on-policy updates to the parameters as samples are collected through interaction with the environment. In addition, a critic is maintained for $Q_\Omega$ which is used to estimate both $Q_U$ and $A_\Omega$ to perform the gradient updates. The critic's one-step target is estimated as,

$$g_t^{(1)} = r_{t+1} + \gamma \left( (1 - \beta_{o_t,\vartheta}) Q_\Omega(s_{t+1}, o_t) + \beta_{o_t,\vartheta} \max_o Q_\Omega(s_{t+1}, o) \right)$$

and $(Q_\Omega(s_t, o_t) - g_t^{(1)})^2$ is minimized.

**Importance Sampling.** Importance sampling is a statistical technique that can be used to evaluate the expectation of a random variable $E_{X \sim p(X)}[f(X)]$ when it is difficult to sample directly from its distribution $p(X)$ [11]. We can instead sample from a different distribution $q(X)$ and use $E_{X \sim q(X)} \left[\frac{p(X)}{q(X)} f(X)\right]$ to compute the expectation. Importance sampling is typically used in RL for off-policy learning [14].

## 3 Importance Sampled Option-Critic (IS-OC)

We now introduce the Importance Sampled Option-Critic algorithm. Our main idea is that an action taken by the agent under the picked option's intra-option policy can be used to update *all* other intra-option policies simultaneously, and not just the intra-option policy that took the action. Intuitively, taking the action allows us to determine whether it was a good action or not through the reward we receive, regardless of which option was used to execute the action. This information about the action should thus be used to adjust the action's probability under *all* the intra-option policies. If the action led us to a large reward, the agent should be much more likely to pick that action, regardless of the option that it picks to execute in the state. Similarly, information from terminating an option can be used to adjust the termination probabilities for all other options available to the agent.

### 3.1 Intra-Option Policy Gradients

We first proceed to rewrite Equation 1 using the log-derivative trick,

$$\sum_{s,o} \mu_\Omega(s, o|s_0, o_0) \sum_a \nabla_\theta \pi_{o,\theta}(a|s) Q_U(s, o, a) = \\ (1-\gamma)\mathbf{E}_{s,o\sim\mu_\Omega}\left[\mathbf{E}_{a\sim\pi_{o,\theta}}\left[\nabla_\theta \log \pi_{o,\theta}(a|s) Q_U(s, o, a)\right]\right]$$

where we have contracted the summations into expectations *w.r.t.* the appropriate probability distributions. We can drop the $(1-\gamma)$ constant and sample $s, o \sim \mu_\Omega$ so that we are now computing a stochastic gradient with respect to the stationary distribution of the policy. This leaves us with the inner expectation *w.r.t.* $\pi_{o,\theta}$, which is used to compute the gradient for the intra-option policy of option $o$.

We now use importance sampling to compute the intra-option policy gradient of this option $o$ with respect to the intra-option policy of a different option $\tilde{o}$ as follows,

$$\mathbf{E}_{a\sim\pi_{o,\theta}}\left[\nabla_\theta \log \pi_{o,\theta}(a|s) Q_U(s, o, a)\right] = \mathbf{E}_{a\sim\pi_{\tilde{o},\theta}}\left[\frac{\pi_{o,\theta}(a|s)}{\pi_{\tilde{o},\theta}(a|s)} \nabla_\theta \log \pi_{o,\theta}(a|s) Q_U(s, o, a)\right] \quad (3)$$

which yields an importance-sampled gradient estimator for the intra-option policy of any option $o$, when an action is sampled from $\tilde{o}$.

Procedurally, suppose the agent is in state $s$ following option $o_i$ and samples an action from $\pi_{o_i,\theta}(\cdot|s)$. We would update the intra-option policies for all options $o_1, \ldots, o_K$, using a stochastic update derived from Equation 3 above $\left(\theta \leftarrow \theta + \alpha_\theta \sum_{j=1}^{K} \frac{\pi_{o_j,\theta}(a|s)}{\pi_{o_i,\theta}(a|s)} \nabla_\theta \log \pi_{o_j,\theta}(a|s) Q_U(s, o_j, a)\right)$ where $\alpha_\theta$ is the learning rate for $\theta$.

### 3.2 Termination Gradients

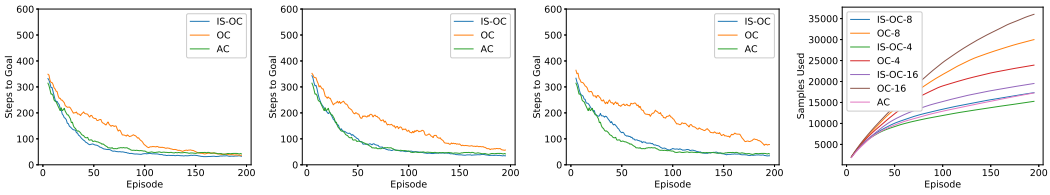
We start by rewriting Equation 2 using the log-derivative trick,

$$-\sum_{o,s'} \mu_\Omega(o, s'|o_0, s_1) \nabla_\vartheta \beta_{o,\vartheta}(s') A_\Omega(s', o) = (1-\gamma)\mathbf{E}_{o,s'\sim\mu_\Omega}[-\beta_{o,\vartheta}(s') \nabla_\vartheta \log \beta_{o,\vartheta}(s') A_\Omega(s', o)]$$

Once again, suppose we sample $o, s' \sim \mu_\Omega$ and drop the $(1-\gamma)$ constant,

$$\begin{aligned} -\beta_{o,\vartheta}(s') \nabla_\vartheta \log \beta_{o,\vartheta}(s') A_\Omega(s', o) &= -\nabla_\vartheta \log \beta_{o,\vartheta}(s') A_\Omega(s', o)[\beta_{o,\vartheta}(s') + \underbrace{0 \times (1 - \beta_{o,\vartheta}(s'))}_{\text{adding 0}}] \\ &= -\nabla_\vartheta \log \beta_{o,\vartheta}(s') A_\Omega(s', o)[\underbrace{1 \times \beta_{o,\vartheta}(s') + 0 \times (1 - \beta_{o,\vartheta}(s'))}_{= \mathbf{E}_{b\sim\beta_{o,\vartheta}}[b] \text{ where } b \in \{0,1\}}] \\ &= -\nabla_\vartheta \log \beta_{o,\vartheta}(s') A_\Omega(s', o) \mathbf{E}_{b\sim\beta_{o,\vartheta}}[b] \end{aligned}$$

where we have rewritten the gradient by introducing an additional random variable $b$ that denotes whether an option termination has occurred $(b = 1)$ or not $(b = 0)$.




(a) 4 Options (b) 8 Options (c) 16 Options (d) Sample efficiency

Figure 1: Comparison of Option-Critic (OC) and Importance Sampled Option-Critic (IS-OC) on the four-rooms navigation domain. Results are averaged over 50 repetitions and the curves are smoothed for clarity using a running mean window of size 10.

We now importance sample the termination gradient of any $o$ with respect to the termination distribution of $\tilde{o}$ as follows,

$$-\nabla_\vartheta \log \beta_{o,\vartheta}(s')A_\Omega(s',o)\mathbf{E}_{b\sim\beta_{o,\vartheta}}[b] = -\nabla_\vartheta \log \beta_{o,\vartheta}(s')A_\Omega(s',o)\mathbf{E}_{b\sim\beta_{\tilde{o},\vartheta}}\left[\frac{\beta_{o,\vartheta}}{\beta_{\tilde{o},\vartheta}} \times b\right] \quad (4)$$

Again, procedurally the agent executes an action under $o_i$, and then arrives in $s'$. It then samples $b \sim \beta_{o_i,\vartheta}(s')$. We can then proceed to update the termination policies for all options $o_1,\ldots,o_K$, using a stochastic update derived from Equation 4, $\left(\vartheta \leftarrow \vartheta - \alpha_\vartheta \left[\nabla_\vartheta \beta_{o_i,\vartheta}(s')A_\Omega(s',o_i) + b\sum_{j=1,j\neq i}^{K} \frac{\beta_{o_j,\vartheta}}{\beta_{o_i,\vartheta}} \nabla_\vartheta \log \beta_{o_j,\vartheta}(s')A_\Omega(s',o_j)\right]\right)$ where $\alpha_\vartheta$ is the learning rate for $\vartheta$.

## 4 Experiments

We present a preliminary experiment on a navigation task in the four-rooms domain of Bacon et al. [2]. The goal of our experiment is to show how the importance sampled gradients can lead to faster learning with a fixed number of episodes.

We reproduced the setup introduced in Bacon et al. [2], and compare the performance of Option-Critic (OC) to Importance Sampled Option-Critic (IS-OC) as well as Actor-Critic (AC), implemented as OC with 1 option[1]. Intra-option policies are parameterized with Boltzmann distributions (temperature of 0.01), and termination distributions with sigmoid functions. In addition, both methods use a baseline to reduce the variance of the gradient estimators.

Results are shown in Figure 1 – in all three cases, we see that IS-OC significantly speeds up learning over OC and matches the speed of AC, indicating that there is almost no slowdown due to the presence of the options unlike in OC. In addition, we note that while OC incurs a slowdown when learning with 8/16 options, the slowdown is negligible for IS-OC – this is perhaps an indication that IS-OC may be able to scale up to a larger number of options than OC which may be useful in more complex domains, but this requires more investigation. IS-OC also uses far fewer samples (and consequently fewer updates) than OC as seen in Figure 1d although each update is computationally more expensive.

In future revisions of this work, we will be testing IS-OC on larger domains such as the Arcade Learning Environment [3].

## 5 Conclusion

In this work, we introduced the Importance Sampled Option-Critic architecture for improving the sample-efficiency of the Option-Critic algorithm. A small set of preliminary experiments confirmed the promise of our approach, and indicated that IS-OC can be used to improve the integration of sample information inside the agent's overall policy to yield faster learning.

[1]We reimplemented Option-Critic in Pytorch and were able to match the performance reported in Bacon et al. [2] on this domain.

## References


[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

[3] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael H. Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013.

[4] Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.*, 13:227–303, 2000.

[5] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *CoRR*, abs/1704.03012, 2017.

[6] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option : Learning options with a deliberation cost. *CoRR*, abs/1709.04571, 2017.

[7] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, 2016.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[9] Junhyuk Oh, Satinder P. Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*, 2017.

[10] Ronald E. Parr and Stuart J. Russell. Reinforcement learning with hierarchies of machines. In *NIPS*, 1997.

[11] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, 2000.

[12] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas R Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550 7676:354–359, 2017.

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

[14] Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *ICML*, 2016.