SOFT OPTIONS CRITIC

WORK IN PROGRESS

December 6, 2019

ABSTRACT

Hierarchical Reinforcement learning aims to solve complex tasks by learning reusable skills that exploit the compositional structure of the task to speed up learning. The option critic architecture successfully demonstrates a way to automate the learning of these skills while solving a specific task. However, this framework uses on-policy updates which makes it highly sample inefficient and impractical for solving control tasks with high dimensional state and action space. Additionally, this framework may not necessarily generate diverse skills and often collapses to learning a single skill in naive tasks. In this work, we show how to extend options the options framework to generate diverse maximum entropy options that specialize in different parts of state-action space. Experimental results show that our proposed framework enables the agent to learn to achieve higher rewards faster than vanilla options-critic, Hierarchical RL via advantage-weighted information maximization framework (AdInfoHRL), and state of the art algorithms like PPO and Soft Actor-Critic in many control tasks in addition to learning more interpretable skills.

1 Introduction

Enabling an agent to learn complex and long tasks has been a challenging problem in RL for several decades. Humans often approach such complex tasks by identifying simpler sub-tasks and learning skills to solve them. Hence, recent research in RL has focused on developing hierarchical frameworks that enable an agent to exploit the compositional structure of the task and learn reusable skills as opposed to learning with primitive actions. The options framework provides a way to learn and represent these skills as temporally extended actions also termed as options. Prior research has shown that these temporal abstractions significantly improve exploration and reduce the complexity of selecting actions. The options-critic architecture uses this framework to proposes a set of theorems that makes it feasible to parameterize and learn various aspects of options in an end-to-end manner using stochastic gradient descent method. However, this framework and several of its variants have two major limitations 1) The framework uses on-policy updates for learning skills, which require new samples to be generated for nearly every update and hence is highly sample inefficient. This makes it impractical for learning complex control tasks with high dimensional state and action space 2) The framework may not necessarily generate diverse and interpretable skills and often result in learning a single skill in simple tasks. To address the issue of sample-efficiency, we employ the Soft-Actor Critic algorithm for learning maximum entropy option-policies from past experiences. Using soft policy improvement theorem Haarnoja et al. (2018a), we derive intra-option policy gradient and intra-option termination gradient. Learning diverse skills which can be selectively reused in future tasks is a desired property in HRL frameworks. To enable this, the agent must learn skills that specialize in different parts of the state space. For example, we would expect a bipedal robot to learn 1 option for jumping and another option for walking. Prior research has shown how to generate diverse skills by either by maximizing the mutual information between state and option in a non-task setting Eysenbach et al. (2018a) or segmenting demonstration trajectories and learning a skill for each segment[Niekum et al. (2012),Konidaris et al. (2010)]. However, in the absence of demonstration data, most HRL framework only maximizes returns and rely on the learning network to implicitly discover abstractions in state-action space in the process. Hence, these frameworks may not learn diverse interpretable skills. We, therefore, propose a framework which clusters state-action tuples such that states with similar distribution over actions as induced by the optimal policy are in the same cluster. The framework then uses this clustering mechanism to train the policy over options to allocate options to different parts of the state-action space. We finally evaluate the proposed soft-option critic framework on OpenAI Roboschool tasks with continuous state and action space. Our experimental results show that not only does the proposed framework enable an agent to achieves higher rewards much faster than vanilla options-critic and state of the art algorithms like PPO and Soft Actor-Critic in many control tasks but also learns more interpretable skills.

2 Related Work

Several frameworks have been proposed to enable an agent to learn skills to solve hierarchical tasks. Bacon et al. (2017) proposed to learn skills by directly maximizing the expected returns using policy gradient method. However, their framework learns a single skill if the environment is simple or the function approximator used is to represent each skill is complex enough to learn a policy for the whole state space. Konidaris et al. (2012) uses MAP change point detection to segment trajectories when its value function is too complex and learns a policy corresponding to each segment. However, this paper assumes that value functions for a sub-goal should be able to be represented using linear function approximator which does not hold in complex environments. Niekum et al. (2012) generates diverse skills by segmenting demonstration trajectories using Beta Process Autoregressive HMM and learns a skill for each segment. Recently Eysenbach et al. (2018b) proposed a method that generates diverse skills by maximizing the mutual information between state and option in a non-task setting. Henderson et al. (2018) learns a gaussian mixture model to represent skills and introduces mutual information and variance-based regularization to encourage specialization of each skill in different parts of the state space. However, this framework forces hierarchy to emerge by adding penalties to encourage diversity and does not always result in learning interpretable skills. Osa et al. (2019) proposed learning latent representations for skills using advantage-weighted information maximization. Although this framework generates diverse skills, it uses deterministic policies to represent skills and also does not generate maximum entropy skills which have been shown to be more robust in previous literature. Hence, in this project, we take inspiration from this work to generate more interpretable and robust skills by using Deep Embedding Clustering with an action-representation loss to learn diverse maximum entropy skills.

3 Background

In Reinforcement learning, the environment is modelled as a Markov Decision Process (MDP) which is defined as a tuple $< S, A, P, r, d_0, \gamma >$ where S is set of states of the environment as perceived by the agent, A is the set of actions that the agent can take, $P : S \times A \rightarrow (S \rightarrow [0, 1])$ is the state transition probability, $r : S \times A \rightarrow R$ is the reward function, d_0 is the initial state distribution ie- $Pr(s_0 = s)$ and $\gamma \in [0, 1]$ is the factor by which rewards are discounted over time. At any discrete time step $t \in \{0, 1, 2..T\}$, the agent observes state $s_t \in S$and takes action $a_t \in A$ which transitions the agent to state s_{t+1} at time step $t + 1$ and the agent receives a reward r_t. In a finite horizon MDP, this process ends after T time steps or sooner if the agent reaches the termination state. When this process ends, we say that an episode is completed and t is reset to 0. In an infinite horizon MDP, $T \rightarrow \infty$. The returns of an episode is defined as $G = \sum_{t=0}^{\infty} r^t R_t$. A policy $\pi : S \times A \in [0, 1]$ is the probability distribution over actions conditioned on the given state. The value function of policy π is defined as the discounted returns starting from state s ie- $V_\pi(s) = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s]$. Likewise, the action-value function of a policy is defined as the discounted returns starting from state s and taking action a - $Q_\pi(s_t, a_t) = \mathbf{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0, a_0]$. A policy is said to be greedy with respect to A policy π is said to be greedy with respect to a given action value function Q if $\pi(s_t, a_t) > 0$ and $a_t = argmax_a Q(s_t, a)$. The value function V^π and action-value function Q^π can be learnt using one-step TD learning TD(0) which is a special case of $TD(\lambda)$(Sutton et al [1998]). The action value is updated using the equation $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta$ where α is the step size. We define δ is the TD(0) error $\delta = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$.

Options Framework The options framework (Sutton, Precup, and Singh 1999; Precup 2000) represents options as temporally extended actions. Options $\omega \in \Omega$ are markovian and are defined as a tuple$(I_\omega, \pi_\omega, \beta_\omega)$ where $I_\omega \subseteq S$ is the initiation set which comprises of states in which the option can be initiated, π_ω is the intra-option policy which gives the option's probability distribution over actions conditioned on states, and $\beta_\omega : S \rightarrow [0, 1]$ is a termination function which determines the probability of terminating the option in a given state. Options can be viewed as skills acquired by the agent to solve sub-tasks within a task. An MDP with a set of options becomes a Semi-Markov Decision Process (SMDP) Sutton et al. (1999). This SMDP has an optimal value function over options $V_\Omega(s)$ and option-value function $Q_\Omega(s, \omega)$. Bellman equation can be used for updating the value of a state-option pair Sutton et al. (1999) is given as $Q(s_t, \omega_t) \leftarrow Q(s_t, \omega_t) + \alpha[r_t + \gamma(1 - \beta_{\omega,v}(s_t))Q(s_{t+1}, \omega_t) + \gamma\beta_{\omega,v}(s_t)max_{\omega_{t+1} \in \Omega} Q(s_{t+1}, \omega_{t+1}) - Q(s_t, \omega_t)]$ where ω is obtained from the policy over options π_Ω.

Options critic framework The options-critic architecture Bacon et al. (2016) introduces a way to apply policy gradient algorithms to temporally extended actions to directly maximize the expected discounted returns.

The value of executing an action at a given state-option pair s_t, w_t given by : $Q_\omega : S \times A \rightarrow R$ where $Q_\omega(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t)((1 - \beta_{\omega,v}(s_{t+1}))Q_\Omega(s_{t+1}, \omega_t) + \beta_{\omega,v}(s_{t+1})V_\Omega(s_{t+1}))$. The expected discounted returns starting at a state s_0 and option ω_0 is represented as $\rho(\Omega, \theta, \mu, s_0, \omega_0) = \mathbf{E}_{\mathbf{\Omega},\theta,\omega}[\sum_{t=0}^{\infty} \gamma^t r^t \mid s_0, \omega_0]$. Th authors derive the intra-option gradient theorem which states that the gradient of expected discounted returns with respect to the option policy parameters and initial conditions (s_0, ω_0) can be evaluated as $\frac{\partial}{\partial\theta}\rho(\pi, s_0, \omega_0) = \sum_{s_t,\omega_t}[\nu(s_t, \omega_t \mid s_0, \omega_0) \sum_a \frac{\partial}{\partial\theta}\pi_{\omega_t,\theta}(s_t, a_t)Q_U(s_t, \omega_t, a_t)]$ where $\mu(s_t, \omega_t \mid s_0, \omega_0)$ is the discounted state-option pair distribution with $\mu(s_t, \omega_t \mid s_0, \omega_0) = \sum_{t=0} \infty \gamma^t P(s_t = s, \omega_t = \omega | s_0, \omega_0)$. Likewise, the termination gradient theorem provides the gradient of the expected discounted returns with respect to option termination parameters μ and the initial condition (s_1, ω_0) as: $\frac{\partial}{\partial\nu}\rho(\pi, s_1, \omega_0) = -\sum_{s_{t+1},\omega_t} \mu(s_{t+1}, \omega_t \mid s_1, \omega_0)\frac{\partial}{\partial\nu}\beta_{\omega,v}(s_t)A(s_{t+1}, \omega_t)$ where A is the advantage function given as $A_\Omega(s_t, \omega_t) = Q_\Omega(s_t, \omega_t) - V_\Omega(s_t)$

Soft Actor Critic The Soft Actor Critic architecture Haarnoja et al. (2018b) provides an off-policy actor critic algorithm based on maximum entropy framework that outperforms policy gradient methods like PPO, TRPO in terms of sample efficiency. The algorithm is derived from the maximum entropy variant of policy iteration method. It optimizes the following objective.

$$J(\pi) = \sum_{t=0}^{\infty} \mathbf{E}_{s_t, a_t \sim \rho_\pi} \left[\sum_{l=t}^{\infty} \gamma^{l-t} \mathbf{E}_{s_l \sim p, a_l \sim \pi} \left[r(s_t, a_t) + \alpha H(\pi(. \mid s_t) \mid s_t, a_t) \right] \right]$$

The temperature parameter α indicates the relative importance of the entropy term against reward and determines the stochasticity of the optimal policy. This objective maximizes the expected discounted returns and entropy for future states starting from every state-action pair (s_t, a_t) weighted by its probability ρ_π under current policy.
The soft policy iteration alternates between soft policy evaluation and soft policy improvement. In policy evaluation , soft Q value is computed iteratively, starting from any function Q:S $\times$ A $\rightarrow \mathbb{R}$ and repeatedly applying the modified Bellman operator T^π given by:

$$T^\pi Q(s_t, a_t) \equiv R(s_t, a_t) + \gamma \mathbf{E}_{\mathbf{s_{t+1} \sim p}}[V(s_{t+1})]$$

$$\text{where} \tag{4}$$

$$V(s_t) = \mathbf{E}_{\mathbf{a_t} \sim \pi}[Q(s_t, a_t) - \alpha \log \pi(a_t \mid s_t)] \tag{5}$$

In the policy improvement step, the policy is updated to follow the distribution of exponential of the soft Q function. Hence the policy is updated according to the following objective:

$$\pi_{new} = \arg \min_{\pi' \in \prod_\theta} D_{KL} \left(\pi'(\cdot \mid s_t) \middle| \frac{exp(Q^{\pi_{old}}(s_t, \cdot))}{Z^{\pi_{old}(s_t)}} \right) \tag{8}$$

Deep Embedded Clustering
Deep Embedded clustering is a pioneer clustering algorithm which has been popularly used to benchmark performance of other clustering algorithms. It consist of an autoencoder parameterized by ψ which is pre-trained using reconstruction loss $((l)(\psi)$ to output latent representations of the input z. The authors uses K-Means to derive cluster centers u in latent space and derives soft-cluster assignment q based on Student-t distribution with degree of freedom set to 1.

$$q_{ij} = \frac{(1 + ||z_i - u_j||^{\frac{2}{\alpha}})^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + ||z_j - u_{j'}||^{\frac{2}{\alpha}})^{-\frac{\alpha+1}{2}}} \tag{1}$$

$$p_{ij} = \frac{\frac{q_{ij}^2}{f_j}}{\sum_{j'} \frac{q_{ij}^2}{f_{j'}}} \tag{2}$$

where q_{ij} is the probability that the i_{th} input z_i belongs to j_{th} cluster u_j in latent feature space and f_j is the frequency of occurence of j_{th} cluster.
This assignment is further refined using KL Divergence between auxiliary target p_{ij} distribution derived from the soft-cluster assignment and the soft-cluster assignment distribution to learn learning latent representations that encourage hard-cluster assignments.

$$L_{DC}(\psi) = KL(P||Q) + L_{reconstruction} = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} + l(\psi) \tag{3}$$

The network parameters ψ is optimized using Stochastic Gradient Descent and cluster centers u are updated using KMeans Clustering in latent space after every K iterations.

4 Soft Option Critic

In this section, we will first describe how to derive meaningful clusters in latent-space of state-action tuples. We will then present a method to learn diverse maximum entropy options using this clustering mechanism. Here onwards, we would interchangeably refer to policy over options as inter-option policy and option policy as intra-option policy.

4.1 Learning latent representations via Deep Embedded Clustering

To learn diverse options, each option must be localized to a distinct sub-part of the state-action space. To enable this, Osa et al. (2019) proposes learning options that correspond to different modes of the advantage function. Since finding modes of the advantage function is a hard problem, the authors assume that state-action pairs that have high advantage value are more likely to occur. Hence finding modes of the advantage function $A^{\pi}(s, a)$ correspond to finding modes in the state-action density induced by the optimal policy π. To find the modes of state-action space density induced by the optimal policy, we use the trajectories of the most recent policies to train an autoencoder to estimate $p(\omega|s_t, a_t; \psi)$ where $\omega \in \Omega$. This autoencoder is training using Deep Embedded Clustering Xie et al. (2015) to encourage the hard-clustering of state-action pairs. We also use the Virtual Adversarial training regularization introduced by Miyato et al. (2015) to ensure that the autoencoder does not memorize the input and learns meaningful latent representations of state-action pairs.

$$J(\psi) = L_{DC}(\psi) + l_{vat} \tag{4}$$

$$\tag{5}$$

$L_{DC}(\psi)$ is the Deep Embedded Clustering loss Xie et al. (2015) consisting of the KL divergence loss between auxiliary target distribution and soft-cluster assignments and reconstruction loss; and $l_{vat} = KL(p(\omega|s^{noise}, a^{noise}; \psi)||p(\omega|s, a; \psi))$ where $s^{noise} = s + \epsilon$, $a^{noise} = a + \epsilon$ and $epsilon \sim \mathcal{N}(0, 1)$.
Additionally, to ensure that states with similar distributions over actions induced by the optimal policy have smaller Euclidean distance in the latent feature space, we add the following modified actionable distance loss proposed by Ghosh et al. (2018) to learn actionable state representations in complex environments.

$$D_{Act} = \mathbf{E}_{\mathbf{s}}[[D_{KL}(\pi(a|s_1), \pi(a|s_2)) + D_{KL}(\pi(a|s_2), \pi(a|s_1))]] \tag{6}$$

$$J(\psi) = J(\psi) + \mathbf{E}_{(s_1,a_1),(s_2,a_2)\sim\pi}[||\psi(s_1, a_1) - \psi(s_2, a_2)||_2 - D_{Act}(s_1, s_2)]^2 \tag{7}$$

The option-policy network is used to determine which experience tuples sampled from a replay-buffer which maintains the agent's past experiences; should be used to train each option-policy network and corresponding value function network.

4.2 Soft Option Evaluation

For a fixed option policy, the Soft Q-value function is computed iteratively, starting from any function $Q : S \times \Omega \times A$ and repeatedly applying a modified Bellman backup operator T^{π} given by:

$$\begin{aligned} T^{\pi}Q_{\omega}(s_t, a_t) \equiv & R(s_t, a_t) \qquad (8) \\ & + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} \mid s_t, a_t)((1 - \beta_{\omega,\upsilon}(s_{t+1}))Q_{\Omega}(s_{t+1}, \omega_t) + \beta_{\omega,\upsilon}(s_{t+1})V_{\Omega}(s_{t+1}).Q_{\Omega}(s_t, \omega_t)) \qquad (9) \end{aligned}$$

where Ω is the inter-option policy and

$$Q_{\Omega}(s, \omega) = \int \pi_{\omega}(a|s)Q_{\omega}(s, a)da \tag{10}$$

$$\pi_{\Omega}(\omega|s) = p(\omega|s) = \frac{exp(Q_{\Omega}(s, \omega))}{\sum_{\omega'} exp(Q_{\Omega}(s, \omega'))} \tag{11}$$

$$V_{\Omega}(s) = \sum_{\omega \in \prod_{\omega}} \pi_{\Omega}(\omega|s)Q_{\Omega}(s, \omega) \tag{12}$$

We prove the convergence of the modified Bellman backup operator in the appendix section.

4.3 Soft Option Improvement

In the soft policy improvement step, we minimize the KL divergence between the intra-option policy distribution and normalized form of exponential of the new intra-option Q value function. We derive the soft intra-option policy update by taking the gradient of the KL divergence objective with respect to intra-option policy parameters.

$$\pi_{\omega}^{new} = argmin_{\pi'_{\omega} \in \prod_{\omega}} D_{KL}\left(\pi'_{\omega}(\cdot \mid s_t) \middle| \frac{exp(Q^{\pi_{\omega}^{old}}(s_t, \cdot))}{Z^{\pi_{\omega}^{old}(s_t)}}\right) \tag{11a}$$

This objective is used for training option-policies and results in maximizing the discounted expected reward and entropy over actions for future states originating from every state-option-action tuple (s_t, w_t, a_t) weighted by its probability ρ_π under the current policy.

$$J(\Omega) = \sum_{t=0}^{\infty} \mathbf{E}_{\mathbf{s_t},\mathbf{w_t},\mathbf{a_t} \sim \rho_\pi}[\sum_{l=t}^{\infty} \gamma^{l-t} \mathbf{E}_{\mathbf{s_l} \sim \mathbf{p}, \mathbf{w_l} \sim \pi_{\mathbf{\Omega}}, \mathbf{a_l} \sim \pi_{\omega_\mathbf{t},\theta}}[r(s_t, a_t) + \alpha_{\omega_t,\theta_t} H(\pi_{\omega_t,\theta}(. \mid s_t) \mid s_t, w_t, a_t)]] \tag{13}$$

4.4 Architecture

Since the proposed framework should be able to solve complex tasks with continuous state and action space, we use function approximators to approximate intra-option Q-value function $Q_\omega(s_t, a_t)$, intra-option policy $\pi_{\omega,\theta}(s_t, a_t)$, option policy $p(\omega|s_t, a_t; \psi)$ and termination function $\beta_{\omega,v}(s_t)$. ϕ, θ, ψ and v represent the function approximators for these networks in the order specified above. The updates make use of target intra-option Q-value network whose weights are updated with exponential moving average of the intra-option Q-value network weights. We represent target intra-option Q-value function by $\bar{Q_\omega}(s_t, a_t)$. We represent the intra-option policy as a Gaussian function whose mean and covariance is given by the inter-option policy network. The inter-option network outputs probabilities over options for a given input state. We also use two intra-option Q-value networks and use the minimum of the two Q-values to mitigate positive bias that degrades performance in policy improvement step of value-based methods. We maintain an off-policy replay-buffer to store all the agent's experiences and a semi on-policy buffer to store the most recent N experiences. We randomly sample a mini-batch from the off-policy replay buffer and update the intra-option policy intra Q-value function in an off-policy manner after every M timesteps. Likewise, when the semi on-policy buffer is full, we update the option policy $p(\omega|s_t, a_t; \psi)$ using the recent experience tuples and empty the semi on-policy replay buffer when done. We now derive off-policy updates for each of these networks.

4.5 Soft Option Evaluation and Update:

In the option evaluation step, the intra-option Q-value function is trained to minimize the soft Bellman residual error.

$$\begin{aligned} J_Q(\phi) = \mathbf{E}_{\mathbf{s_t},\mathbf{w_t},\mathbf{a_t}}[\tfrac{1}{2}(Q_{\phi,\omega_t}(s_t,a_t) - (R(s_t,a_t) + \gamma \mathbf{E}_{\mathbf{s_{t+1}}}[\textstyle\sum_{w_{t+1}}(1-\beta_{w,v}(s_{t+1}))I_{w_t=w_{t+1}} \pi_{\omega_t,\theta}(a_{t+1}|s_{t+1}) Q_{\bar{\phi},\omega_t}(s_{t+1},a_{t+1}) \\ + \beta_{w,v}(s_{t+1})(\pi_\Omega(w_{t+1}|s_{t+1}) \pi_{\omega_{t+1},\theta}(a_{t+1}|s_{t+1}) Q_{\bar{\phi},\omega}(s_{t+1},a_{t+1}))]))^2] \end{aligned}$$

Differentiating with respect to ϕ we get:

$$\begin{aligned} \frac{\partial}{\partial \phi} J_Q(\phi) = \frac{\partial}{\partial \phi} Q_{\omega_t,\phi}(s_t, a_t)(Q_{\phi,\omega_t}(s_t, a_t) \\ -(R(s_t, a_t) + \gamma \mathbf{E}_{\mathbf{s_{t+1}}}[\sum_{w_{t+1}}((1 - \beta_{w,v}(s))I_{w_t=w_{t+1}} \pi_{\omega_t,\theta}(a_{t+1}|s_{t+1}) \bar{Q}_{\phi,\omega_t}(s_{t+1}, a_{t+1}) \\ +\beta_{w,v}(s_{t+1})(\pi_\Omega(w_{t+1}|s_{t+1}) \pi_{\omega_{t+1},\theta}(a_{t+1}|s_{t+1})(\bar{Q}_{\phi,\omega_{t+1}}(s_{t+1}, a_{t+1}))))])) \\ \phi = \phi - \lambda_\phi \nabla J_Q(\phi) \end{aligned}$$

4.6 Intra-option policy update:

In the policy improvement step, we update intra-option policy distribution towards exponential of the intra-option Q value function. We derive the intra-option policy update by taking the gradient of the KL divergence objective in equation 11a . Simplifying this objective, we get:

$$J_\pi(\omega, \theta) = \mathbf{E}_{\mathbf{s_t}}[\alpha_{\omega,\theta} log \pi_{\omega,\theta}(a_t|s_t) - Q_{\omega_t}(s_t, a_t)]$$

We derive the soft intra-option gradient by taking the gradient of the objective $J(\Omega)$ with respect to intra-option policy parameters θ. Since the output of the intra-option policy network is a gaussian distribution over the space of actions, we use reparametrisation trick to obtain a low variance estimate of the gradient of loss with respect to inter-option policy parameters. We reparameterize the output of the inter-option policy network as follows:

$$a_t = f'_{\omega,\theta}(\epsilon; s_t) \tag{14}$$

$$\frac{\partial}{\partial\theta} J_\pi(\Omega) = \frac{\partial}{\partial\theta}(\mathbf{E}_{\mathbf{s_t},\epsilon\sim\mathbf{N(0,1)}}[-(Q(s_t, w_t, a_t) - \alpha_{\omega,\theta}\log\pi_{\omega,\theta}(a_t|s_t))]) \tag{15}$$

$$\frac{\partial}{\partial\theta} J(\omega,\theta) = \alpha_{\omega,\theta}\frac{\partial}{\partial\theta}\log\pi_{\omega,\theta}(s_t, w_t, a_t) + \left(\alpha_{\omega,\theta}\frac{\partial}{\partial a_t}\log\pi_{\omega,\theta}(s_t, a_t) - \frac{\partial}{\partial a_t}Q_\phi(s_t, w_t, a_t)\right)\frac{\partial}{\partial\theta}f'_{\omega,\theta}(\epsilon; s) \tag{16}$$

$$\theta \leftarrow \lambda_\theta \nabla_\theta J_\pi(\omega,\theta) \tag{17}$$

4.7 Termination function update:

The option critic termination gradient theorem states that the goodness of the termination function can only be evaluated upon entering the next state. We derive the termination update by taking the gradient of inter-option Q-value function with respect to the termination function parameters υ:

$$\frac{\partial}{\partial\upsilon}(J_\pi(\upsilon)) = -\frac{\partial}{\partial\upsilon}(R(s_t, a_t) + \gamma\mathbf{E}_{\mathbf{s_{t+1}}}[\sum_{w_{t+1}}(1-\beta_{w,v}(s_{t+1}))I_{w_t=w_{t+1}}\bar{Q}(s_{t+1}, w_{t+1}) \tag{18}$$

$$+\beta_{w,v}(s_{t+1})(\pi_\Omega(w_{t+1}|s_{t+1})\bar{Q}(s_{t+1}, w_{t+1}))]) \tag{19}$$

$$= \frac{\partial}{\partial\upsilon}\beta_{w,v}(s_{t+1})(Q(s_{t+1}, w_t) - V_\psi(s_{t+1})) \tag{20}$$

$$\upsilon \leftarrow \lambda_\upsilon \nabla_\upsilon J(\upsilon) \tag{21}$$

4.8 Soft Option Critic Algorithm

```
Input: $\psi, \phi_1, \phi_2, \psi, \theta, T, \upsilon, update\_num, M$
$n \leftarrow n$
$D_{off\_policy} \leftarrow []$
$D_{on\_policy} \leftarrow []$
$\bar{\phi_1} \leftarrow \phi_1, \bar{\phi_2} \leftarrow \phi_2$
$\alpha_{\theta,\omega_i} \leftarrow T$
for each timestep do
  $s \leftarrow s_0$
  done $\leftarrow$ False
  Choose $\omega$ according to $p\omega|s_t$
  for each environment step do
    $a_t \sim \pi_{w_t,\theta}(a_t \mid s_t)$
    Take action $a_t$ and observe state $s_{t+1}, done$
    $D_{on\_policy} \leftarrow D_{on\_policy} \bigcup \{s_t, a_t, r(s_t, a_t), s_{t+1}, done\}$
    $D_{off\_policy} \leftarrow D_{off\_policy} \bigcup \{s_t, a_t, r(s_t, a_t), s_{t+1}, done\}$
    if $beta_{\omega_t,\upsilon}$ terminates in $s_{t+1}$ then
      choose new $\omega$ according to $p\omega|s_t$
    end if
  end for
  if timestep mod M =0 then
    for each iteration in update_num do
      sample mini-batch from $D_{off\_polcy}$
      estimate $p(\omega|s_t, a_t)$ to each experience tuple
      Assign $\omega_t = \arg\max p(\omega|s_t, a_t)$ to each experience tuple
      $\theta \leftarrow \lambda_\theta \nabla_\theta J_\pi(\theta, \omega)$
      $\phi_i \leftarrow \lambda_\phi \nabla_\phi J(\phi_i) \forall i \in \{1, 2\}$
      $\alpha_{\theta,\omega_i} \leftarrow \lambda_{\alpha_{\theta,\omega_i}} \nabla_{\alpha_{\theta,\omega_i}} J(, \alpha_{\theta,\omega}) \forall i \in \{1, 2..|\omega|\}$
      $\upsilon \leftarrow \lambda_\upsilon \nabla_\upsilon J(\upsilon)$
      $\bar{\phi_i} \leftarrow \tau\phi_i + (1-\tau)\bar{\phi_i} \forall i \in \{1, 2\}$ Update target network weights.
    end for
  end if
  if $D_{on\_policy}$ is full then
    Update option-policy by minimizing Eq 6
    Clear on-policy replay buffer
  end if
end for
```

Algorithm 1: Soft Option Critic

5 Experiments

The goal of our experiments is to compare the sample complexity of the proposed framework with vanilla option critic and other state of the art algorithms like PPO and SAC. We use bullet Roboschool Benchmark Suite (BulletPhysics (2015)) which is a port of OpenAI Roboschool environments (Dhariwal et al. (2017)). This suite consists of a set of robotic control tasks with continuous state and action space. The state-space represents the joint information of the robots and the action space represents the torque applied at various joints. We evaluate the options framework on 5 robotic locomotion tasks - HopperBulletEnv-v0, HalfCheetahBulletEnv-v0, AntBulletEnv-v0, Walker2DBulletEnv-v0 and ReacherBulletEnv-v0. We also try to understand the effect of adding sparsity and variance regularization on the skills generated by the proposed framework.

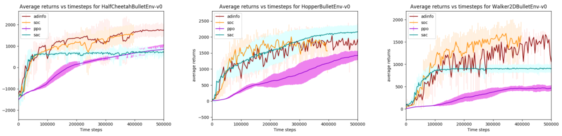

Figure 1: Average returns observed during training in Hierarchical RL via ADInfoHRL, SOC, SAC, PPO framework

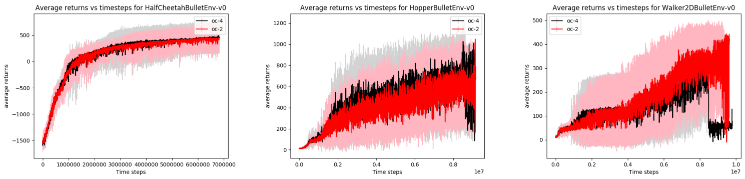

Figure 2: Average returns observed during training Option-critic framework

6 Experimental Setup and Hyperparameters

We use the PPO baseline implementation (Dhariwal et al. (2017)) provided by OpenAI for our experiments. For Soft Actor Critic, we use the implementation (Haarnoja (2018)) provided by the author. We use the same hyperparameters as that provided by the authors of SAC and PPO in their original papers for Mujoco tasks. We implement a deep version of the options-critic framework which uses advantage Actor Critic as its underlying RL algorithm. The architecture comprises of an inter-option Q-value network and an intra-option policy network. The Q-value network is implemented using a deep neural network with two hidden layers and tanh activation on the output of the first two layers. This network takes state vector appended with one hot encoded option as input and outputs the value function estimate of the option at a given state. The policy network shares the same architecture as Q-value network but takes state vector appended with one hot encoded option as input and outputs the mean and log of the standard deviation of the normal distribution over actions. The soft option critic is implemented using a deep neural network with 2 hidden layers as function approximators for inter-option Q value function, intra-option Q-value function, inter-option policy network, option termination function, and intra-option Q value function. The values of the hyperparameters used for options-critic and soft options critic is provided in the Appendix section.

7 Evaluation

Figure 1 shows the total average returns during training over 5 trials. The SOC framework, SAC and PPO have trained over 1 million steps and the policy is evaluated after every 2000 steps over 10 episodes. Results show that the SOC framework requires significantly less no of timesteps to achieve high returns as compared to options-critic framework. The framework also learns faster than soft actor critic and ppo in most tasks. Figure 2 shows that options-critic framework requires around 10 million steps to achieve the same order of returns as achieved by Soft Option critic in 1 million steps. Figure 3 shows a sample sequence of options used by SOC framework with and without regularization for solving the locomotion task in Ant environment after the training period. We observe temporal consistency in options selected by SOC framework with sparsity and variance regularization.

8 Conclusion

We have introduced a novel off-policy variant of options-critic framework based on maximum entropy framework and Deep Embedded Clustering for solving complex control tasks with high dimensional state and action space. Our experiments show that this framework outperforms options-critic, AdInfoHRL and other baselines like PPO and SAC.

9 Appendix

Lemma 1
Consider the Bellman backup operator T in Equation 6 and a mapping $Q^0 : S \times \Omega \times A \rightarrow R$ with $|A| < \infty$ and define $Q^{k+1} = TQ^k$. Then the sequence Q^k will converge to the soft Q-value of π as $k \rightarrow \infty$.
Proof:
We define the entropy augmented reward as

$$r_{\omega,\theta}(s_t, a_t) = r(s_t, a_t) + \mathbf{E}_{\mathbf{s_{t+1} \sim p}}[H(\pi_{\omega,\theta}(\cdot \mid s_{t+1}))]$$

and rewrite the update rule as:

$$Q_U(w_t, s_t, a_t) = r_{\omega,\theta}(s_t, a_t) + \gamma \mathbf{E}_{\mathbf{s_{t+1} \sim p}}[U(s_{t+1}, w_t)]$$

In the proof of intra-option q learning in Sutton et al 1998, it was proved that expected value of the update operator $r + \gamma U(s', w)$ yields a contraction ie-

$$[\mathbf{E}[r(s,a) + \gamma U(s', \omega)] - Q^*(s, w)] \leq \gamma \max_{s'', w''} \mid Q(s'', w'') - Q^*(s'', w'') \mid$$

Hence, we can define an evaluation operator TQ over intra-option Q value function space $S \times \Omega \times A$, with similar metric as that used for $Q(s, \omega)$:

$$T(Q_U(w_t, s_t, a_t)) = r_{\omega,\theta}(s_t, a_t) + \gamma \mathbf{E}_{\mathbf{s_{t+1} \sim p}}[U(s_{t+1}, w_t)]$$

Following analysis similar to that for $Q(s, \omega)$, we can show that the contraction holds.

References

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2016.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

BulletPhysics. Bullet. `https://github.com/bulletphysics/bullet3`, 2015.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *CoRR*, abs/1802.06070, 2018a. URL `http://arxiv.org/abs/1802.06070`.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018b.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies. *arXiv preprint arXiv:1811.07819*, 2018.

Tuomas Haarnoja. Soft actor critic. `https://github.com/haarnoja/sac`, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018b.

Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew G. Barto. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1162–1170. Curran Associates, Inc., 2010. URL `http://papers.nips.cc/paper/3903-constructing-skill-trees-for-reinforcement-learning-agents-from-demonstration-trajectories.pdf`.

George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *ICLR 2016*, 2015.

S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246, Oct 2012. doi: 10.1109/IROS.2012.6386006.

Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246. IEEE, 2012.

Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. Hierarchical reinforcement learning via advantage-weighted information maximization. *arXiv preprint arXiv:1901.01365*, 2019.

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

Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2015.