Attention-Over-Actions Option-Critic

Computer Science Extended Essay
Word Count: about 2971

Research Question

How can localized options be trained in the Option-Critic architecture?

Abstract

Option discovery in Reinforcement Learning has become quite popular recently. Since the proposal of Option-Critic, many follow up papers has focused on improving it, many of which ended up with a set of diverse and localized options. In this essay, I will propose a framework for algorithms to produce localized options, then base on the framework, I will derive a new algorithm that categorize sub-task based on the use of actions. This algorithm will be called Attention-Over-Actions Option-Critic because the action vector is Hadamard multiply with an attention mechanism.

1 Introduction

As human we operate in high level actions. For example when driving a car, we make decisions about turning left or right instead of thinking about which muscle to contract. We human have the ability to temporally abstract a chain of actions into one single high level action.

In Reinforcement Learning, we have a way to capture this idea of temporally abstract action by using options [1]. When options are defined, learning the policy-over-options are very straight forward, by using SMDP Q-Learning [2] or Intra-Option Learning [3]. However, if the options are not given and need to be learned, things get a lot harder since it requires knowing what makes an option good.

Many people argue that a good options should be diverse and localized [4]. One of the recent papers [5] follows this assumption by requiring options to attend to different features of the state. This essay focus on extending this idea of abstraction from state to action.

This essay will be structured as follows: First, preliminary and related work will be presented to give a context to what I am trying to do. Second, previous work will be analyzed to figure out why they work. Third, a framework will be proposed based on the observations made in the analysis. Forth, an algorithm will be derived from the framework. Finally, the algorithm will be evaluated in Four Room and the Mujoco environment [6].

2 Preliminary

This section only acts as a summary. You are assumed to have basic knowledge about Hierarchical Reinforcement Learning.

Markov Decision Process

Markov decision process (MDP) [7] is a mathematical framework for modeling decision making in a stochastic environment. It is defined as a tuple: $< \mathcal{S}, \mathcal{A}, r, \gamma, P >$ where:

$\mathcal{S}$ is the set of states.

$\mathcal{A}$ is the set of actions

$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function

$\gamma \in [0, 1)$ is the discount factor that ensure the cumulative reward $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ converges

$P : \mathcal{S} \times A \times \mathcal{S} \to [0, 1]$ is the transition model which gives the probability for a particular transition to occur.

All MDPs are said to have the Markov Property, which means that $P(s, a, s')$ only depends on the current state and action. In an MDP, A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is responsible for choosing the action, after an action is chosen, the environment transitions to a new state according to $P(s, a, s')$, also a reward is given to the policy, and the cycle continues until the policy arrives a termination state.

Reinforcement Learning

Reinforcement Learning (RL) [7] is a machine learning paradigm which allows an agent to learn from interaction in an MDP. The agent's goal is to maximizes long term return $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

Actor-Critic [8] is one of the popular classes of algorithms that harvest the advantage of both Q-Learning and Policy Gradient. A critic network is trained to minimize the one step TD error $r+\gamma \max_{a'} Q(s', a')-Q(s, a)$, while an actor network simultaneously takes gradient step $Q(s, a)\nabla \log \pi(a|s)$.

Option Framework

Options [1] are temporally extended actions. They are defined as a tuple: $< \mathcal{I}_\omega, \pi_\omega, \beta_\omega >$, where:

$\mathcal{I}_\omega \subseteq \mathcal{S}$ is the initiation set that define which state the option can be selected

$\pi_\omega : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the internal policy

$\beta_\omega : \mathcal{S} \to [0, 1]$ is the termination probability.

In an MDP, a policy-over-options $\pi_\Omega : \mathcal{S} \times \Omega \to [0, 1]$, where Ω is the set of options, selects one of the options to execute, then actions are chosen by the current option's π_ω from then on, until the option terminated according to its β_ω, then π_Ω selects an option again. An option has a chance to terminate after an action is executed. Options can be viewed as a generalization for actions, since an option can become an action by setting $\mathcal{I}_\omega = \mathcal{S}$, $\pi_\omega(a|s) = 1$ and $\beta_\omega(s) = 1$ everywhere.

The policy-over-options π_Ω can be derived by first training Q_Ω using SMDP Q-Learning [2] $Q(s, \omega) \leftarrow Q(s, \omega) + \alpha(r + \gamma^k \max_{\omega'} Q(\omega', \omega') - Q(s, \omega))$ or Intra-Option Learning [3] $Q(s, \omega) \leftarrow Q(s, \omega) + \alpha(r + \gamma((1 - \beta(s'))Q(s', \omega) + \beta(s') \max_{\omega'} Q(s', \omega')) - Q(s, \omega))$, then make the π_Ω greedy policy of Q_Ω.

3 Related Work

In this section, some of the other option discovery algorithms will be summarized.

Option-Critic

Option-Critic [9] is an RL algorithm inspired by Action-Critic [8]. termination functions β_ω and option-policies π_ω takes gradient steps to maximize expected return, while a option-value function Q_Ω take gradient steps to minimize error between itself and the empirical return.

Deliberation Cost

Since optimal policy can be achieved even without using options, if options are trained only to maximize expected return, the options may slowly degenerate, which means they either terminate every steps or dominate. Deliberation Cost [10] is a way to encourage longer option duration by punishing the option for every termination.

Interest Option-Critic

The original Option-Critic assumes that options can be initiated everywhere, Interest Option-Critic [4] tries to remove this assumption by introducing interest functions $I : \mathcal{S} \times \Omega \to [0, 1]$ as an approximation for the initiation set. Experimental results show that options learned by Interest Option-Critic is localized.

Termination-Critic

While other work in this section focus on improving Option-Critic by adding features to the algorithm, Termination-Critic [11] aims to change the objective of the termination function β. Instead of maximizing the expected return, β is trained to minimize the entropy of the termination state. In other words, making the termination state more predictable.

Attention Option-Critic

Attention Option-Critic [5] implements attention mechanism into Option-Critic. Different options are trained to attend to different features of the state. The attention units were trained to not only maximize the expected return, but also other things like maximizing difference between attention of different options.

4 Exploration

An analysis on localization will be conducted in this section.

What is Localization?

Localization is about options each responsible for a sub-task, or another way of looking at it is options each representing a skill. For example in environments like Four Rooms or T-maze, each localized options will be used in a small area of the 2D space. However, in environments where agent can move further in the state space per step, such as visual inputs, each localized options will be used in states that are near temporally. One of the signs for localization is having long options duration. However, defining and measuring localization quantitatively is hard, which is why most work evaluate these option discovery algorithms qualitatively, by observing the agent acting for an episode in the environment.

Why Localization?

To understand why we want localization, first we need to answer a fundamental question: Why do we even use options in the first place? Is it to maximize expected return? However, optimal policy can be achieved using only primitive actions. If options cannot give us a higher return, why do we even need options? Some researchers suggest that options should speed up planning [10] [11] and also options should be transferable [4] [5]. If this is what a good option should be like, then a set of localized options would be beneficial. Localized options are more interpretable, which makes it easily reusable when transferred to a different environment. Also, interpretable options can speed up planning because each options have its clear purpose and usage.

How Localization is achieved?

Now I will analyze how some of the previous work achieve localization of options.

Attention Option-Critic

In Attention Option-Critic [5], each options are trained to attend to different features of the state. My hypothesis is that the attention mechanism can act as a constraint on what kind of policy each option can have. Each features of the state represents a piece of information about the state. When performing a sub-task, not all the features are necessary. Each sub-task requires different subset of features. Since the attention mechanism limits the subset of features given to an option, the option cannot learn sub-task that requires features outside of the subset of features it was given, or else the option will perform poorly. In the algorithm, each option is trained to have diverse attention, which force each option to learn to complete a different sub-task. For example, there is 3 options and an RGB 2D image is the features of the state. Suppose the 3 options each attend to one of the RGB channels, and one of the sub-tasks is checking if there is a red circle on the image. In this case, only the option with attention on the red channel can complete this sub-task.

Deliberation Cost

In Deliberation Cost [10], options are encouraged to be more temporally extended. My hypothesis is that the algorithm does not directly make options more localized, it instead eliminates all options that are not localized for sure. Since options that terminate every steps are not localized for sure, adding a Deliberation Cost can eliminates all of these options, which makes training localized options more probable.

Termination-Critic

In Termination-Critic [11], option termination states' entropy is being minimized, and experimental results show that option trained by this usually choose to terminate in bottleneck states (frequently visited states). My hypothesis is that bottleneck states are usually the start or end of a sub-task, having the option terminate at these states essentially chains termination with initiation. I will illustrate this with a simple example: In the Four Rooms environment, assume that the sub-task is walking from one doorway to another. The two doorway are bottleneck states because the agent must go through them. Since the agent can take on many paths, all the other states are not bottleneck states. When the agent get to the next doorway, another option can be immediately initiated.

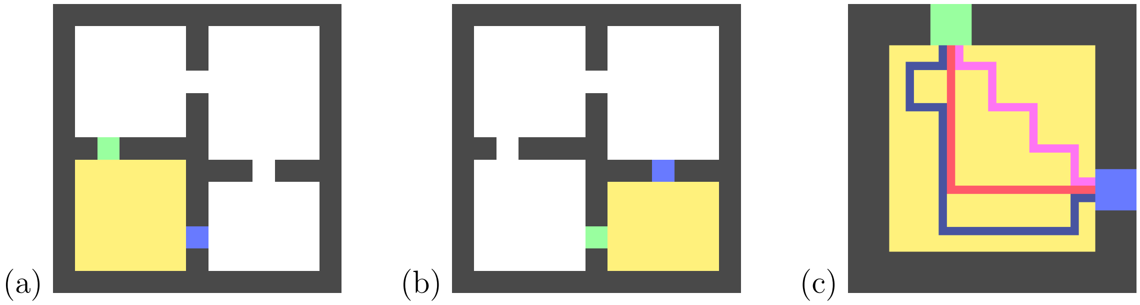

Figure 1: (a) and (b) are two options in the theoretic example. Green is the start of the option, blue is the end of the option, yellow is the intermediate states. (c) is a zoomed-in version of the bottom left room. Dark blue, red and pink lines are some of the paths the option can take.

Interest Option-Critic

To me Interest Option-Critic just seems like magic. Firstly, it is trained to maximize total return only, but it still achieved localization. Secondly, in the code, the policy-over-options is also trained with gradient ascent, which makes the policy-over-options the same as the interest function, except that one is softmax and one is sigmoid. My best guess is that the interest function provides an easier way to learn localized options. The combination of sigmoid and softmax function is easier to achieve localization than just a softmax function. And according to the experimental results in the original paper, Interest Option-Critic actually perform slightly better than Option Critic, which means that there is a chance that localized option can have a higher return. So Interest Option-Critic can and have the motivation to achieve localized option while Option-Critic may stuck in local maximum.

5 The Localization Framework

Naturally, the next question that will be asked is: What do all of these algorithms have in common? Now I will propose a frame work that can act as an abstraction for all of these algorithms.

The Localization Framework

1. Categorization
Divide the whole task into meaningful sub-tasks based on a certain criterion
2. Assignment
Assign the divided sub-tasks to different options
3. Optimization
Train options to perform well in the sub-task it is given and also improve the initial categorization of sub-tasks
4. Selection
Form the policy-over-options to select option in different situation

This framework is inspired by Adaboost [12], which is an Ensemble Learning algorithm from Supervised Learning. There are a lot of similarities between Ensemble Learning and Option Learning, this has already been pointed out in previous work [13], the individual weak classifiers can be thought of as options, each weak classifier is responsible for classifying a small subset of the training data, just like how each option is responsible for a sub-task. In Adaboost, a bunch of weak classifiers are trained sequentially, each of them focuses on training data that is classified poorly by the previous weak classifiers. Since this training process can be seen as dividing training example into category, then assign it to different weak classifiers, it inspires the Categorization and Assignment steps in the Localization Framework. Also, the Optimization step in the framework is reminiscent of the weak classifiers learning to classify the training examples. After a lot of weak classifiers are trained, Adaboost combines them together to form a boosted classifier. The boosted classifier is the weighted sum of all the weak classifiers, the weighting is somewhat like a selection process, so it inspires the Selection step in the framework.

	Categorization	**Assignment**	**Optimization**	**Selection**
Deliberation Cost	Divide states based on their temporal closeness	The deliberation cost is put on the termination function of each option	Option minimize deliberation cost and maximize return, the temporal categorization is very vague and can be changed easily	Choose the option with maximum expected return including deliberation cost
Attention Option-Critic	Divide sub-tasks based on the features they need from state	The algorithm assign an attention mechanism to each option	Option maximize return, the attention mechanism maximize return and also difference between attention	Choose the option with maximum expected return
Termination-Critic	Divide sub-task based on bottleneck states	The termination functions are trained to terminate in bottleneck states	Internal policy maximize return, termination function minimize entropy	Choose the option with maximum expected return

Table 1: All previous algorithms can be fitted into the Localization Framework, except for Interest Option-Critic because I still couldn't really figure out why it works

All of the previous algorithms can be seen as an implementation of this framework. This framework is the abstraction of algorithms that produce localized options, so it is very useful in deriving a new algorithm in the next section.

6 Attention-Over-Actions Option-Critic

Now that there is a framework, I can just follow the framework and derive a new algorithm. The following algorithm will be called Attention-Over-Actions Option-Critic because it perform abstraction on the action space, this algorithm is largely inspired by Attention Option-Critic.

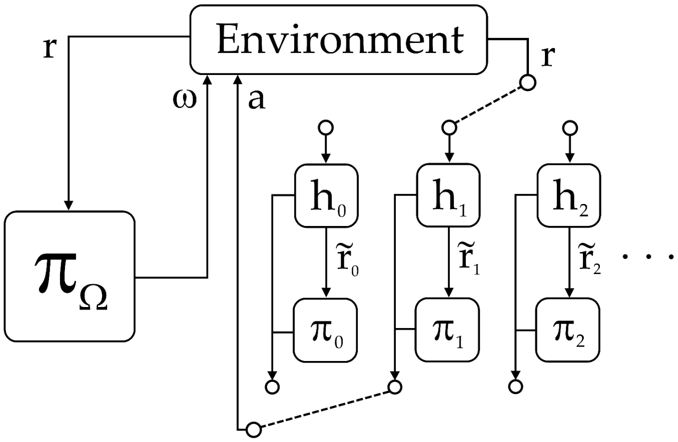

Figure 2: Visualization of the interaction between the environment and the Attention-Over-Actions Option-Critic algorithm. Note that the state is ignored here for clearer presentation.

1. Categorization

In Attention Option-Critic, the Categorization step divide sub-tasks based on features needed, this works because each sub-task requires a different subset of features. The following algorithm borrow the idea of attention over state features and apply it to actions, so each option will attend to different sets of actions. The intuition for this is that each sub-task will not need to use all the actions, for example, when you are watching a film, you will not need to move your hands. This is essentially performing actions abstraction, since the option can attend to a subset of the actions.

2. Assignment

The following algorithm assigns the sub-task in a similar way as Attention Option-Critic, an attention mechanism $h_{\omega,\phi} : \mathcal{A} \to [0,1]$ parameterized by ϕ will be given to each option. But instead of masking the state features, it masks the probability for choosing each actions. So the final probability π_{h_ω} for option ω to choose action a will be:

$$\pi_{h_\omega}(a|s) = \frac{\pi_{\omega,\theta}(a|s)h_{\omega,\phi}(a)}{\sum_{a'} \pi_{\omega,\theta}(a'|s)h_{\omega,\phi}(a')}$$

where $\pi_{\omega,\theta}$ is the internal policy of option ω and is parameterized by θ.

π_{h_ω} can also be expressed in matrix form:

$$\pi_{h_\omega}(s) = \frac{\pi_{\omega,\theta}(s) \odot h_{\omega,\phi}}{\pi_{\omega,\theta}(s)^\intercal h_{\omega,\phi}}$$

Where all of these are column matrix with each component corresponding to one action, and $\odot$ is the Hadamard product.

3. Optimization

Optimization is probably the step that requires the most decision made when designing this algorithm, but luckily optimization of the attention mechanism and that of the option can designed separately.

Option Optimization

Let's first consider the optimization of the option. Normally, when optimizing the internal policy, we will perform gradient ascent on the return:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta Q_\Omega(s, \omega)$$

where $Q_\Omega(s, \omega)$ is the expected return for choosing ω in s

I can now directly reuse the result from the Option-Critic paper, and only change $\pi_{\omega,\theta}$ to π_{h_ω}:

$$\nabla_\theta Q_\Omega(s, \omega) = E[\nabla_\theta \pi_{h_\omega}(a|s) Q_\omega(s, \omega, a)]$$

However, this is not desirable because the behavior of the optimal policy would not change, any policy can be achieved by simply dividing the attention mechanism and then normalize:

$$\widetilde{\pi}_{\omega,\theta}(a|s) = \frac{\pi_{\omega,\theta}(a|s)}{h_{\omega,\phi}(a) \sum_{a'} (\pi_{\omega,\theta}(a'|s)/h_{\omega,\phi}(a'))}$$

What I really want is for the option to accept the attention mechanism, instead of fighting against it. I want the option to find the next best action provided that there is a constraint. So a better way to optimize the internal policy is to devalue the Q-value of the unattended actions, this way the option will not know that it can actually cheat the system by dividing the attention mechanism. One way to devalue the Q-value is to punish the policy for choosing an action that the option is not attending to. Just like Deliberation Cost, the reward r is transformed into a pseudo-reward $\widetilde{r}_\omega$:

$$\widetilde{r}_\omega(s, a) = r(s, a) - \rho_\omega(a)$$

It seems to me that the choice of the punishment ρ_ω can be arbitrary, as long as it is large when $h_{\omega,\phi}$ is small. So I will just choose it like this:

$$\rho_\omega(a) = \xi R(1 - h_{\omega,\phi}(a)^n)$$

$$\text{where } R = \max_{s,a} r(s, a) - \min_{s,a} r(s, a)$$

$$\text{and } n \geq 1 \geq \xi > 0$$

There is some intuition behind this definition. n is for making sure that all actions with low attention will be treated almost the same, while actions with high attention will be treated differently even for a small difference. R is an attempt to scale the punishment with the reward, so if all the rewards are suddenly doubled, the punishment will still have the same effect. ξ can be interpreted as an exploratory constant, because this value scales the punishment, which can also be interpreted as the motivation to try actions that it is not attending to. This punishment ρ will be called the Distraction because it is high when the option chooses actions it is not attending to.

Now we can learn expected return after choosing an action $\widetilde{Q}_U$ and expected return after choosing an option $\widetilde{Q}_\Omega$ to approximate the expected pseudo-return using Q-Learning. After we have the Q-values, we can train $\pi_{\omega,\theta}$ and $\beta\omega, \nu$ just like in the original Option-Critic.

Attention Mechanism Optimization

Now let's consider the optimization of $h_{\omega,\phi}$. I want the categorization to be different for all options because or else all options will just aim for the sub-task with highest return. I also want the options to focus on as little actions as possible while still having acceptable performance. Essentially what I need is the algorithm to consider the trade-off between these objectives and achieve a balance between them. A nice way to do this is to add all of these objective up and then perform gradient ascent on the sum:

$$\phi \leftarrow \phi + \alpha_\phi \nabla_\phi \sum_o (w_o O_o)$$

where o is the index of an objective, w_o is the weight of the objective, O_o is the objective function. This method has been used for Attention Option-Critic too. Now I will list out the objectives that I want the option to consider:

1. Perform well
2. Different from other options
3. The components of the attention mechanism is close to 0 or 1
4. Focus on small set of actions

For the first objective, I can just use $Q_\Omega(s, \omega)$ like in Attention Critic. Note that this objective uses the real Q-value instead of the pseudo-Q-value.

$$O_1 = \max_h Q_\Omega(s, \omega)$$

For the second objective, I will minimize cosine similarity just like in Attention Critic. Cosine similarity is desirable because magnitude doesn't matter, only the angle does. A small change in a short vector will correspond to a large change in the angle. While small change in a long vector will not change the angle much. Which means that the cosine similarity is more sensitive to large similar components. This is what we want because when we are comparing two attention mechanism, we only do not want both to have high attention on the same action, we don't care if both have low attention on the same action.

$$O_2 = \min_h \sum_{h' \neq h} \frac{< h', h >}{||h'|| + ||h||}$$

For the third objective, I will minimize entropy in the attention mechanism. Entropy measures the uncertainty in a probability distribution, so h must be normalized first. This actually also slightly optimize the forth objective since only attending to 1 actions have the lowest uncertainty.

$$O_3 = \max_h < \frac{h}{||h||}, \log \frac{h}{||h||} > \text{ where log is elementwise}$$

For the forth objective, I will minimize the Minkowski distance of the attention mechanism. This is similar to minimizing the length of the attention mechanism, which discourage focusing on too many actions. Having $p < 1$ is for making small values more prominent, which slightly optimize the third objective.

$$O_4 = \min_h ||h|| \text{ where } || \times || \text{ is Minkowski distance with } p < 1$$

4. Selection

Any policy-over-options that favor higher Q-value options will work in this case, because the option will need the right set of actions in order to perform well, or else it will fail horribly. So the Q-value already encoded which option has the right set of actions. This means that policies like ϵ-Greedy should work for this algorithm.

Intuition for this algorithm

Option Framework is very similar to a construction site, options are the workers and actions are the tools(e.g. Hammer or Shovel). In this algorithm, each option pay attention to only a small set of actions and learn a policy that use them well. This can be drawn parallel with workers only focusing on learning a few tools and master them. Each worker will be responsible for a sub-task in a construction project, in other words, there will be division of labour.

In this algorithm, the components in the attention mechanism are not binary and can take on any real numbers between 0 and 1. In the context of construction site, there are two interpretation for a non-integer focus on a tool. First it can be interpreted as the worker not trying hard enough when learning the tool. Interestingly, this interpretation is parallel to the first method suggested in the Optimization section: $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta Q_\Omega(s, \omega)$. The worker have the incentive to learn a good tool even if he is not focusing on it, just like how the option have the incentive to use a good action even if it is not attending to it. The second interpretation is that the worker is given a weaker tool, this interpretation is parallel to the punishment method suggested in the Optimization section: $\widetilde{r}_\omega(s, a) = r(s, a) - \rho_\omega(a)$. The workers will not have the incentive to learn a tool that they are not focusing on, even if the normal version of the tool can bring a higher wage, because the worker is only given a weaker version of the tool.

One of the objectives that the attention mechanism need to optimize is the global Q-value $Q_\Omega(s, \omega)$, while the option's internal policy optimizes a local Q-value $\widetilde{Q}_\Omega(s, \omega)$. This can be drawn parallel with the manager in a construction project, since the manager usually see things from a broader perspective.

Algorithm 1 Pseudocode for Attention-Over-Actions Option-Critic (AOAOC)

```
$s \leftarrow s_0$
Choose $\omega$ according to the policy-over-options $\pi_\Omega(s)$
**repeat**
    Choose $a$ according to $\pi_{h_\omega}(a|s)$
    Take action $a$ in $s$, observe $s'$, $r$

    **1. Options evaluation:**
    $\widetilde{r}_\omega \leftarrow r - \rho_\omega(a)$
    $\widetilde{\delta} \leftarrow \widetilde{r}_\omega - \widetilde{Q}_U(s, \omega, a)$
    $\delta \leftarrow r - Q_U(s, \omega, a)$
    **if** $s'$ is non-terminal **then**
        $\widetilde{\delta} \leftarrow \widetilde{\delta} + \gamma(1 - \beta_\omega(s'))\widetilde{Q}_\Omega(s', \omega) + \gamma\beta_\omega(s') \max_{\omega'} \widetilde{Q}_\Omega(s', \omega')$
        $\delta \leftarrow \delta + \gamma(1 - \beta_\omega(s'))Q_\Omega(s', \omega) + \gamma\beta_\omega(s') \max_{\omega'} Q_\Omega(s', \omega')$
    **end if**
    $\widetilde{Q}_U(s, \omega, a) \leftarrow \widetilde{Q}_U(s, \omega, a) + \widetilde{\alpha}\widetilde{\delta}$
    $Q_U(s, \omega, a) \leftarrow Q_U(s, \omega, a) + \alpha\delta$

    **1. Options improvement:**
    $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \log \pi_{h_\omega}(a|s)\widetilde{Q}_U(s, \omega, a)$
    $\nu \leftarrow \nu + \alpha_\nu \nabla_\nu \beta_{\omega,\nu}(s')(\widetilde{Q}_\Omega(s', \omega) - \widetilde{V}_\Omega(s'))$
    $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi \sum_o (w_o O_o)$
    **if** $\beta_{\omega,\nu}$ terminates in $s'$ **then**
        choose new $\omega$ according to the policy-over-options $\pi_\Omega(s)$
    **end if**
    $s \leftarrow s'$
**until** $s'$ is terminal
```

7 Evaluation

8 Conclusion

9 Future Direction

This algorithm currently only focuses on discrete action space, a possible future direction might be extending this to continuous space. For vector actions, it might be good to attend to only some of the components of the vector, while ignoring other components by either randomly selecting values for them or keeping them the same as in the last state of the previous option. For scalar action, one direction is to use a one dimensional gaussian distribution as the attention mechanism, and multiply it with the original action distribution. However, some sort of trick may need to be deployed to speed up the process of normalization.

10 Bibliography

[1] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181 – 211, 1999.

[2] S. Bradtke and M. Duff, "Reinforcement learning methods for continuous-time markov decision problems," 12 1994.

[3] R. Sutton, D. Precup, and S. Singh, "Intra-option learning about temporally abstract actions.," pp. 556–564, 01 1998.

[4] K. Khetarpal, M. Klissarov, M. Chevalier-Boisvert, P.-L. Bacon, and D. Precup, "Options of interest: Temporal abstraction with interest functions," 2020.

[5] R. Chunduru and D. Precup, "Attention option-critic," 2020.

[6] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[7] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning series, MIT Press, 2018.

[8] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," 2000.

[9] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," 2016.

[10] J. Harb, P.-L. Bacon, M. Klissarov, and D. Precup, "When waiting is not an option : Learning options with a deliberation cost," 2017.

[11] A. Harutyunyan, W. Dabney, D. Borsa, N. Heess, R. Munos, and D. Precup, "The termination critic," 2019.

[12] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory* (P. Vitányi, ed.), (Berlin, Heidelberg), pp. 23–37, Springer Berlin Heidelberg, 1995.

[13] S. Zhang, H. Chen, and H. Yao, "Ace: An actor ensemble algorithm for continuous control with tree search," 2018.

11 Appendix

A Notations

Markov Decision Process
$\mathcal{S}$ — Set of states
$\mathcal{A}$ — Set of actions
r — Reward function or reward
γ — Discount factor
P — Transition model
π — Policy
s — State
a — Action
s_T — Termination state
Option Framework

Ω — Set of options
$\pi_{\omega,\theta}$ — Internal policy of option ω
β_ω — Termination probability of option ω
$\mathcal{I}_\omega$ — Initiation set of option ω
π_Ω — Policy-over-options
Q_U — Expected return for choosing an action
Q_Ω — Expected return for choosing an option
θ — Parameter for internal policy
ν — Parameter for termination probability
ω — option
Attention-Over-Actions Option-Critic
$h_{\omega,\phi}$ — Attention Mechanism
π_{h_ω} — Final probability
ϕ — Parameter for attention mechanism
ρ — Distraction
R — Difference between max and min reward
n — Power constant in ρ
ξ — Exploratory constant in ρ
$\widetilde{r}$ — Pseudo-reward
$\widetilde{Q}_U$ — Expected pseudo-return from choosing an action
$\widetilde{Q}_\Omega$ — Expected pseudo-return from choosing an option
$\widetilde{V}_\Omega$ — Expected pseudo-return for being in a state
$\widetilde{\alpha}$ — Learning rate for $\widetilde{Q}_U$
α — Learning rate for Q_U
α_θ — Learning rate for internal policy
α_ν — Learning rate for termination probability
α_ϕ — Learning rate for attention mechanism
R — Minimum reward clipped by 0
O_o — Objective function
w_o — Weight of objective function
o — Objective
δ — One step Q-value error
$\widetilde{\delta}$ — One step pseudo-Q-value error
Operations
$\odot$ — Hadamard product (elementwise product)
∇ — Gradient
$^\intercal$ — Matrix transpose
$E[\ ]$ — Expected value
$\leftarrow$ — Assignment
$<,>$ — Dot product
$||.||$ — Norm of a vector unless specify

B Proof

B.1 Derivative of Objective 1

Let h_ω^a be the element in the attention vector h_ω that corresponds to the action a, i.e. $h_\omega = [h_\omega^{a_0}, h_\omega^{a_1}, ...]$

$$\frac{\partial Q_\Omega(s,\omega)}{\partial h_\omega^a} = \sum_{s,\omega} \mu_\Omega(s,\omega|s_0,\omega_0) \sum_a \frac{\partial \pi_{h_\omega}(a|s)}{\partial h_\omega^a} Q_U(s,\omega,a)$$

$$= E_{s,\omega,a\sim\pi_{h_\omega}}[\frac{\partial \log \pi_{h_\omega}(a|s)}{\partial h_\omega^a} Q_U(s,\omega,a)]$$

The steps above follows directly from the Option-Critic appendix. The next step is to express $\frac{\partial \log \pi_{h_\omega}(a|s)}{\partial h_\omega^a}$ in a simpler form.

$$\frac{\partial \log \pi_{h_\omega}(a|s)}{\partial h_\omega^a} = \frac{\partial}{\partial h_\omega^a}[\log \frac{\pi_\omega(a|s) h_\omega^a}{\sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}}]$$

$$= \frac{\partial}{\partial h_\omega^a}[\log \pi_\omega(a|s) + \log h_\omega^a - \log \sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}]$$

$$= \frac{\partial}{\partial h_\omega^a}[\log \pi_\omega(a|s)] + \frac{\partial}{\partial h_\omega^a}[\log h_\omega^a] - \frac{\partial}{\partial h_\omega^a}[\log \sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}]$$

$\pi_\omega(a|s)$ is independent of h_ω^a, hence $\frac{\partial}{\partial h_\omega^a}[\log \pi_\omega(a|s)] = 0$

$$= \frac{\partial}{\partial h_\omega^a}[\log h_\omega^a] - \frac{\partial}{\partial h_\omega^a}[\log \sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}]$$

$$= \frac{1}{h_\omega^a} - \frac{\pi_\omega(a|s)}{\sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}}$$

$$= \frac{1}{h_\omega^a} - \frac{\pi_\omega(a|s)}{\sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}} \times \frac{h_\omega^a}{h_\omega^a}$$

By definition $\pi_{h_\omega}(a|s) = \frac{\pi_\omega(a|s) h_\omega^a}{\sum_{a'} \pi_\omega(a'|s) h_\omega^{a'}}$

$$= \frac{1}{h_\omega^a} - \frac{\pi_{h_\omega}(a|s)}{h_\omega^a}$$

$$= \frac{1 - \pi_{h_\omega}(a|s)}{h_\omega^a}$$

Substitute back to the original calculation,

$$\frac{\partial Q_\Omega(s,\omega)}{\partial h_\omega^a} = E_{s,\omega,a \sim \pi_{h_\omega}}[\frac{1 - \pi_{h_\omega}(a|s)}{h_\omega^a} Q_U(s,\omega,a)]$$

B.2 Derivative of Objective 2

Let h_ω^a be the element in the attention vector h_ω that corresponds to the action a, i.e. $h_\omega = [h_\omega^{a_0}, h_\omega^{a_1}, ...]$ Since we would like to minimize the cosine similarity, the negative cosine similarity will be used instead.

$$\sum_{h_{\omega'} \neq h_\omega} \frac{\partial}{\partial h_\omega^a} \frac{< h_\omega, h_{\omega'} >}{||h_\omega|| \times ||h_{\omega'}||} = \sum_{h_{\omega'} \neq h_\omega} \frac{||h_\omega|| \times ||h_{\omega'}|| \frac{\partial}{\partial h_\omega^a} < h_\omega, h_{\omega'} > - < h_\omega, h_{\omega'} > \frac{\partial}{\partial h_\omega^a} ||h_\omega|| \times ||h_{\omega'}||}{(||h_\omega|| \times ||h_{\omega'}||)^2}$$

I will calculate each of the derivative separately:

$$\frac{\partial}{\partial h_\omega^a} < h_\omega, h_{\omega'} >= \frac{\partial}{\partial h_\omega^a} \sum_{a'} h_\omega^{a'} h_{\omega'}^{a'} = h_{\omega'}^a$$

$$\frac{\partial}{\partial h_\omega^a} ||h_\omega|| \times ||h_{\omega'}|| = ||h_{\omega'}|| \frac{\partial}{\partial h_\omega^a} \sqrt{\sum_{a'} (h_\omega^{a'})^2} = ||h_{\omega'}|| \frac{h_\omega^a}{\sqrt{\sum_{a'} (h_\omega^{a'})^2}}$$

Substitute back to the original calculation,

$$\sum_{h_{\omega'} \neq h_\omega} \frac{\partial}{\partial h_\omega^a} \frac{< h_\omega, h_{\omega'} >}{||h_\omega|| \times ||h_{\omega'}||} = \sum_{h_{\omega'} \neq h_\omega} \frac{||h_\omega|| \times ||h_{\omega'}|| h_{\omega'}^a - < h_\omega, h_{\omega'} > ||h_{\omega'}|| \frac{h_\omega^a}{\sqrt{\sum_{a'} (h_\omega^{a'})^2}}}{(||h_\omega|| \times ||h_{\omega'}||)^2}$$

$$= \sum_{h_{\omega'} \neq h_\omega} \frac{h_{\omega'}^a}{||h_\omega|| \times ||h_{\omega'}||} - \frac{< h_\omega, h_{\omega'} > h_\omega^a}{(||h_\omega|| \times ||h_{\omega'}||) \times ||h_\omega||^2}$$

B.3 Derivative of Objective 3

Let h^a_ω be the element in the attention vector h_ω that corresponds to the action a, i.e. $h_\omega = [h_\omega^{a_0}, h_\omega^{a_1}, ...]$ Since entropy usually applies to probabilities, I will normalize the attention unit into $\overline{h_\omega} = \frac{h_\omega}{\sum_{a'} h_\omega^{a'}}$

$$\frac{\partial H(\overline{h_\omega})}{\partial h^a_\omega} = \frac{\partial \sum_{a'} \overline{h^{a'}_\omega} \log \overline{h^{a'}_\omega}}{\partial h^a_\omega}$$

$$= \frac{\partial \overline{h^a_\omega} \log \overline{h^a_\omega}}{\partial h^a_\omega} + \sum_{\overline{h^{a'}_\omega} \neq \overline{h^a_\omega}} \frac{\partial \overline{h^{a'}_\omega} \log \overline{h^{a'}_\omega}}{\partial h^{a'}_\omega}$$

$$= \frac{\partial \overline{h^a_\omega} \log \overline{h^a_\omega}}{\partial \overline{h^a_\omega}} \times \frac{\partial \overline{h^a_\omega}}{h^a_\omega} + \sum_{\overline{h^{a'}_\omega} \neq \overline{h^a_\omega}} \frac{\partial \overline{h^{a'}_\omega} \log \overline{h^{a'}_\omega}}{\partial \overline{h^{a'}_\omega}} \times \frac{\partial \overline{h^{a'}_\omega}}{h^{a'}_\omega}$$

I will calculate each of the derivative separately:

$$\frac{\partial \overline{h^a_\omega} \log \overline{h^a_\omega}}{\partial \overline{h^a_\omega}} = \log \overline{h^a_\omega} \frac{\partial \overline{h^a_\omega}}{\partial \overline{h^a_\omega}} + \overline{h^a_\omega} \frac{\partial \log \overline{h^a_\omega}}{\partial \overline{h^a_\omega}} = \log \overline{h^a_\omega} + 1$$

$$\frac{\partial \overline{h^a_\omega}}{\partial h^a_\omega} = \frac{\partial}{\partial h^a_\omega} \frac{h^a_\omega}{\sum_b h^b_\omega} = \frac{\sum_b h^b_\omega - h^a_\omega}{(\sum_b h^b_\omega)^2}$$

$$\frac{\partial \overline{h^{a'}_\omega}}{\partial h^a_\omega} = \frac{\partial}{\partial h^a_\omega} \frac{h^{a'}_\omega}{\sum_b h^b_\omega} = \frac{-h^{a'}_\omega}{(\sum_b h^b_\omega)^2}$$

Substitute back to the original calculation,

$$\frac{\partial \overline{h^a_\omega} \log \overline{h^a_\omega}}{\partial \overline{h^a_\omega}} = (\log \overline{h^a_\omega} + 1) \frac{\sum_b h^b_\omega - h^a_\omega}{(\sum_b h^b_\omega)^2} + \sum_{\overline{h^{a'}_\omega} \neq \overline{h^a_\omega}} (\log \overline{h^{a'}_\omega} + 1) \frac{-h^{a'}_\omega}{(\sum_b h^b_\omega)^2}$$

$$= \frac{(\log \overline{h^a_\omega} + 1)}{\sum_b h^b_\omega} + \sum_{h^{a'}_\omega} (\log \overline{h^{a'}_\omega} + 1) \frac{-h^{a'}_\omega}{(\sum_b h^b_\omega)^2}$$

B.4 Derivative of Objective 4

Let h^a_ω be the element in the attention vector h_ω that corresponds to the action a, i.e. $h_\omega = [h_\omega^{a_0}, h_\omega^{a_1}, ...]$ Since we would like to minimize the length, the negative length will be used instead.

$$\frac{\partial ||h_\omega||}{\partial h^a_\omega} = \frac{\partial}{\partial h^a_\omega} \sqrt{\sum_{a'} (h^{a'}_\omega)^2}$$

$$= \frac{h^a_\omega}{\sqrt{\sum_{a'} (h^{a'}_\omega)^2}} = \frac{h^a_\omega}{||h_\omega||}$$

C Experimental Details

D Code