

1609.05140 - The Option-Critic Architecture

The Option-Critic Architecture

Pierre-Luc Bacon and **Jean Harb** and **Doina Precup**

Reasoning and Learning Lab, School of Computer Science

McGill University

{pbacon, jharb, dprecup}@cs.mcgill.ca

- **Yunqiu Xu**
 - Overview:
 - End-to-end HRL
 - Only need to specify the number of options
 - Do not need additional reward function or sub-goals
 - Option is similar to sub-task
-

1. Introduction

- Why use temporal abstraction: can represent knowledge about action courses that happen at different time scales
- Classical option framework (**Sutton 1999, Precup 2000**) : 定义不同time scales的 action courses, 用于无缝 learning and planning
 - Markovian option $\omega \in \Omega = (I_\omega, \pi_\omega, \beta_\omega)$
 - $I_\omega \in \mathcal{S}$: initiation state set
 - π_ω : intra-option policy, similar to sub-task, but no need for inner reward function
 - $\beta_\omega : \mathcal{S} \rightarrow [0, 1]$: termination function, take a state as input, output whether this option is ended
 - **Assumption** $\forall s \in \mathcal{S}, \forall \omega \in \Omega : s \in I_\omega$, i.e. every option can be used everywhere, i.e. 对任意状态, 不存在不可用的选项, 无非是分数可能差点

- **思考: 训练termination function会不会带来额外的bias**
- Challenges for existing work:
 - Learning with subgoals maybe expensive
 - 学习单个任务很慢: 需要重用similar tasks中学到的options, 然而因为单个任务没什么与之相似的任务, 学习速度很慢
- Our work: option-critic
 - 同时学习intra-option policies, termination functions and over-option policy
 - 仅仅需要制定number of desired options
 - 不需要subgoals, additional rewards以及demonstrations
 - 也可以加入额外的reward

2. Learning Options

- 对于某个状态选择某个option $\omega \in \Omega$ 的Q函数

$$Q_{\Omega}(s, \omega) = \sum_a \pi_{\omega, \theta}(a | s) Q_U(s, \omega, a) \quad , \quad (1)$$

- 这个是master选择option的Q函数, 状态 s 已经确定, 选择哪个 ω 需要我们进行评估
- 对于某个状态选择某个option, 在这个option下选择某个动作

$Q_U(s, \omega, a) : \mathcal{S} \times \Omega \times \mathcal{A} \rightarrow \mathcal{R}$ 的Q函数

$$Q_U(s, \omega, a) = r(s, a) + \gamma \sum_{s'} P(s' | s, a) U(\omega, s') \quad . \quad (2)$$

- 这个是option选择动作的Q函数, s 和 ω 都已经确定, 选择哪个action需要我们进行评估
- 另一种表示方法将其看作 expectation of next states, 有助于进行简化(只需要计算 Q_{Ω} 即可)
- 其中 $U(\omega, s') : \Omega \times \mathcal{S} \rightarrow \mathcal{R}$ 为"option utility function", 这里 s' 为在某个状态 s 下选用动作 a 到达的下一个状态

$$U(\omega, s') = (1 - \beta_{\omega, \vartheta}(s')) Q_{\Omega}(s', \omega) + \beta_{\omega, \vartheta}(s') V_{\Omega}(s') \quad (3)$$

- U 不是Q函数, 而用于选择两种状态
- 若 $\beta_{\omega}(s') = 1$, 则该option在这个状态 s' 终止 $\rightarrow U(\omega, s') = V_{\Omega}(s')$, 计算TD

value

- 若 $\beta_{\omega}(s') = 0$, 则该option ω 还未终止, 继续进行公式(1) $\rightarrow U(\omega, s') = Q_{\Omega}(s', \omega)$, 计算Q value
- 后面还对这些公式的梯度进行了推导, 并引入了advantage function
 - 某些情形下, 每个动作的总回报都不为负
 - 那么所有的梯度值都大于等于0, 此时每个动作出现的概率都会提高
 - 这在很大程度下减缓了学习的速度, 而且也会使梯度的方差很大
 - 因此需要对reward使用某种标准化操作来降低梯度的方差 \rightarrow 减去baseline
 - baseline为TD value时即为advantage function

$$A_{\Omega}(s', \omega) = Q_{\Omega}(s', \omega) - V_{\Omega}(s')$$

3. Algorithms and Architecture

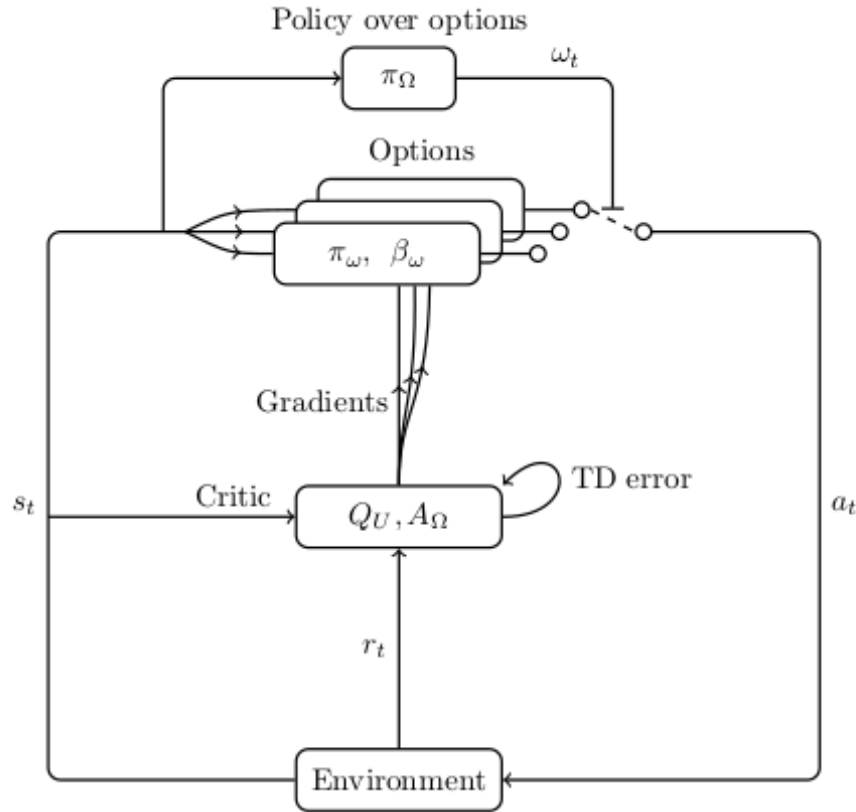


Figure 1: Diagram of the option-critic architecture. The option execution model is depicted by a *switch* \perp over the *contacts* \circ . A new option is selected according to π_Ω only when the current option terminates.

- Architecture:
 - Actor: intra-option policies, termination functions and oper-options policy
 - Critic: Q_U, A_Ω
 - 若 π_Ω 是基于贪婪的, 则根据公式(2), 可得到one-step off-policy update target $g_t^{(1)}$ (update target of $Q_\Omega(s, \omega)$):

$$g_t^{(1)} = r_{t+1} + \gamma \left((1 - \beta_{\omega_t, \vartheta}(s_{t+1})) \sum_a \pi_{\omega_t, \theta}(a | s_{t+1}) Q_U(s_{t+1}, \omega_t, a) + \beta_{\omega_t, \vartheta}(s_{t+1}) \max_\omega \sum_a \pi_{\omega, \theta}(a | s_{t+1}) Q_U(s_{t+1}, \omega, a) \right),$$

- Algorithm

- 这里 $\alpha, \alpha_\theta, \alpha_{\theta_{\text{花体}}}$ 分别为critic, intra-option policies以及termination function的学习速率
- 对于某一个状态 s , 使用over-options policy π_Ω 选择一个option ω
- 在该option终止前, 每次基于其选择一个动作 a , 然后先进入option evaluation阶段
- Options evaluation:
 - 首先初始化 δ 为获得的reward和估计的Q value的差值
 - 若 s' 非终止状态, δ 加上在该状态下继续选择该option ω 的Q估计值
 - 注意若 s' 为终止状态不会跳过这个if判断, 而是让 δ 加上Q估计的最大值 (选择在这个状态下能最大化Q值的option), 相当于选了一个新的"最佳的"option
 - 将该option ω 在该状态 s 下选择该动作 a 的Q值 (intra-option) 表示为

$$Q_U(s, \omega, a) \leftarrow Q_U(s, \omega, a) + \alpha \delta = Q_U(s, \omega, a) + \alpha [r + \gamma Q_\Omega(s', \omega) - Q_U(s, \omega, a)]$$

这里就和DQN很像了, 区别在Q估计值 Q_Ω 为对option的选择

- Options improvement: 对 ω 及这个option的termination function的参数进行更新
- 若 s' 为终止状态, 跳出循环, 重新选择一个option
- 既学习 Q_U 又学习 Q_Ω 很费时间, 由于 Q_U 为expectation of next states (见公式(2)), 可以只学习 Q_Ω , 然后根据 Q_Ω 求取 Q_U 的估计 → 在ALE的实验中用了这个

Algorithm 1: Option-critic with tabular intra-option Q-learning

$s \leftarrow s_0$
Choose ω according to an ϵ -soft policy over options
 $\pi_\Omega(s)$
repeat
 Choose a according to $\pi_{\omega,\theta}(a | s)$
 Take action a in s , observe s', r

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

 2. Options improvement:
 $\theta \leftarrow \theta + \alpha_\theta \frac{\partial \log \pi_{\omega,\theta}(a | s)}{\partial \theta} Q_U(s, \omega, a)$
 $\vartheta \leftarrow \vartheta - \alpha_\vartheta \frac{\partial \beta_{\omega,\vartheta}(s')}{\partial \vartheta} (Q_\Omega(s', \omega) - V_\Omega(s'))$

 if $\beta_{\omega,\vartheta}$ *terminates in* s' **then**
 choose new ω according to ϵ -soft($\pi_\Omega(s')$)
 $s \leftarrow s'$

until s' *is terminal*

4. Experiment

- Navigate room: 检测在忽然改变(定位目标改变)的环境中恢复的能力

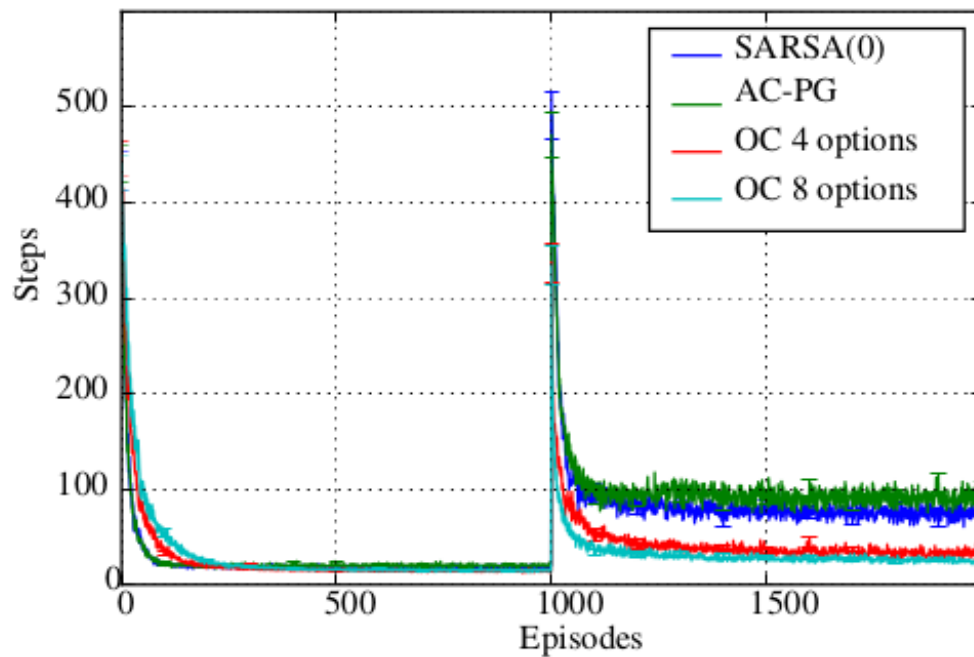


Figure 2: After a 1000 episodes, the goal location in the four-rooms domain is moved randomly. Option-critic (“OC”) recovers faster than the primitive actor-critic (“AC-PG”) and SARSA(0). Each line is averaged over 350 runs.

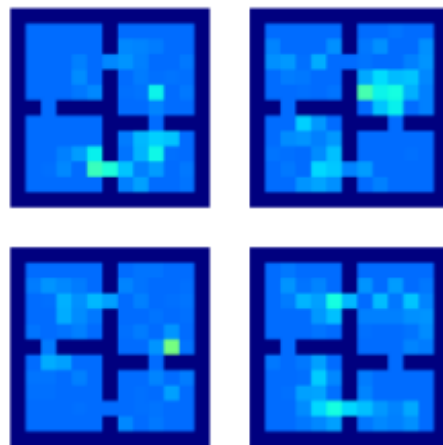


Figure 3: Termination probabilities for the option-critic agent learning with 4 options. The darkest color represents the *walls* in the environment while lighter colors encode higher termination probabilities.

- Pinball:

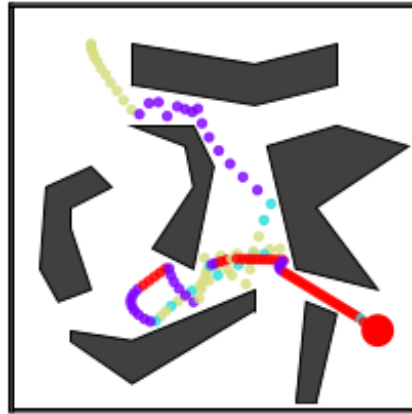


Figure 4: Pinball: Sample trajectory of the solution found after 250 episodes of training using 4 options. All options (color-coded) are used by the policy over options in successful trajectories. The initial state is in the top left corner and the goal is in the bottom right one (red circle).

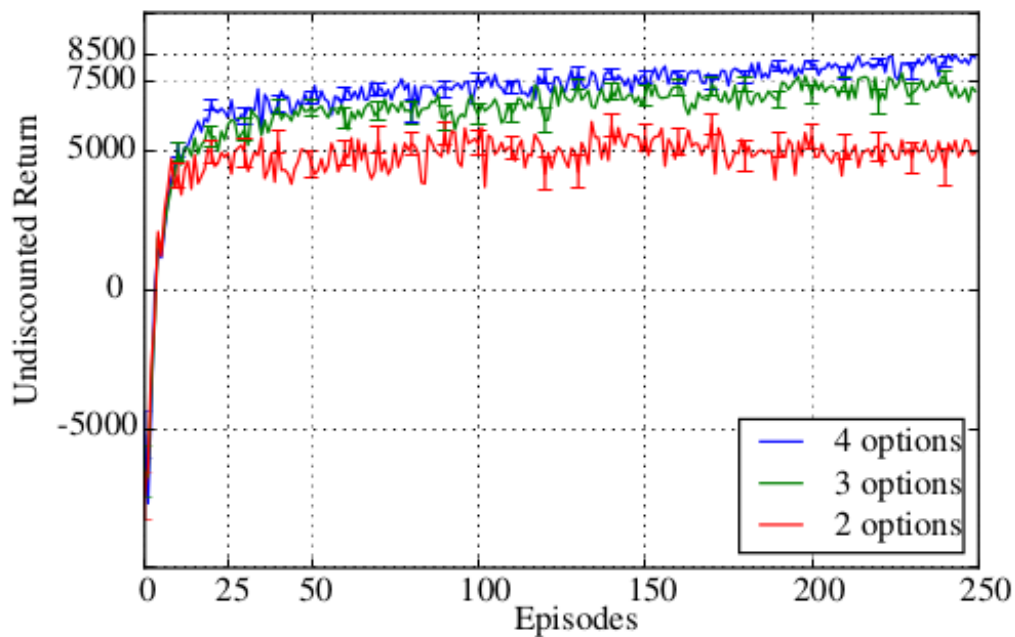


Figure 5: Learning curves in the Pinball domain.

- ALE: 注意之前对 Q_U 计算的简化
 - 用DNN表示intra-option policies以及termination functions, 同时用其计算critic

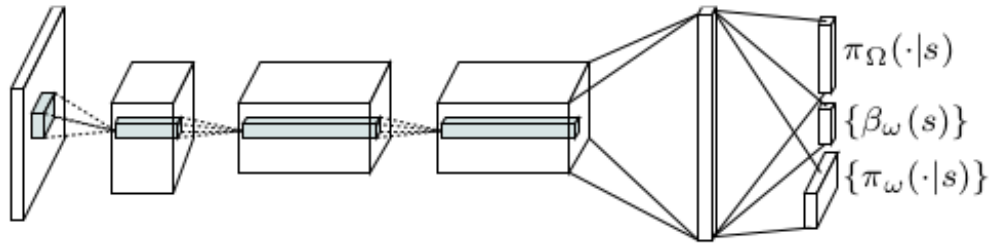


Figure 6: Deep neural network architecture. A concatenation of the last 4 images is fed through the convolutional layers, producing a dense representation shared across intra-option policies, termination functions and policy over options.

- 防止gradient shrink: 向advantage function加一个很小的term $\xi = 0.01$, 这样做类似regularization, 在advantage function接近optimal时令其仍然为正

$$A_{\Omega}(s, \omega) + \xi = Q_{\Omega}(s, \omega) - V_{\Omega}(s) + \xi$$

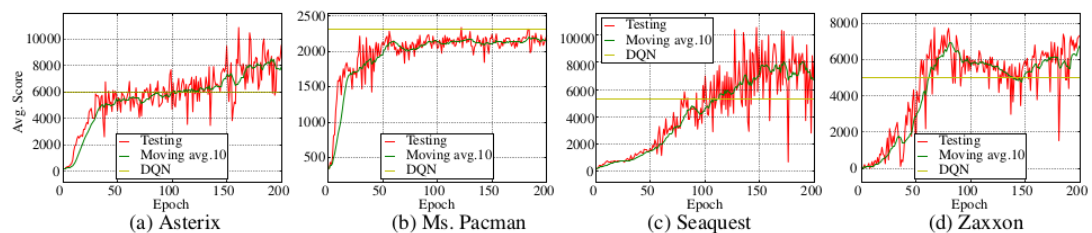


Figure 8: Learning curves in the Arcade Learning Environment. The same set of parameters was used across all four games: 8 options, 0.01 termination regularization, 0.01 entropy regularization, and a baseline for the intra-option policy gradients.

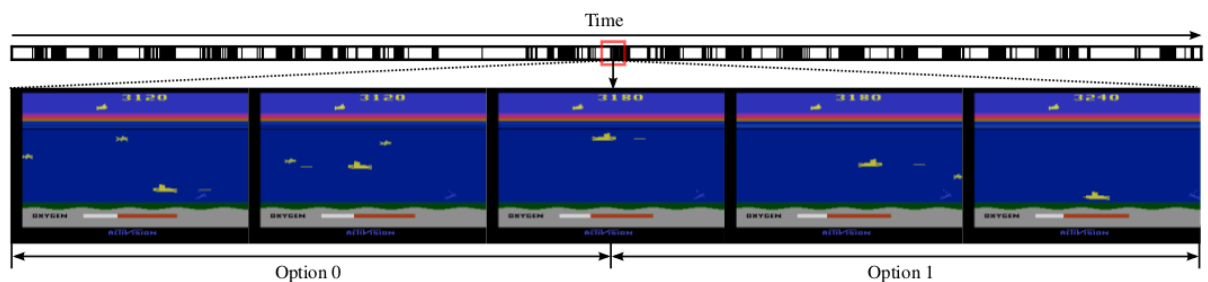


Figure 9: Up/down specialization in the solution found by option-critic when learning with 2 options in Seaquest. The top bar shows a trajectory in the game, with “white” representing a segment during which option 1 was active and “black” for option 2.

5. Summary

- Option-critic
 - Goal: learning faster, updating intra-option policies and termination functions slower → 之后的工作A2OC通过增加deliberation cost来延长每个option的时间
 - 可以实现end-to-end HRL, 仅仅需要预先指定option的数量

- 我理解为每个option都是一个独立的policy, 然后对于 π_{Ω} 根据 state 选 option 的过程其实就类似于 π_{ω} 根据state选action的过程
- 不需要额外的reward和sub-goal, 也可以稍加修改来增加additional reward
- 增加additonal reward可以激励 **"options that are temporally extended by adding a penalty whenever a switching event occurs"**, 这句不大理解, 是想要激励什么样的options???
- Option-critic的不足
 - 本工作的一个假设: 在任何地方都可以使用任何option, 但实际上会不会存在某些地方只能用某几种option呢, 比如工具包括厨具农具, 在餐厅只能在厨具中进行选择
 - FeUDal Network中也提到了OC结构, 说是容易退化到以下两种情况:
 - 整个任务只使用一个option不选择别的, 相当于是学了单个network
 - 每一步都换一个option, 微调过多, 相当于把option当成了一个action function → **A2OC进行了一些提升**
 - 因此和OC不同, FeUDal Network的master policy是会为worker policy设定显式的目标去完成的
- Further reading:
 - A2OC (A3C + OC) : 1709.04571 - When waiting is not an option: Learning options with a deliberation cost
 - 1710.11089 - Eigenoption Discovery through the Deep Successor Representation
 - 1711.03817 - Learning with Options that Terminate Off-Policy
 - 1802.03236 - Learning Robust Options
- Implementations:
 - Basic OC:
 - Theano + Lasagne for ALE : https://github.com/jeanharb/option_critic
 - TF, with deliberation cost : <https://github.com/bhairavmehta95/option-critic>
 - TF implementation : <https://github.com/yadrimz/option-critic>
 - Extensions:
 - prioritized_option_critic on 4-rooms : https://github.com/YuejiangLIU/prioritized_option_critic
 - A2OC : https://github.com/jeanharb/a2oc_delib
 - Eigenoption-Critic : https://github.com/ioanachelu/EigenOption-Critic_SR

