1709.04571 - When Waiting is not an Option : Learning Options with a Deliberation Cost

- **Yunqiu Xu**
- Related reading:
  - CASL, integration of this work: Omidshafiei et al, 2017. Crossmodal Attentive Skill Learner
  - Source code: https://github.com/jeanharb/a2oc_delib

---

1. Introduction

- Challenges:
  - What have been done: how to learn
  - What we need to tackle: **what good options should be**
- Our work: A2OC
  - Leverage the bounded rationality framework $\rightarrow$ what would make good temporal abstractions for an RL system
  - Option-critic architecture, Bacon et al, 2017

2. Preliminaries

- **Here I borrow notes from CASL**
- An option $\omega \in \Omega$ consists of:
  - Initiation set $I \subseteq S$
  - Intra-option policy $\pi_\omega : S \rightarrow A$, this is **sub-policy**
  - Termination condition $\beta_\omega : S \rightarrow [0, 1]$
  - Given a state, master policy π select an option (suitable initiation set), then its intra-option policy will be executed to reach terminate state of this subtask $\rightarrow$ a new state for next iteration until final end

- $Q_\Omega(s,\omega)$: Option value function for option $\omega \in \Omega$

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s)\left((r(s,a) + \gamma \sum_{s'} T(s'|s,a)U(s',\omega)\right)$$

- $U(s',\omega)$: option utality function

$$U(s',\omega) = (1-\beta_\omega(s'))Q_\Omega(\omega,s') + \beta_\omega(s')(V_\Omega(s') - c)$$

  - If $\beta_\omega(s') = 1$, sub-task ends $\rightarrow U(s',\omega) = V_\Omega(s') - c \rightarrow$ Master policy
  - If $\beta_\omega(s') = 0$, still sub-task $\rightarrow U(s',\omega) = Q_\Omega(\omega,s')$
- c : deliberation cost, add penalty when options terminate $\rightarrow$ **let options terminate less frequently**
- $V_\Omega(s')$: value function over options (master policy π_{Omega})

$$V_\Omega(s') = \sum_\omega \pi_\Omega(\omega|s')Q_\Omega(\omega,s')$$

3. Algorithm

Algorithm 1: Asynchronous Advantage Option-Critic

Initialize global counter $T \leftarrow 1$
Initialize thread counter $t \leftarrow 1$
$c \leftarrow 0$
repeat
  $t_{start} = t$
  $s_t \leftarrow s_0$
  Reset gradients: $dw \leftarrow 0$, $d\theta_\beta \leftarrow 0$ and $d\theta_\pi \leftarrow 0$
  Choose o_t with an ϵ-soft policy over options $\mu(s_t)$
  repeat
    Choose a_t according to $\pi_\theta(\cdot|s_t)$
    Take action a_t in s_t, observe r_t, s_{t+1}
    $\widetilde{r}_t \leftarrow r_t + c_t$
    if *the current option o_t terminates in s_{t+1}* **then**
      choose new o_{t+1} with ϵ-soft($\mu(s_{t+1})$)
      $c \leftarrow \eta$
    else
      $c \leftarrow 0$
    end
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until *episode ends or $t - t_{start} == t_{max}$ or ($t - t_{start} > t_{min}$ and o_t terminated)*
  $G = V_\theta(s_t)$
  for $k \in t-1, ..., t_{start}$ **do**
    $G \leftarrow \widetilde{r}_k + \gamma G$
    Accumulate thread specific gradients:
    $dw \leftarrow dw - \alpha_w \frac{\partial (G - Q_\theta(s_k, o_k))^2}{\partial w}$
    $d\theta_\pi \leftarrow d\theta_\pi + \alpha_{\theta_\pi} \frac{\partial \log \pi_\theta(a_k|s_k)}{\partial \theta_\pi} (G - Q_\theta(s_k, o_k))$
    $d\theta_\beta \leftarrow$
      $d\theta_\beta - \alpha_{\theta_\beta} \frac{\partial \beta_\theta(s_k)}{\partial \theta_\beta} (Q_\theta(s_k, o_k) - V_\theta(s_k) + \eta)$
  end
  Update global parameters with thread gradients
until $T > T_{\max}$

4. Experiment

5. Conclusion

- Use deliberation cost as a way to incentivize the creation of options which persist for a longer period of time.
  - Better performance
  - Prevent options from terminate frequently
- I just take a simple browse of this work, maybe ... to be continued :)