HRL: Options

B. Ravindran

Options Framework

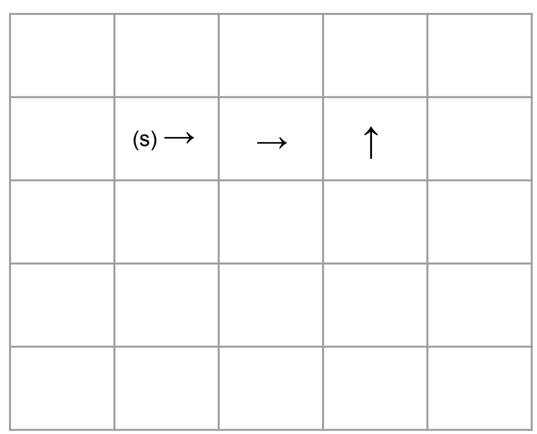
- Macro Actions: Sequence of actions put together to be used as a ‘single’ action.
- Options:
    - Initiation Set $\mathcal{I}_o \subseteq S$
    - Policy π_o
    - Termination $\beta_o : S \rightarrow [0, 1]$

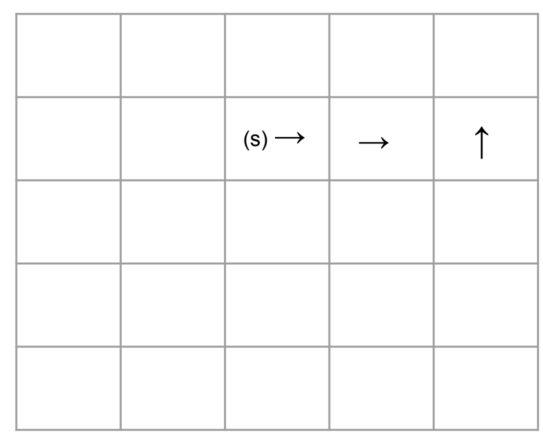
Types of Options

- Markov Options:
  - π_o depends only on current state

- Semi-Markov Options:
  - π_o depends on history since option started

example:
Semi-Markov Option
π_o:(Right)+(Right)+(Up)





Learning with Options

- SMDP Q-Learning:

- In SMDP Q-Learning, if a primitive action was selected in a state, the value of the state-action pair is updated according to the regular Q-Learning update rule.
- If the agent selected an option o, no state-action values are updated until o terminates.
- At this point, the cumulative, discounted reward received during the execution of the option is used to update the value of the option in the state s in which it was initiated.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[\bar{r}_{t+\tau} + \gamma^{\tau} \max_{a'} Q\big(s_{t+\tau}, a'\big) - Q(s_t, a_t) \right]$$

$$\bar{r}_{t+\tau} = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{\tau-1} r_{t+\tau}$$

Intra-Option Q-Learning

- At every step, the state-action value for the primitive action as well as the state-action value for all options that would have selected the same action are updated, regardless of the option in effect.

- Let $\pi_o - a_1(for\, s_1)\,,\, a_2\,(for\, s2)\ldots\ldots$
- For options,

$$Q(s_{1,}o) = Q(s_1, o) + \alpha[r_1 + \gamma Q(s_2, o) - Q(s_1, o)]$$ If not terminating at s2

$$= Q(s_1, o) + \alpha\left[r_1 + \gamma \max_a Q(s_2, a) - Q(s_{1,}o)\right]$$ If terminating at s2

[Tutorial] - Implement as a single function with termination probability

Learning with Options

- For primitive actions (state-action pairs), we use regular Q-learning update.

$$Q(s_1, a_1) = Q(s_1, a_1) + \alpha\left[r_1 + \gamma \max_a Q(s_1, a) - Q(s_1, a_1)\right]$$
$$Q(s_2, a_2) = \ldots$$

- Additionally, an option execution allows us to update for all other options that are consistent with the first option. (for every other option o' that would have selected the same action a in particular states)

- Suppose $\pi_o(s_1) = a^{\#} \,\&\, \pi_{o'}(s_1) = a^{\#}$
- Even when executing option o, o' can be updated

$$Q(s_1, o') = Q(s_1, o') + \alpha(r_1 + \gamma Q(s_2, o') - Q(s_1, o'))$$

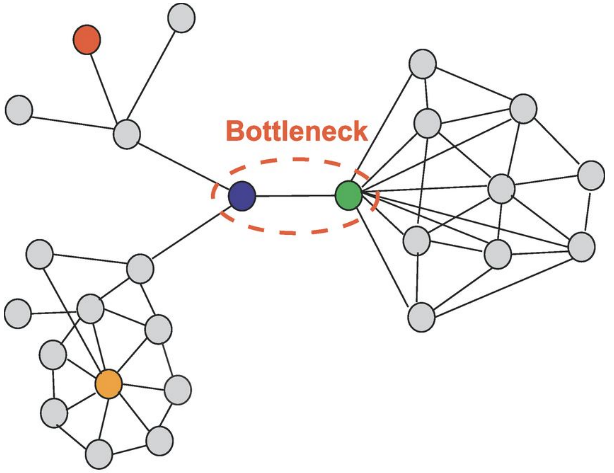
Option Discovery

- What makes an option ‘good’?
  - Reusability
  - Cuts down on exploration
  - Transfer Learning
  - Explainability

- We can use surrogate measures to aid option discovery,
  - Bottlenecks/Access States(eg: Doorways)
    - Diverse Density
  - Graph Partitions
    - Betweenness
  - Frequency of Changes
  - Bisimulation Metrics

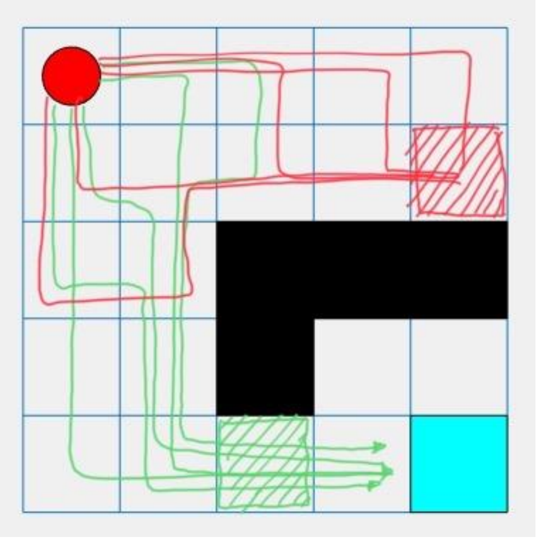
Finding Bottlenecks

- MDP can be segmented and modeled as a graph. (Graph: nodes -> states; edge -> action)
- Find components of graph which are weakly connected. (Graph partitioning)
- States where weak connections happen are ‘bottleneck’ states



Diverse Density

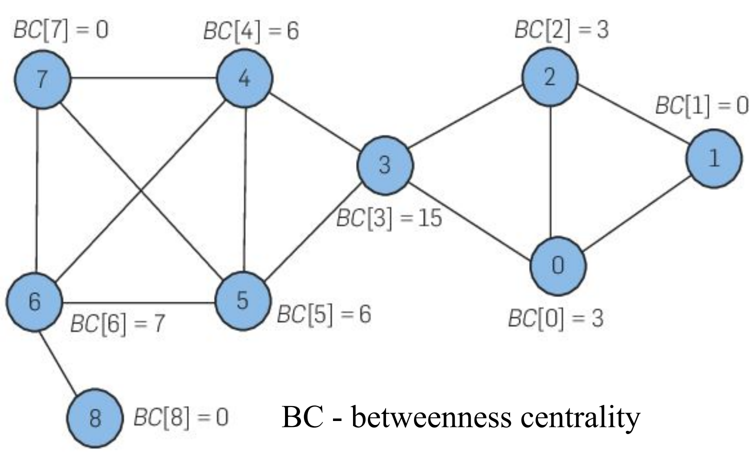
- Assume we’ve obtained a bunch of successful & unsuccessful trajectories. (Experience Replay)
- Find states that appear frequently on successful and rarely on unsuccessful trajectories.
- “The agent needs to get through these states to reach the goal”.



RED: unsuccessful trajectories
GREEN: successful trajectories

Betweenness Centrality

- Pick (all possible) pair of nodes on the graph and calculate the shortest path.
- A node has high betweenness if many shortest-paths pass through it.
- States with high betweenness centrality can be considered bottlenecks.



BC - betweenness centrality

Small World Options

- Random options inserted with the expected length of these options following a certain probability distribution.

- Exploration time can be cut down significantly.

- Makes best use of data in comparison.