How to construct good temporal abstractions

Doina Precup
McGill University
Joint work with Pierre-Luc Bacon and Jean Mehreb-Harb

EWRL, December 2016
Options framework

• Suppose we have an MDP $\langle S, A, r, P, \gamma \rangle$

• An option $\omega$ consists of 3 components
  – An initiation set of states $I_\omega \subseteq S$ (aka precondition)
  – A policy $\pi_\omega : S \times A \rightarrow [0, 1]$
    $\pi_\omega(a|s)$ is the probability of taking $a$ in $s$ when following option $\omega$
  – A termination condition $\beta_\omega : S \rightarrow [0, 1]$
    $\beta_\omega(s)$ is the probability of terminating the option $\omega$ upon entering $s$

• Eg., robot navigation: if there is no obstacle in front ($I_\omega$), go forward ($\pi_\omega$) until you get too close to another object ($\beta_\omega$)

Cf. Sutton, Precup & Singh, 1999; Precup, 2000
Introducing options in an MDP induces a related semi-MDP

Hence *all planning and learning algorithms* from classical MDPs transfer directly to options (Cf. Sutton, Precup & Singh, 1999; Precup, 2000)

But planning and learning with options can be much faster!
Frontier: Option Discovery

- Options can be given by a system designer
- If subgoals / secondary reward structure is given, the option policy can be obtained, by solving a smaller planning or learning problem (cf. Precup, 2000)
- What is a good set of subgoals / options?
- This is a representation discovery problem
- Studied a lot over the last 15 years
- Bottleneck states and change point detection currently the most successful methods
Goals of our current work

- Explicitly state an *optimization objective* and then solve it to find a set of options
- Handle both *discrete and continuous* set of state and actions
- Learning options should be *continual* (avoid combinatorially-flavored computations)
- Options should provide *improvement within one task* (or at least not cause slow-down...)

EWRL, December 2016
Actor-critic architecture

- Clear optimization objective: average or discounted return
- Continual learning
- Handles both discrete and continuous states and actions
Option-critic architecture

- Parameterize internal policies and termination conditions
- Policy over options is computed by a separate process (planning, RL, ...)

Figure 1: The option-critic architecture consists of a set of option policies and a critic. Gradients can only take place when a termination event is encountered. Any action could fail with probability 0.1, in which case the agent would simply remain in the same state. The option-critic architecture consists in a parametrized policy over options which can be solved readily using a parametrized critic.
Formulation

• The option-value function of a policy over options $\pi_\Omega$ is given by

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

where

$$Q_U(s, \omega, a) = r_a(s) + \gamma \sum_{s'} P_a(s'|s)U(\omega, s')$$

• The last quantity is the utility from $s'$ onwards, *given that we arrive in $s'$ using $\omega$*

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

• We parameterize the internal policies by $\theta$, as $\pi_\omega, \theta$, and the termination conditions by $\nu$, as $\beta_\omega, \nu$

• *Note that $\theta$ and $\nu$ can be shared over the options!*
Main result: Gradient updates

• Suppose we want to optimize the expected return: \( \mathbb{E} \{ Q_{\pi\Omega}(s, \omega) \} \)

• The gradient wrt the internal policy parameters \( \theta \) is given by:

\[
\mathbb{E} \left\{ \frac{\partial \log \pi_{\omega,\theta}(a|s)}{\partial \theta} Q_U(s, \omega, a) \right\}
\]

This has the usual interpretation: take better primitives more often inside the option.

• The gradient wrt the termination parameters \( \nu \) is given by:

\[
\mathbb{E} \left\{ -\frac{\partial \beta_{\omega,\nu}(s')}{\partial \nu} A_{\pi\Omega}(s', \omega) \right\}
\]

where \( A_{\pi\Omega} = Q_{\pi\Omega} - V_{\pi\Omega} \) is the advantage function.

This means that we want to lengthen options that have a large advantage.
Results: Options transfer

- 4-rooms domain, tabular representations, value functions learned by Sarsa
- Learning in the first task no slower than using primitives
- Learning once the goal is moved faster with the options
Results: Nonlinear function approximation

- Atari simulator, DQN to learn value function over options, actor as above

- Performance matching or better than DQN
Results: Learned options are intuitive

- In rooms environment, terminations are more likely near hallways (although there are no pseudo-rewards provided)

- In Seaquest, separate options are learned to go up and down
What are beneficial options

• Successful simultaneous learning of terminations and option policies
• But, as expected, *options shrink over time* unless a margin is required for the advantage
• Intuitively, using longer options increase the speed of learning and planning (but may lead to a worse result in call-and-return execution)
• What is the right tool to formalize this intuition?
Recall: Planning with options

• Options have corresponding reward and transition models
• Reward $r_\omega(s)$: gives the expected return during $\omega$’s execution from $s$
• Transition model $P_\omega(s'|s)$: gives a sub-probability distribution over next states (reflecting the discount factor $\gamma$ and the option duration) given that $\omega$ executes from $s$
• Planning with option models is just like planning with primitives (relied on generalized Bellman equations)

$$V_{\pi_\Omega}(s) = \sum_{\omega} \pi_\Omega(\omega|s) \left( r_\omega(s) + \sum_{s'} P_\omega(s, s') V_{\pi_\Omega}(s') \right)$$

• Models of options also obey Bellman equations, e.g.:

$$r_\omega(s) = \sum_a \pi_\omega(a|s) \left( r_a(s) + \gamma \sum_{s'} (1 - \beta_\omega(s')) P(s'|s, a) r_\omega(s') \right)$$
Some algebra

• Plugging this into the generalized Bellman equations we get:

\[ V_{\pi \Omega}(s) = \sum_{\omega} \pi_{\Omega}(\omega|s) \sum_a \pi_\omega(a|s) \left( r_a(s) + \gamma \sum_{s'} P_a(s'|s)(1 - \beta_\omega(s'))r_\omega(s') \right) \]

\[ + \gamma \sum_{\omega} \pi_{\Omega}(\omega|s) \left( \sum_{s'} P_\omega(s, s')V_{\pi \Omega}(s') \right) \]

• Manipulating the first term further, we get: \( r_\sigma(s) + \gamma \sum_{s'} H(s, s')r_\omega(s') \)

where:

\[ r_\sigma(s) = \sum_a \sum_\omega \pi_{\Omega}(\omega|s)\pi_\omega(a|s)r_a(s) = r_\sigma(s) \]

where \( \sigma \) is the “flattened” policy which marginalizes over options

\[ H(s, s') = \sum_a \sum_\omega \pi_{\Omega}(\omega|s) \sum_{s'} P_a(s'|s)(1 - \beta_\omega(s')) \]
Re-writing the policy evaluation equation

With a similar manipulation to the second part of the equation, we get a similar term:

\[ F(s, s') = \gamma(P_\sigma(s, s') - H(s, s')) + \gamma \sum_{\bar{s}} H(s, \bar{s})F(\bar{s}, s') \]

Putting it all together, we get a modified backup:

\[ V_{\pi\Omega} = (I - \gamma H)^{-1}r_\sigma + \gamma(I - \gamma H)^{-1}(P_\sigma - H)V_{\pi\Omega} \]

Note this is a linear system of equations
Matrix splitting (Varga, 1962)

- Suppose you want to solve a linear system $Ax = b$ with $A$ non-singular
- Eg in policy evaluation, we want to solve: $(I - \gamma P_\pi)V_\pi = r_\pi$
- A preconditioner is a way to transform $A$ in order to make the system more easily solvable
- Suppose you can write $A = M - N$, where $M$ is also non-singular
- $M$ and $N$ are a splitting of matrix $A$
- Then, the solution of $Ax = b$ system can be obtained by the following iteration: $x_{k+1} = M^{-1}Nx_k + M^{-1}b$
- $M$ can be viewed as a pre-conditioner and can be designed to manipulate the spectral radius of $M^{-1}N$
Connection to other algorithms

- Consider the task of learning a value function for policy $\sigma$ (marginal over primitives)
- Matrix $A = I - \gamma P_\sigma$
- If we do matrix splitting with $M = I - \gamma H$ and $N = \gamma(P_\sigma - H)$, we get exactly:

$$V_{\pi_\Omega} = (I - \gamma H)^{-1}r_\sigma + \gamma(I - \gamma H)^{-1}(P_\sigma - H)V_{\pi_\Omega}$$

- For options that have the same intra-option policy, having longer duration (lower $\beta$ everywhere) leads to better spectral radius, so fewer iterations
- However, $H$ needs to be acquired (which would be harder)
A proposal: Deliberation cost

• Assumption: *executing a policy is cheap, deciding what to do is expensive*
  – Many choices may need to be evaluated (branching factor over actions)
  – In planning, many next states may need to be considered (branching factor over states)
  – Evaluating the function approximator might be expensive (e.g. if it is a deep net)

• Deliberation is also expensive in animals:
  – Energy consumption (to engage higher-level brain function)
  – Missed opportunity cost: thinking too long means action is delayed
Problem formulation

• Let \( c(s, \omega) \) be the immediate cost of deliberating to choose \( \omega \in s \)
• In the call-and-return model, it is easy to see that we have a value function that expresses total deliberation cost given by the following Bellman equation:

\[
Q_c(s, \omega) = -c(s, \omega) + \sum_{s'} P_\omega(s'|s) \sum_{\omega'} \pi_\Omega(\omega'|s') Q_c(s', \omega')
\]

• We can obtain \( Q_c \) using learning, value iteration etc
• New objective: maximize reward with reasonable effort

\[
\max_{\Omega} \mathbb{E}[Q_\Omega(s, \omega) + \xi Q_c(s, \omega)]
\]

• \( \xi \geq 0 \) controls the trade-off between value and computation effort
Illustration: 4 rooms

- Emphasizing deliberation cost, shifts the policy towards using options
- Number of iterations of planning is smaller for higher deliberation cost penalties
- When options are learned in one task and then used to plan in a different task, options obtained with deliberation costs are more robust
Conclusions and future work

• Option-critic allows using policy gradient ideas for continual option construction
• Matrix splitting and pre-conditioners - an interesting tool for analysis
• However, the problem of designing good pre-conditioners is hard (unless there is special structure in environment) so may not be useful for direct algorithm design
• Including deliberation cost as an optimization criterion gives rise to more robust options
• Lots of things to do:
  – More empirical work!
  – Incorporating initiation sets in option-critic (currently options initiate at every state)
  – Theoretical properties of deliberation cost (relationship to action-gap, time-regularized options)