Advances in option construction: The option-critic architecture

Pierre-Luc Bacon and Doina Precup
McGill University
With thanks to Jean Harb

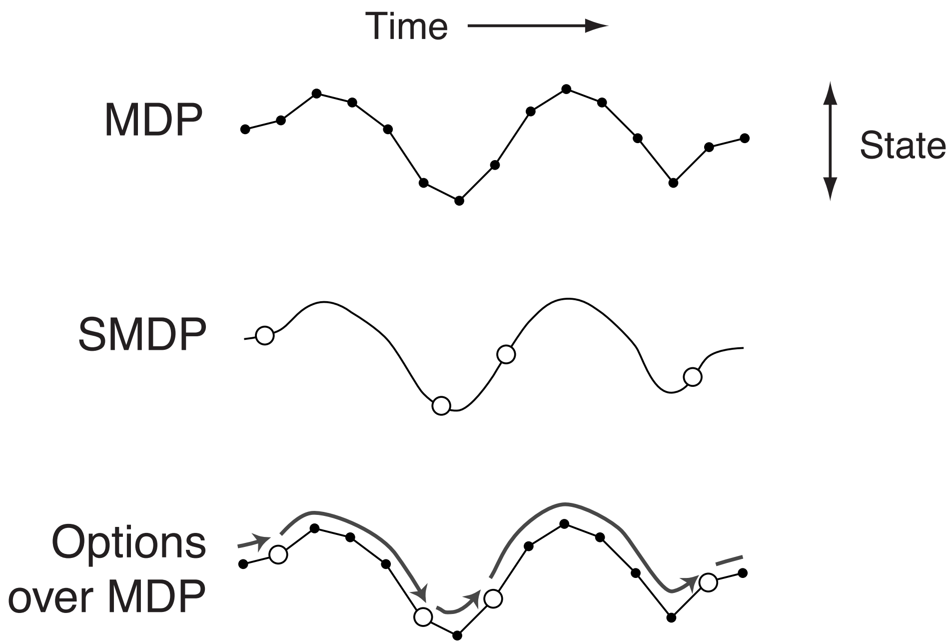
ICML Abstraction in RL Workshop, June 2016

Options framework

- Suppose we have an MDP $\langle \mathcal{S}, \mathcal{A}, r, P, \gamma \rangle$
- An *option* ω consists of 3 components
  - An *initiation set* of states $I_\omega \subseteq \mathcal{S}$ (aka precondition)
  - A *policy* $\pi_\omega : \mathcal{S} \times \mathcal{A} \to [0, 1]$
    $\pi_\omega(a|s)$ is the probability of taking a in s when following option ω
  - A termination condition $\beta_\omega : \mathcal{S} \to [0, 1]$:
    $\beta_\omega(s)$ is the probability of terminating the option ω upon entering s
- Eg., robot navigation: if there is no obstacle in front (I_ω), go forward (π_ω) until you get too close to another object (β_ω)
- One can use a cumulative density function for the termination as well (cf. Comanici and Precup, 2010)

Cf. Sutton, Precup & Singh, 1999; Precup, 2000

MDP + Options = Semi-Markov Decision Precess

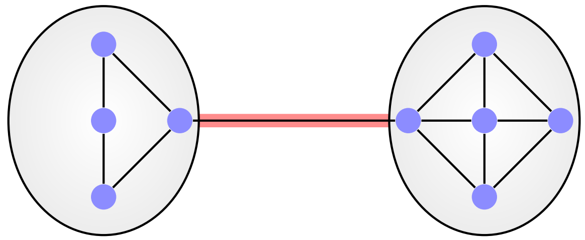


- Introducing options in an MDP induces a related semi-MDP
- Hence *all planning and learning algorithms* from classical MDPs transfer directly to options (Cf. Sutton, Precup & Singh, 1999; Precup, 2000)
- But planning and learning with options can be much faster!

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

Bottleneck states

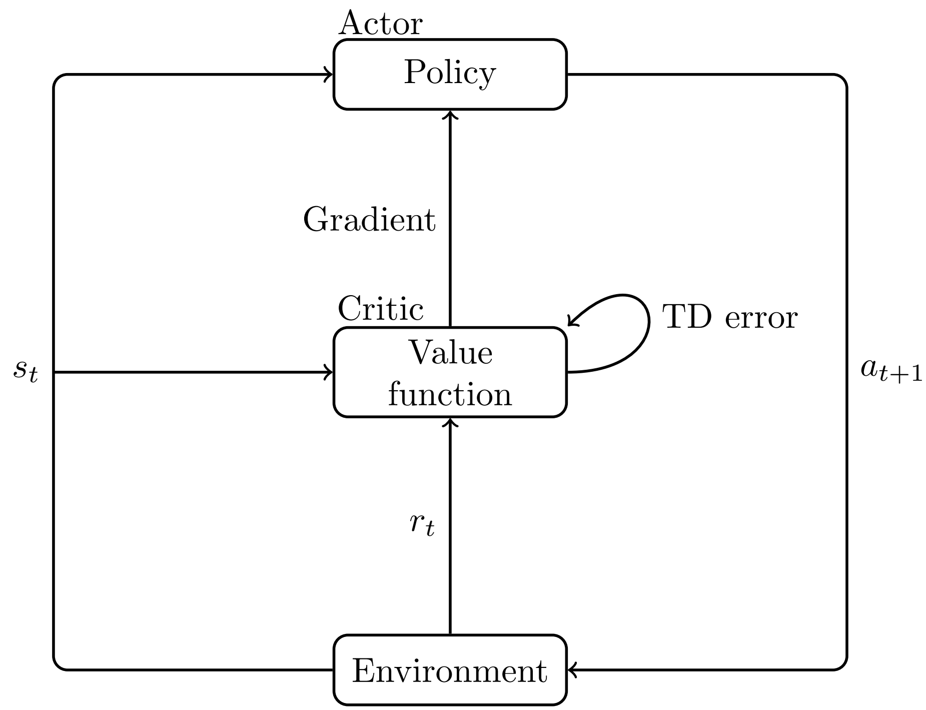


- Perhaps the most explored idea in options construction
- A bottleneck is a special state, which is visited more often than others, allows "circulating on the graph"
- Lots of different approaches!
  - Frequency of states (McGovern et al, 2001, Stolle & Precup, 2002)
  - Graph partitioning / state graph analysis (Simsek et al, 2004, Menache et al, 2004, Bacon & Precup, 2013)
  - Information-theoretic ideas (Peters et al., 2010)
- People seem quite good at generating these (cf. Botvinick, 2001, Solway et al, 2014)
- *Main drawback: expensive both in terms of sample size and computation*

Goals of our current work

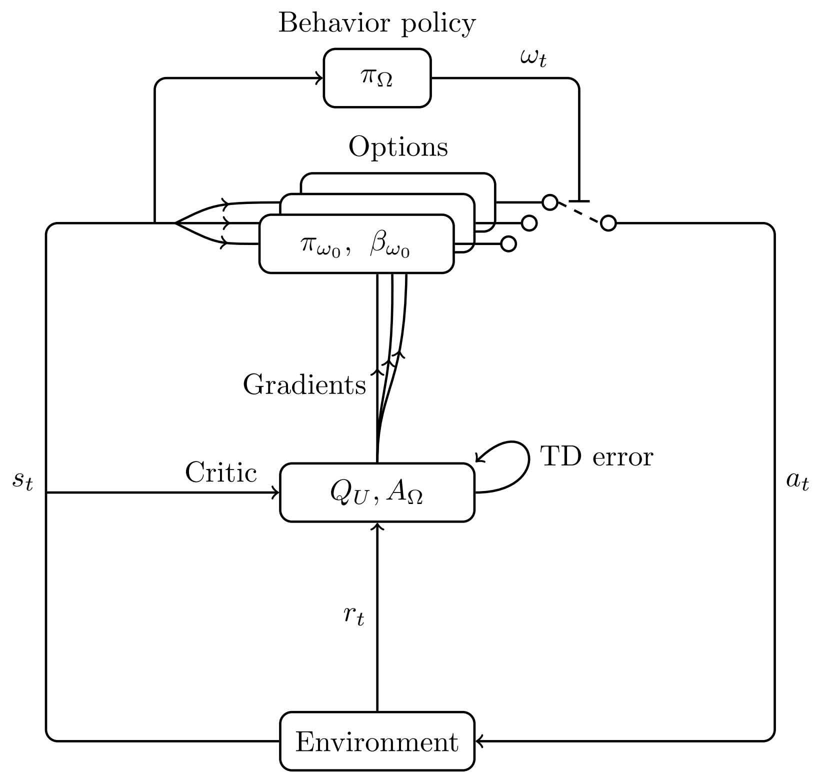
- Explicitly state an *optimization objective* and then solve it to find a set of options
- Handle both *discrete and continuous* set of state and actions
- Learning options should be *continual* (avoid combinatorial-flavored computations)
- Options should provide *improvement within one task* (or at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

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, ...)

Formulation

- The option-value function of a policy over options π_Ω is given by

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

where

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

- The last quantity is the utility from s' onwards, *given that we arrive in s' using ω*

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

- We parameterize the internal policies by θ, as $\pi_{\omega,\theta}$, and the termination conditions by ν, as $\beta_{\omega,\nu}$
- *Note that θ and ν can be shared over the options!*

Main result: Gradient updates

- Suppose we want to optimize the expected return: $\mathbb{E}\left\{Q_\Omega(s,\omega)\right\}$
- The *gradient wrt the internal policy parameters* θ is given by:

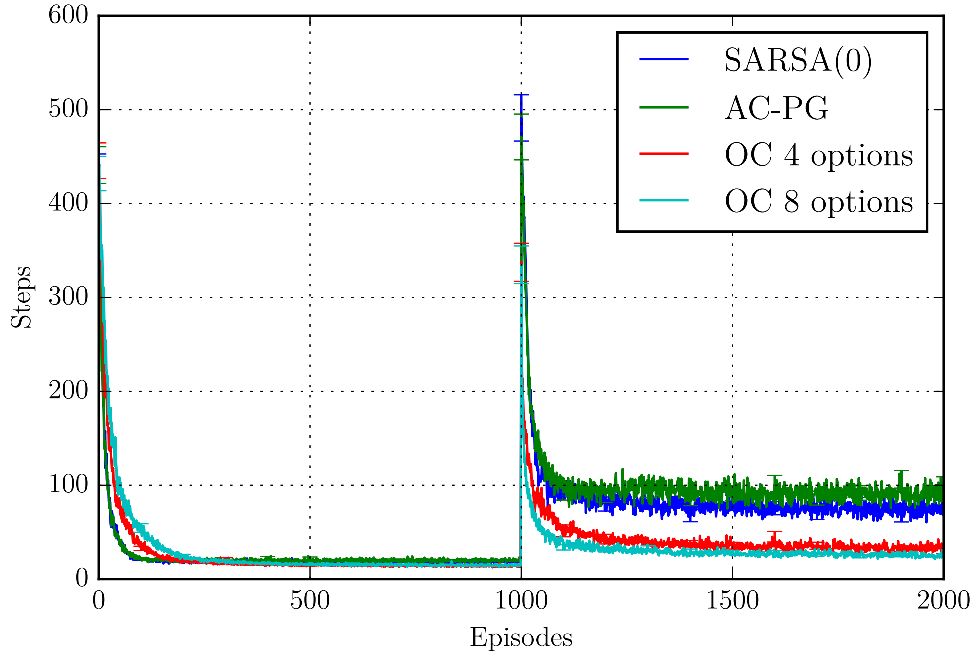
$$\mathbb{E}\left\{\frac{\partial \log \pi_{\omega,\theta}(a|s)}{\partial \theta} Q_U(s,\omega,a)\right\}$$

  This has the usual interpretation: *take better primitives more often* inside the option
- The *gradient wrt the termination parameters* ν is given by:

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

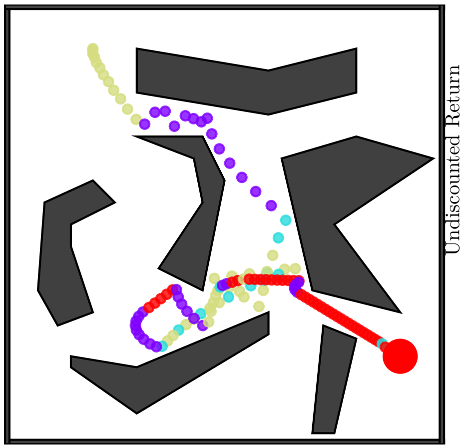
  where $A_\Omega = Q_\Omega - V_\Omega$ is the advantage function
  This means that we want to *lengthen options that have a large advantage*

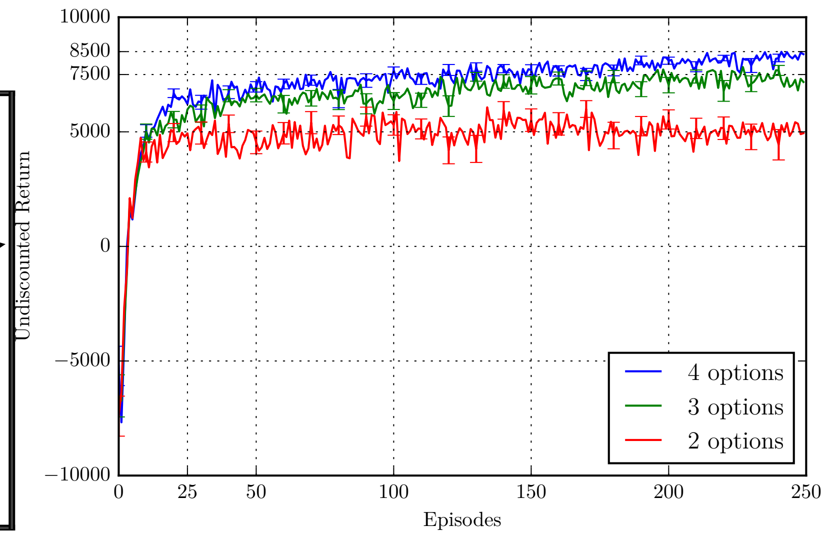
Results: Options transfer



- 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

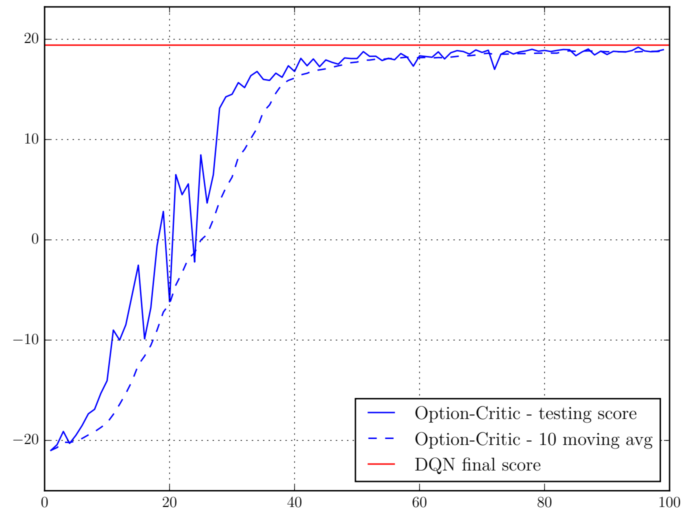
Results: Linear function approximation





- Internal option policies, termination conditions and policy over options all learned simultaneously
- Only number of options and function approximator are given
- Linear function approximation for the value functions, logistic for the terminations
- Interesting, extended options are learned

Results: Nonlinear function approximation



- Atari simulator, Pong, DQN to learn value function over options
- Internal policies for options are given: repeat the same primitive action (one option per primitive)
- Termination policies represented as a logistic over the DQN features
- Successful simultaneous learning of terminations and option policies
- But, as expected, *options shrink over time*

A new 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

Example: Immediate cost = number of actions

- With primitive actions: average cost of $|\mathcal{A}|$ per time step
- With options only: average cost of $|\Omega|$ incurred *only when re-deciding what to do*
- If we re-decide on average every kth step, and if $|\Omega| < k|\mathcal{A}|$, deliberation with options is cheaper
- Even if we use both options and primitive actions, average deliberation is cheaper if $|\Omega| < (k-1)|\mathcal{A}|$

Problem formulation

- Let $c(s, \omega)$ be the immediate cost of deliberating to choose ω 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:

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

  where P_γ is the discounted transition sub-probability (sums to $< \gamma$)
- We can obtain Q_c using learning, value iteration etc
- *New objective: maximize reward with reasonable effort*

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

- $\xi \geq 0$ controls the trade-off between value and computation effort

Interesting properties

- Immediate cost of deliberation is computed based on properties of the environment
- We do not need pseudo-rewards for sub-goals!
- Value function over options is still obtained accurately
- If $\xi = 0$ we simply optimize returns as before

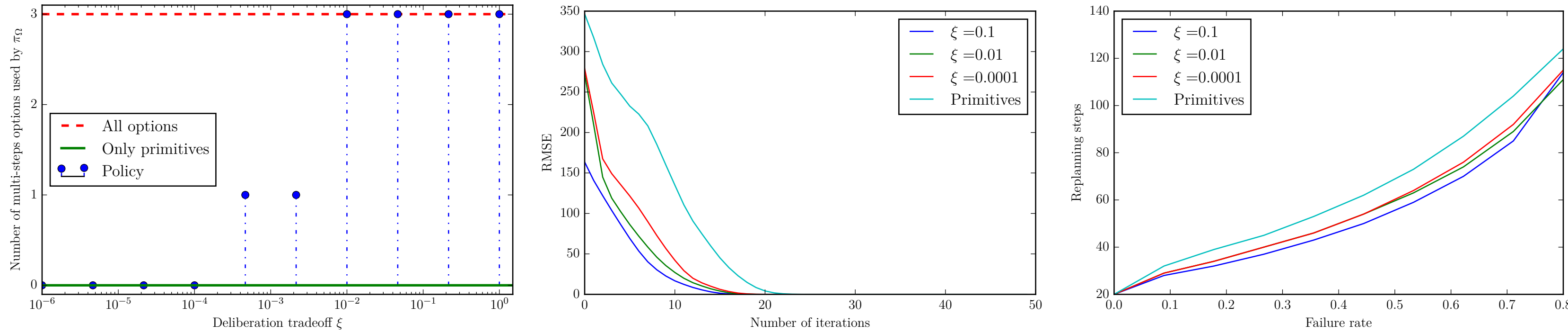
Illustration: 4 rooms

- If we want to do planning, backing up from multiple states is expensive so we reflect this in an immediate cost:

$$c(s, \omega) = \sum_{s'} \mathbb{I}_{P(s'|s,\omega) > \epsilon} |\Omega(s')|$$

- $\epsilon \in [0, 1]$ is a constant that can be used to ignore transitions of low probability
- We use this in the 4 rooms domain, using option-critic to learn the options and dynamic programming to find π_Ω
- Option policies and terminations parameterized as before

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

Relationship to other ideas

- Human problem solving: Solway et al (2014) proposed a Bayesian model selection framework to explain subgoal learning by humans trying to navigate an unknown map
- Humans found subgoals “around” bottleneck states
- Inspection of their criterion (Bayesian fit to the data) shows strong similarity with using a cost equal to the branching factor between options, plus the branching factor within the active option
- *Deliberation costs can also explain the value of options for exploration*
  - Travelling quickly around the environment means values will become accurate more quickly
  - The best action becomes clear earlier, which would make it easier to choose

Conclusions

- Option-critic allows using policy gradient ideas for continual option construction
- Including deliberation cost as an optimization criterion gives rise to more robust options
- Lots of things to do:
  - More empirical evidence!
  - Incorporating initiation sets in option-critic (currently options initiate at every state)
  - Theoretical properties of deliberation cost (relationship to action-gap, time-regularized options)
  - Relationship to transfer learning