Option Discovery Algorithms

Lu Wang
East China Normal University
Overview

• Introduction
  • Temporal Abstraction in RL
  • Options
  • Semi-MDP

• Option Discovery Algorithms
  • policy gradient based methods
  • information-theoretic based
  • Variational options discovery
Temporal abstraction

High level steps
• Grind the beans, measure the right quantity of water, boil the water

Low level steps
• Wrist and arm movements while adding coffee to the filter, ...
Temporal abstraction in AI

A cornerstone of AI planning since the 1970’s:


It has been shown to:

• Generate shorter plans
• Reduce the complexity of choosing actions
• Provide robustness against model misspecification
• Improve exploration by taking shortcuts in the environment
Temporal abstraction in RL

How can an agent represent stochastic, closed-loop, temporally-extended courses of action?

- HAMs (Parr & Russell 1998; Parr 1998)
- MAXQ (Dietterich 2000)
- **Options** (Sutton, Precup & Singh 1999; Precup 2000)

**options - skills - macros - temporally abstract actions**
(Sutton, McGovern, Dietterich, Barto, Precup, Singh, Parr...)
Example

Actions
• North, East, South, West

Reward
• +1 for transitions into G
• 0 otherwise

$\gamma = 0.9$
Options

- A generalization of actions
- Starting from an initiation state, specify a way of choosing actions until termination
- Example: go-to-hallway
Markov Options

• An option can be represented as a triple:
  \[ o = < I, \pi, \beta > \]
  - \( I \subseteq S \) is the set of states in which \( o \) may be started.
  - \( \pi: S \times A \rightarrow [0,1] \) is the policy followed during \( o \).
  - \( \beta: S \rightarrow [0,1] \) is the probability of terminating in each state.

Example:
go-to-hallway
One-Step Options

- A primitive action $a \in \bigcup_{s \in S} A_s$ of the base MDP is also an option, called a one-step option:
  - $I = \{s : a \in A_s\}$
  - $\pi(s, a) = 1, \forall s \in I$
  - $\beta(s) = 1, \forall s \in S$
Markov vs. Semi-Markov Options

• Markov option: policy and termination condition depend only on the current state

• Semi-Markov option: policy and termination condition may depend on the entire history of states, actions, and rewards since the initiation of the option
  • Options that terminate after a pre-specified number of time steps
  • Options that execute other options
Semi-Markov Options

• Let $H$ be the set of possible **histories** (segments of experience terminates in $s_\tau$, $\tau = t + k$)

\[ H = < s_t, a_t, r_t, s_{t+1}, \ldots, s_\tau > \]

• An semi-Markov option is represented as a triple:

\[ o = < I, \pi, \beta > \]

  • $I \subseteq S$ is the set of states in which $o$ may be started
  • $\pi: H \times A \rightarrow [0,1]$ is the policy followed during $o$
  • $\beta: H \rightarrow [0,1]$ is the probability of terminating in each state
Policy over Options

• Let $\mu$ be the policy over options. $\mu$ selects an option $o \in O_{s_t}$ according to probability distribution $\mu(s_t)$
  $$\mu: S \times O \rightarrow [0,1]$$
• $\mu$ determines a conventional policy over actions, or flat policy, $\pi = flat(\mu)$. 
Value functions for options

• Define $Q^\mu(s, o)$ the value of taking option $Q^\mu(s, o)$ in state $s$ under policy $\mu$, as
  
  $Q^\mu(s, o) \overset{\text{def}}{=} E\{r_t + \gamma r_{t+1} + \cdots | o \text{ initiated in } s \text{ at time } t, \mu \text{ followed after termination} \}$

  
  $Q^*(s, o) \overset{\text{def}}{=} \max_{\mu \in \Pi(O)} Q^\mu(s, o)$

• $\Pi(O)$ is the set of all policies selecting only from options in $O$
Options define a Semi-Markov Decision Process (SMDP)

- The state trajectory of an MDP is made up of discrete-time transitions and homogeneous discount.
- SMDP comprises larger, continuous-time transitions and discrete events and interval-dependent discount.
- Options enable an MDP trajectory to be analyzed in either way.  
  \[ \text{MDP} + \text{Options} = \text{SMDP} \]
SMDPs

• The amount of time between one decision and the next is a random variable $\tau$
• Transition probabilities $p(s', \tau|s, a)$
• Bellman equations

$$V^*(s) = \max_{a \in A_s} \left[ R(s,a) + \sum_{s', \tau} \gamma^\tau P(s', \tau|s, a) V^*(s') \right]$$

$$Q^*(s,a) = R(s,a) + \sum_{s', \tau} \gamma^\tau P(s', \tau|s, a) \max_{a' \in A_s} Q^*(s', a')$$
Option models

The reward of $o$:

- Let $\varepsilon(o, s, t)$ denote the event of $o$ being initiated in state $s$ at time $t$.

$$r_s^o = E\{r_t + \gamma r_{t+1} + \cdots + \gamma^{\tau-1} r_{t+\tau} | \varepsilon(o, s, t)\}$$

Transition probabilities:

- For all $s \in S$, $p(s', \tau)$ is the probability that the option terminates in $s$ after $\tau$ steps.

$$p_{ss'}^o = \sum_{\tau=1}^{\infty} p(s', \tau) \gamma^{\tau}$$
Bellman optimality Equation

\[ V_0^*(s) \overset{\text{def}}{=} \max_{o \in O_s} [r_s^o + \sum_s p(s'|s,o) V_0^*(s')] \]

\[ Q_0^*(s,o) \overset{\text{def}}{=} r_s^o + \sum_s p(s'|s,o) \max_{o' \in O_{s'}} Q_0^*(s',o'), \]

- Bellman optimality equations can be solved, exactly or approximately, using methods that generalize the usual **DP** and **RL** algorithms.
Illustration: Rooms Example

4 stochastic primitive actions

8 multi-step options
(to each room's 2 hallways)
Learning room-by-room is much faster than cell-by-cell
SMDP Q-learning backups

- At state $s$, initiate option $o$ and execute until termination
- Observe termination state $s'$, number of steps $\tau$, discounted return $r$

\[
Q_{k+1}(s, o) \overset{\text{def}}{=} (1 - \alpha_k)Q_k(s, o) + \alpha_k(r + \gamma^\tau \max_{o \in O_s} Q_k(s', o))
\]
Looking inside options

• SMDP methods apply to options, but only when they are treated as opaque indivisible units.

• *Interrupting* options before they would terminate naturally according to their termination conditions.
Intra-option Q-learning

On every transition:

\[ Q_{k+1}(s_t, o) = (1 - \alpha_k)Q_k(s_t, o) + \alpha_k \left[ r_{t+1} + \gamma U_k(s_{t+1}, o) \right] \]

where

\[ U_k(s, o) = (1 - \beta(s))Q_k(s, o) + \beta(s) \max_{o' \in O} Q_k(s, o') \]

is an estimate of the value of state-option pair \((s, o)\) upon arrival in state \(s\).
References


Options Learning

Options are typically learned using sub-goals and “pseudo-rewards”.

- Tabular cases (Wiering & Schmidhuber, 1997; Schaul et al., 2015, Ofir Nachum et al. 2018) utilize row states as sub-goals.
- Pre-defined sub-goals (Tessler et al., 2016; Kulkarni et al., 2016)
- **Options Discovery**
Option Discovery Algorithms

• policy gradient based methods:
  • The Option-Critic (Bacon et al., 2017)
  • Deep Discovery of Options (DDO) (Fox et al., 2017)
  • FeUdal Networks (Alexander et al., 2017)

• Information-theoretic based methods:
  • Variational Intrinsic Control (Gregor et al., 2016)
  • Diversity is All You Need (DIAYN) (Eysenbach et al., 2018)
  • (Florensa et al., 2017)

• Eigenoptions: (Machdo et al., 2017; Liu et al., 2017)

• Variational options discovery: VALOR (Achiam et al., 2018)
The option-critic
Insight

• Options can be learned end-to-end jointly with a policy-over-options using policy gradients.
• The policy (actor) is decoupled from its value function.
• The critic provides feedback to improve the actor
• Learning is fully online

• Parameterize internal policies and termination conditions
• Policy over options is computed by a separate process
The option-value function

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

- Where \( Q_U : S \times \Omega \times A \to \mathbb{R} \) is the value of executing an action in the context of a state-option pair

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

\[ U(\omega, s') = (1 - \beta_{\omega, \theta}(s')) Q_\Omega(s', \omega) + \beta_{\omega, \theta}(s') V_\Omega(s') \]

\( \omega \) – option \hspace{2cm} \( \pi_\Omega \) – policy over options

\( \pi_{\omega, \theta} \) – the intra-option policy \hspace{2cm} \( \beta_{\omega, \theta} \) – termination

\( U : \Omega \times S \to \mathbb{R} \) – the option-value function upon arrival,
Main result: Gradient updates

- 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.
FeUdal Networks for Hierarchical Reinforcement Learning
Insight

- Policy-over-options changing options at every step, i.e., high-level deviates from its own mission
- Levels of hierarchy within an agent communicate via explicit goals
FeUdal Networks (FUN, 1993)

- Proposed by Dayan & Hinton in 1993
- Let high-level managers set tasks to sub-managers, who learn how to satisfy those goals.
  - Sub-Managers learn to maximize their reinforcement in the context of the command
FuN model description

\[ z_t = f^\text{percept}(x_t) \]
\[ s_t = f^\text{Mspace}(z_t) \]
\[ h_t^M, \hat{g}_t = f^\text{Mrnn}(s_t, h_{t-1}^M); g_t = \hat{g}_t / ||\hat{g}_t|| \]
\[ w_t = \phi(\sum_{i=t-c}^{t} \hat{g}_i) \]
\[ h_t^W, U_t = f^\text{Wrnn}(z_t, h_{t-1}^W) \]
\[ \pi_t = \text{SoftMax}(U_t w_t) \]

\[ z: \text{Embedding of env. } x \]
\[ h^M: \text{Internal state of manager} \]
\[ h^W: \text{Internal state of worker} \]
\[ g: \text{Goal} \]
\[ w: \text{Embedding of goal } g \]
\[ c: \text{Prediction horizon} \]
\[ U: \text{Output of worker} \]
\[ \pi: \text{Vector of prob. over actions} \]

\[ f^\text{percept}: \text{CNN, 1st layer: 16 8x8 filters w/ stride 4, 2nd layer 32 4x4 filters w/ stride 2, fully connected layer has 256 hidden units.} \]
\[ f^\text{Mspace}: \text{Fully conn. layer, computes state space.} \]
\[ f^\text{Wrnn}: \text{Standard LSTM w/ 256 hidden units, computes goal.} \]
\[ f^\text{Mrnn}: \text{Dilated LSTM w/ 256 hidden units (will be explained detailed later).} \]
FuN model description

\[ s_t = \phi(x_t) \in R^d \]

- Complementary representations
- Multiple time scale
Learning

Bad idea:
• train feudal network end-to-end using a policy gradient algorithm operating on the actions taken by the Worker

Good idea:
• independently train Manager to predict advantageous directions in state space and to intrinsically reward the Worker to follow these directions
The agents goal

Maximize the discounted return

\[ R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]

• The agent’s behavior is defined by its action-selection policy \( \pi \). FuN produces a distribution over possible actions.
Manager’s transition policy

• Consider $g_t = g(s_t, \theta)$

$\mu(\cdot)$: High-level policy selecting among subpolicies

$o$: Sub-policy

• Transition policy:

$$\pi_{t}^{TP}(s_{t+\epsilon} | s_t) = p(s_{t+\epsilon} | s_t, \mu(s_t, \theta))$$

• Transition policy gradient:

$$\nabla_\theta \pi_t^{TP} = \mathbb{E}[(R_t - V(s_t))\nabla_\theta \log p(s_{t+\epsilon} | s_t, \mu(s_t, \theta))]$$

• It is assumed that,

$$p(s_{t+\epsilon} | s_t, \mu(s_t, \theta)) \propto e^{d_{\cos}(s_{t+\epsilon} - s_t, g_t)} \text{ (von Mises-Fisher distribution)}$$
Gradient for the goal:

\[ \nabla g_t = A_t^M \nabla \theta d_{\text{cos}}(s_{t+c} - s_t, g_t(\theta)) \]

where

\[ A_t^M = R_t - V_t^M(x_t, \theta) \] – advantage function

\[ V_t^M(x_t, \theta) \] – value function estimate from the internal critic

\[ d_{\text{cos}}(\alpha, \beta) = \alpha^T \beta / (||\alpha|| ||\beta||) \] – cosine similarity
Workers intrinsic reward

• Worker’s policy \( \pi \) is trained to maximize \( R_t + \alpha R_t^I \).

\[ r_t^I = \frac{1}{c} \sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_{t-i}) \] : Intrinsic reward

\( R_t \): Extrinsic discounted return

\( \alpha \): Hyperparameter to blend intrinsic and extrinsic reward

\( R_t^I \): Intrinsic discounted return

\( c \): horizon

• Worker policy gradient:

\[
\nabla \pi_t = A_t^D \nabla_\theta \log \pi(a_t | x_t; \theta)
\]

\[
A_t^D = (R_t + \alpha R_t^I - V_t^M(x_t, \theta))
\]
Diversity is All You Need: Learning Skills without a Reward Function
Insight

• Learning skills without reward
$Z \sim p(z)$ – a latent variable; A policy conditioned on a fixed $Z$ as a “skill”
How it works

\[ F(\theta) \triangleq I(S; Z) + H[A \mid S] - I(A; Z \mid S) \]

\[ = (H[Z] - H[Z \mid S]) + H[A \mid S] - (H[A \mid S] - H[A \mid S, Z]) \]

\[ = H[Z] - H[Z \mid S] + H[A \mid S, Z] \]

\( I (\cdot; \cdot) \) and \( H[\cdot] \) – mutual information and Shannon entropy
Maximize \( I(S; Z) \) – the skill should control which states the agent visits; the skill can be inferred from the states visited.
Minimize \( I(A; Z \mid S) \) – that states, not actions, are used to distinguish skills
Maximize \( H[A \mid S] \) – maximize the entropy of mixture policy
Implementation

\[ F(\theta) \triangleq I(S; Z) + \mathcal{H}[A \mid S] - I(A; Z \mid S) \]

\[ = (\mathcal{H}[Z] - \mathcal{H}[Z \mid S]) + \mathcal{H}[A \mid S] - (\mathcal{H}[A \mid S] - \mathcal{H}[A \mid S, Z]) \]

\[ = \mathcal{H}[Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[A \mid S, Z] \]

Encourage \( p(z) \) to have high entropy – Fix \( p(z) \) to be uniform
Minimize \( H[Z \mid S] \) – Add in to the reward function. \( r_z(s, a) \triangleq \log q_\phi(z \mid s) - \log p(z) \)
Maximize \( H[A \mid S, Z] \) – using soft actor critic.

As we cannot integrate over all states and skills to compute \( p(z \mid s) \) exactly, we approximate this posterior with a learned discriminator \( q_\phi(z \mid s) \).
Review

• Options
  • A generalization of actions

• SMDP
  • MDP + Options = SMDP
  • Temporal abstraction

• Options discovery
  • The option-critic
  • FeUdal
  • DIAYN
Thanks