Hierarchical Reinforcement Learning

Amir massoud Farahmand
Farahmand@SoloGen.net
Markov Decision Problems

- Markov Process: Formulating a wide range of dynamical systems
- Finding an optimal solution of an objective function
- [Stochastic] Dynamics Programming
- Planning: Known environment
- Learning: Unknown environment
\[ V^\pi(s) = E\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s, \pi \} \]

\[ V^\pi(s) = \sum_{a \in A_s} \pi(s, a)[R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^\pi(s')] \]

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

\[ Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a' \in A_{s'}} Q^*(s', a') \]
Reinforcement Learning (1)

• Very important Machine Learning method
• An approximate online solution of MDP
  - Monte Carlo method
  - Stochastic Approximation
  - [Function Approximation]
Reinforcement Learning (2)

- Q-Learning and SARSA are among the most important solutions of RL

\[ Q_{k+1}(s, a) = (1 - \alpha_k)Q_k(s, a) + \alpha_k [r + \gamma \max_{a' \in A_{s'}} Q_k(s', a')] \]
Curses of DP

• Curse of Modeling
  - RL solves this problem

• Curse of Dimensionality
  - Approximating Value function
  - Hierarchical methods
Hierarchical RL (1)

- Use some kind of hierarchy in order to ...
  - Learn faster
  - Need less values to be updated (smaller storage dimension)
  - Incorporate a priori knowledge by designer
  - Increase reusability
  - Have a more meaningful structure than a mere Q-table
Hierarchical RL (2)

- Is there any unified meaning of hierarchy? NO!
- Different methods:
  - Temporal abstraction
  - State abstraction
  - Behavioral decomposition
  - ...

Hierarchical RL (3)

- Feudal Q-Learning [Dayan, Hinton]
- Options [Sutton, Precup, Singh]
- MaxQ [Dietterich]
- HAM [Russell, Parr, Andre]
- HexQ [Hengst]
- Weakly-Coupled MDP [Bernstein, Dean & Lin, ...]
- Structure Learning in SSA [Farahmand, Nili]
- ...
Feudal Q-Learning

- Divide each task to a few smaller sub-tasks
- State abstraction method
- Different layers of managers
- Each manager gets orders from its super-manager and orders to its sub-managers
Feudal Q-Learning

• Principles of Feudal Q-Learning
  - **Reward Hiding**: Managers must reward sub-managers for doing their bidding *whether or not* this satisfies the commands of the super-managers. Sub-managers should just learn to obey their managers and leave it up to them to determine what it is best to do at the next level up.
  
  - **Information Hiding**: Managers only need to know the state of the system at the granularity of their own choices of tasks. Indeed, allowing some decision making to take place at a coarser grain is one of the main goals of the hierarchical decomposition. Information is hidden both downwards - sub-managers do not know the task the super-manager has set the manager - and upwards - a super-manager does not know what choices its manager has made to satisfy its command.
Feudal Q-Learning
Feudal Q-Learning
Options: Introduction

• People do decision making at different time scales
  - Traveling example

• It is desirable to have a method to support this temporally-extended actions over different time scales
Options: Concept

- Macro-actions
- Temporal abstraction method of Hierarchical RL
- Options are temporally extended actions which each of them is consisted of a set of primitive actions
- Example:
  - Primitive actions: walking NSWE
  - Options: go to \{door, cornet, table, straight\}
    - Options can be Open-loop or Closed-loop
- Semi-Markov Decision Process Theory [Puterman]
Options: Formal Definitions

\[ \langle I, \pi, \beta \rangle \]
- policy \( \pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1] \)
- termination condition \( \beta : \mathcal{S}^+ \rightarrow [0, 1] \)
- initiation set \( I \subseteq \mathcal{S} \)

\[ I = \{s : a \in \mathcal{A}_s\} \]
\[ \beta(s) = 1, \ \forall s \in \mathcal{S} \]
\[ \pi(s, a) = 1, \ \forall s \in I \]

\[ V^\pi(s) \overset{\text{def}}{=} E \{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid E(\pi, s, t) \} , \]
Options: Rise of SMDP!

- Theorem: MDP + Options = SMDP

\[
p_{ss'}^\pi = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}, \quad r_s^\pi = E\{r_{t+1} \mid s_t = s, a_t = a\},
\]

\[
p_{ss'}^o = \sum_{k=1}^{\infty} p(s', k) \gamma^k,
\]

\[
r_s^o = E\left\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} \mid \mathcal{E}(o_s, s, t)\right\}
\]
Options: Value function

\[ V^\mu(s) = E \left\{ r_{t+1} + \cdots + \gamma^{k-1}r_{t+k} + \gamma^k V^\mu(s_{t+k}) \mid \mathcal{E}(\mu, s, t) \right\} \]

\[ = \sum_{o \in \mathcal{O}_s} \mu(s, o) \left[ r^o_s + \sum_{s'} p^o_{ss'} V^\mu(s') \right] \]

\[ Q^\mu(s, o) = E \left\{ r_{t+1} + \cdots + \gamma^{k-1}r_{t+k} + \gamma^k V^\mu(s_{t+k}) \mid \mathcal{E}(o, s, t) \right\} \]

\[ = E \left\{ r_{t+1} + \cdots + \gamma^{k-1}r_{t+k} + \gamma^k \sum_{o' \in \mathcal{O}_s} \mu(s_{t+k}, o') Q^\mu(s_{t+k}, o') \mid \mathcal{E}(o, s, t) \right\} \]

\[ = r^o_s + \sum_{s'} p^o_{ss'} \sum_{o' \in \mathcal{O}_s} \mu(s', o') Q^\mu(s', o'). \]
Options:
Bellman-like optimality condition

\[
V^*_\phi(s) = \max_{o \in \mathcal{O}} E \left\{ r + \gamma^k V^*_\phi(s') \mid \mathcal{E}(o,s) \right\}
\]

\[
Q^*_\phi(s, o) = E \left\{ r + \gamma^k \max_{d' \in \mathcal{D}_{s+d}} Q^*_\phi(s', o') \mid \mathcal{E}(o,s) \right\}
\]

\[
Q(s, o) \leftarrow Q(s, o) + \alpha \left[ r + \gamma^k \max_{d' \in \mathcal{D}_{s'}} Q(s', o') - Q(s, o) \right]
\]
Options: A simple example

4 stochastic primitive actions
up
left → right
Fall 33% of the time
down

8 multi-step options
(to each room's 2 hallways)
Options: A simple example
Options: A simple example
Interrupting Options

• Option’s policy is followed until it terminates.

• It is somehow unnecessary condition
  - You may change your decision in the middle of execution of your previous decision.

• Interruption Theorem: Yes! It is better!
Interrupting Options:
An example

Landmarks Problem

Interrupted Solution
(474 Steps)

S

SMDP Solution
(600 Steps)

range (input set) of each run-to-landmark controller
Options: Other issues

• Intra-option \{model, value\} learning
• Learning each options
  – Defining sub-goal reward function
MaxQ

- MaxQ Value Function Decomposition
- Somehow related to Feudal Q-Learning
- Decomposing Value function in a hierarchical structure
MaxQ

Diagram of MaxQ algorithm with nodes and edges representing QGet, QPut, QPickup, QNavigateForGet(t), QNavigateForPut(t), QPutdown, MaxRoot, MaxGet, MaxPut, MaxNavigate(t), QNorth(t), QEast(t), QSouth(t), QWest(t), Pickup, Putdown, and a grid with positions R, G, Y, and B.
MaxQ: Value decomposition

\[ V_i^\pi(s) = V_{a_i}^\pi(s) + \sum_{s'} P_i^\pi(s'|s, a) V_{i'}^\pi(s') \]

\[ Q_i^\pi(s, a) = V_{a_i}^\pi(s) + C_i^\pi(s, a) \]

\[ V_i^\pi(s) = \begin{cases} 
Q_i^\pi(s, \pi_i(s)) & \text{if } i \text{ is composite} \\
\sum_{s'} P(s'|s, i) R(s'|s, i) & \text{if } i \text{ is primitive} 
\end{cases} \]

\[ C_i^\pi(s, a) = \sum_{s'} P_i^\pi(s'|s, a) V_{i'}^\pi(s') \]

\[ V_0^\pi(s) = V_{a_n}^\pi(s) + C_{a_{n-1}}^\pi(s, a_n) + \ldots + C_{a_1}^\pi(s, a_2) + C_0^\pi(s, a_1). \]
MaxQ: Existence theorem

Theorem 1 Let $\pi = \{\pi_i; i = 0, \ldots, n\}$ be a hierarchical policy defined over a full-state MAXQ graph, and let $i = 0$ be the root node of the graph. Then there exist values for $C^\pi_i$ (for internal Max nodes) and $V^\pi_i$ (for primitive, leaf Max nodes) such that $V^\pi_0(s)$ is the expected cumulative reward of following policy $\pi$ in state $s$.

- Recursive optimal policy.
- There may be many recursive optimal policies with different value function.
- Recursive optimal policies are not an optimal policy.
- If $H$ is stationary macro hierarchy for MDP $M$, then all recursively optimal policies w.r.t. have the same value.
MaxQ: Learning

\[ C_i(s, a) := (1 - \alpha_t(i))C_i(s, a) + \alpha_t(i)V_i(s') \]
\[ V_i(s) := (1 - \alpha_t(i))V_i(s) + \alpha_t(i)R(s'|s, i) \]

- **Theorem:** If M is MDP, H is stationary macro, GLIE (Greedy in the Limit with Infinite Exploration) policy, common convergence conditions (bounded V and C, sum of alpha is ...), then with Prob. 1, algorithm MaxQ-0 will converge!
MaxQ

- Faster learning: all states updating
  - Similar to “all-goal-updating” of Kaelbling
MaxQ: State abstraction

• Advantageous
  - Memory reduction
  - Needed exploration will be reduced
  - Increase reusability as it is not dependent on its higher parents

• Is it possible?!
MaxQ: State abstraction

- Exact preservation of value function
- Approximate preservation

Theorem 3 If the following two conditions hold, then the MAXQ graph with abstraction functions \( \chi_i(s, a) \) and \( \chi_i(s) \) can represent the value function of any policy \( \pi \) whose value function can be represented by the MAXQ graph with no abstraction functions:

(a) For all composite Max nodes \( i \), actions \( a \), states \( s \in S_i \), and distinct states \( s_1, s_2 \in \text{Result}_i(s, a) \) whenever \( C_i^\pi(s_1, a) \neq C_i^\pi(s_2, a) \) it is the case that \( \chi_i(s_1, a) \neq \chi_i(s_2, a) \)

(b) For all primitive Max nodes \( i \) and distinct states \( s_1 \) and \( s_2 \), whenever \( V_i(s_1) \neq V_i(s_2) \) it is the case that \( \chi_i(s_1) \neq \chi_i(s_2) \).
MaxQ: State abstraction

• Does it converge?
  - It has not proved formally yet.

• What can we do if we want to use an abstraction that violates theorem 3?
  - Reward function decomposition
    • Design a reward function that reinforces those responsible parts of the architecture.
MaxQ: Other issues

• Undesired Terminal states
• Non-hierarchical execution (polling execution)
  - Better performance
  - Computational intensive
Learning in Subsumption Architecture

• Structure learning
  - How should behaviors arranged in the architecture?

• Behavior learning
  - How should a single behavior act?

• Structure/Behavior learning
SSA: Purely Parallel Case

- manipulate the world
- build maps
- explore
- avoid obstacles
- locomote

sensors
SSA: Structure learning issues

- How should we represent structure?
  - Sufficient (problem space can be covered)
  - Tractable (small hypothesis space)
  - Well-defined credit assignment

- How should we assign credits to architecture?
SSA: Structure learning issues

• Purely parallel structure
  - Is it the most plausible choice (regarding SSA-BBS assumptions)?

• Some different representations
  - Beh. learning
  - Beh/Layer learning
  - Order learning
SSA: Behavior learning issues

• Reinforcement signal decomposition: each Beh. has its own reward function

• Reinforcement signal design: How should we transform our desires into reward function?
  - Reward Shaping
  - Emotional Learning
  - …?

• Hierarchical Credit Assignment
SSA: Structure Learning example

- Suppose we have correct behaviors and want to arrange them in an architecture in order to maximize a specific behavior.
- Subjective evaluation: *We want to lift an object to a specific height while its slope does not become too high.*
- Objective evaluation: *How should we design it?!*
SSA: Structure Learning example
SSA: Structure Learning example