Introduction to Hierarchical Reinforcement Learning

Yuan Gao

University of Helsinki

gaoyuankidult@gmail.com

September 22, 2015

Overview

Markov Decision Process

A Markov decision process is a 4-tuple denoted as $(\mathbb{S},\mathbb{A},T\cdot(\cdot,\cdot),R\cdot(\cdot,\cdot))$
In this tuple

- $\mathbb{S}$ is a finite set of states. It describes all possible states of the space.
- $\mathbb{A}$ is a finite set of actions. It describes all possible states that robot can take.
- $T\cdot(\cdot,\cdot)$ is a function that takes three arguments. e.g $T_{a_t}(s_t, s_{t+1})$ means probability of transferring from s_t to s_{t+1} with action a_t.
- $R\cdot(\cdot,\cdot)$ is also a function that takes three arguments. e.g $R_{a_t}(s_t, s_{t+1})$ means reward of transferring from s_t to s_{t+1} with action a_t.

Policy Function

Policy function $\pi(s) : s \rightarrow a$ to map every state s with action a so that a cumulative reward R is maximized, where $0 \leq \gamma \leq 1$ is a discount factor.

Value Function

We define two functions to further describe this process. First function is statevalue function denoted as $V^{\pi}(\vec{x})$. This means expected value of an agent that follows an policy of π with initial value $\vec{s}$. It characterizes the rewards of following a policy π. Mathematically, it is defined as:

$$V^{\pi}(\vec{s}) = \vec{E}_{\tau}\{\sum_{t=0}^{\infty} \gamma^{t} r_{t} | \vec{s} = \vec{s_0}\} \tag{1}$$

where τ stands for trajectory of agent.

$$Q^{\pi}(\vec{s}, \vec{a}) = \vec{E}_{\tau}\{\sum_{t=0}^{\infty} \gamma^{t} r_{t} | \vec{s} = \vec{s_0}, \vec{a} = \vec{a_0}\} \tag{2}$$

HRL

- Why do we need HRL ?

HRL splits the global goals of a reinforcement learning agent up into smaller subgoals, and then attempts to tackle each subgoal separately.

Concepts of HRL

- Hierarchical Decomposition

Many large problems have hierarchical structure that allows them to be broken down into sub problems. The sub-problems, being smaller are solved more easily.

- Semi Markov Decision Process (SMDP)

SMDP is a generalization of the Markov Decision Process (MDP) where the times between transitions are allowed to be random variables whose distribution may depend upon the current state, the action taken, (possibly) the next state.

Concepts of HRL

- State Abstraction

Methods of this category describe how a problems state space can be divided up into a series of intervals if the states in the same interval block have the same properties in terms of transitions and rewards.

- Abstract Actions

Approaches to HRL employ actions that persist for multiple time-steps. These temporally extended or abstract actions hide the multi-step state-transition and reward details from the time they are invoked until termination.

Concepts of HRL

- Optimality of HRL
  - Hierarchical Optimality

    Policies that are hierarchically optimal are ones that maximize the overall value function consistent with the constraints imposed by the task-hierarchy.

  - Recursive Optimality

    sub-task policies to reach goal terminal states are context free ignoring the needs of their parent tasks.

  - Hierarchical Greedy Optimality

    In Hierarchical Greedy Optimality, a subtask policy proceeding to termination may be sub-optimal and by constantly interrupting the sub-task a better sub-task may be chosen.

Approaches of HRL

- Options, a formalization of abstract actions.
  Options approach is to define each subtask in terms of a fixed policy that is provided by the programmer (or that has been learned in some separate process).

- HAMQ, a partial program approach.
  In the hierarchy of abstract machines (HAM) approach to HRL the designer specifies abstract actions by providing stochastic finite state automata called abstract machines that work jointly with the MDP

Approaches of HRL

- MAXQ value function decomposition including state abstraction. The third approach MAXQ is to define each subtask in terms of a termination predicate and a local reward function

If some behaviors are completely specified use Options; if some behaviors are partially specified use HAM; if less domain knowledge available use MAXQ. We can use all three to specify different behaviors in tandem.

Problems of HRL

- Specify subtasks
  Hierarchical reinforcement learning involves breaking the target Markov decision problem into a hierarchy of sub problems or subtasks. The three general approaches to defining these subtasks are Options, Ham and MAXQ as mentioned in above section III.

- Employ state abstractions within subtasks
  The second design issue is whether to employ state abstractions within subtasks. A subtask employs state abstraction if it ignores some aspects of the state of the environment.

Problems of HRL

- Non-hierarchical execution
  The third design issue concerns the non-hierarchical execution of a learned hierarchical policy.

- Learning algorithm
  The final issue is what form of learning algorithm to employ. Finding online algorithms that work for general hierarchical reinforcement learning has been difficult, particularly within the termination predicate family of methods.

Algorithms of HRL

- H-DYNA Algorithm
  DYNA is a reinforcement learner that uses both real and simulated experience after building a model of the reward and state transition function.

- Hierarchical Exit Q Function (HEXQ)
  The HEXQ (hierarchical exit Q function) approach is a series of algorithms motivated by MAXQ value-function decomposition and bottom-up structure learning.

Algorithms of HRL

- Hierarchical Assignment of Behaviors by Self-organizing (HABS)
  It was developed to overcome the action explosion problem in HASSLE and to allow neural networks to be used as function approximator for the high level policy.

- HASSEL Algorithm.
  It discovers subgoals and the corresponding specialized subtask solvers by learning how to transition between abstract states. In the process subgoal abstract actions are generalized to be reused or specialized to work in different parts of the state-space or to reach different goals.

Flat RL vs HRL

Advantages of HRL over flat RL:

1. Improved exploration as it can take big steps at high levels of abstraction.
2. Learning from fewer trials as fewer parameters must be learned and subtasks can ignore irrelevant features of full state.
3. Faster learning for new problems because subtasks learned on previous problems can be reused
4. Allows transfer at multiple levels of hierarchy, which can speed up learning.
5. Task decompositions are helpful in reducing the size of the problem, and therefore in exorcising the Curse of Dimensionality

Flat RL vs HRL

Limitations of HRL:

1. HRL automatically handles exploration exploitation shift but at the same time it lacks in sufficient exploration and sufficient subtlety.
2. Some of the existing algorithms only work well for the problem which they were designed to solve.

Deep RL Examples

- End-to-end Deep Visuomotor Policy
- Deep Q Network
- Auto-encoder for State Abstraction