Hierarchical iPOMDPs for MTRL

Momchil

December 4, 2017

1 Background

Define MDPs, RL framework POMDPs Multi-task RL HRL = sMDPs Option discovery as hidden state inference POMDP formulation for one environment Multi-task POMDP

1.1 Reinforcement Learning and Markov Decision Processes

In reinforcement learning (RL), tasks are traditionally represented as Markov decision processes (MDPs) [1]. We define an MDP as a tuple $(\mathcal{S}, \mathcal{A}, T, R, I, \mathcal{F})$ where:

- $\mathcal{S}$ is a set of states that the agent can occupy,
- $\mathcal{A}$ is a set of actions that the agent can perform in those states,
- $T(\cdot|s,a)$ is a distribution of next states, such that $T(s'|s,a)$ is the probability that the agent ends up in state $s' \in \mathcal{S}$ after executing action $a \in \mathcal{A}$ in state $s\mathcal{S}$,
- $R(s,a)$ is the amount of reward an agent obtains after executing action a in state s,
- $I(\cdot)$ is a distribution of initial states in which the agent may start the task,
- $\mathcal{F}$ is a set of terminal states in which the task is completed.

The agent begins the task in state $s_0 \sim I$. At each time step t, the agent performs action a_t that takes it from state s_t to state $s_{t+1} \sim T(\cdot|s_t, a_t)$ and receives reward $r_t \sim R(s_t, a_t)$. The agent is said to follow a policy π if it chooses its actions according to $a_t \sim \pi(\cdot|s_t)$.

For a fixed policy π, the state-value function $V_\pi(s)$ represents the total expected discounted reward for each state s. It is the sum of all future rewards that the agent will receive, on average, if it starts in state s and follows π:

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s \right],$$

where γ is the discount rate which determines the present value of future rewards. The goal of the agent is to select an optimal policy π^* that maximizes $V_{\pi^*}(s)$. The optimal value function $V_{\pi^*}(s)$ satisfies the Bellman optimality equation [2]:

$$V_{\pi^*}(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s'} V_{\pi^*}(s')\ T(s'|s,a) \right\}$$

The Bellman equation serves as a basis of a number of algorithms for solving optimal control problems.

RL algorithms often fall into one of two broad categories: model-free algorithms, in which the agent learns V_π directly; and model-based algorithms, in which the agent learns R and T to compute V_π. Computational cognitive neuroscience has borrowed this dichotomy as a formal description of the distinction between habitual and goal-directed behaviors [3]. Model-free accounts of habitual behaviors posit that animals learn automatic responses to stimuli if they tend to lead to rewarding outcomes. Model-based accounts of goal-directed behaviors posit that animals learn the statistics of the environment independently of the rewards.

Model-free accounts are less computationally intensive, however they require many training examples and have to be re-trained whenever the reward distribution changes. This is consistent with empirical observations that overtrained animals exhibit fast reaction times and rigid behavioral patterns that tend to persist even reward contingencies change. Conversely, model-based accounts predict that agents will flexibly adapt to changes in reward distributions, at the cost of slower reaction times due to the greater computational complexity. Model-based accounts explain phenomena such as latent learning and accord with the intuition that people don't have to learn a new policy, say, every time they go shopping for different groceries.

1.2 Hierarchical Reinforcement Learning and Semi-Markov Decision Processes

A long-standing challenge for traditional RL is the combinatorial explosion that occurs when planning and learning take place over long time horizons. This challenge been addressed by hierarchical reinforcement learning (HRL), which breaks down the problem into sub-problems at multiple levels of abstraction.

One strand of HRL extends the agent's action repertoire to include *options* [4] – temporally-extended sequences of actions, sometimes referred to as subroutines, partial policies, or macro-actions. Each option consists of a sequence of actions that are executed as a single behavioral unit. Actions in the original MDP are referred to as *primitive actions*, in order to distinguish them from options.

Formally, including options turns the underlying MDP into a discrete-time semi-Markov decision process (SMDP), which allows transitions between states to take variable amounts

of time. Thus we augment the original MDP with the following definitions:

- $\mathcal{O}$ is a set of options,
- π_o is a policy for each option $o \in \mathcal{O}$, such that while o is being executed, actions are selected according to $a \sim \pi_o(\cdot|s)$,
- $\mathcal{F}_o$ is a set of subgoal states for each option o, such that π_o terminates when reaching any one of those states,
- $T(\cdot, \cdot|s, o)$ is a joint distribution of terminal states and step counts for each option o, such that $T(s', t|s, o)$ is the probability that the agent ends up in subgoal state $s' \in \mathcal{F}_o$ exactly t time steps after executing option $o \in \mathcal{O}$ in state $s \in \mathcal{S}$,
- $R(s, o)$ is the total expected discounted reward for starting in state s and executing option o.

The rewards can be computed as:

$$R(s, o) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_t \sim \pi_o\right]$$

It is also convenient to define a discounted distribution of terminal states [4]:

$$T_\gamma(s'|s, o) = \sum_{t=0}^{\infty} T(s', t|s, a)\gamma^t$$

In the SMDP, the agent follows a policy μ defined over options, such that options are chosen according to $o_t \sim \mu(\cdot|s_t)$. The Bellman equation for the optimal options policy μ^* thus becomes:

$$V_{\mu^*}(s) = \max_o \left\{ R(s, o) + \sum_{s'} V_{\mu^*}(s')\ T_\gamma(s'|s, o) \right\}$$

An options policy μ in the SMDP uniquely determines a *flat policy* π consisting solely of primitive actions in the original MDP. Thus any solution of the SMDP can be mapped back to a solution in the original MDP.

Introducing options allows agent to "jump" to distant subgoal states and automatically execute all primitive actions required to get there, without performing any additional computations. If the subgoals and the options policies are chosen appropriately, exploration

and planning can be performed much more efficiently by allowing the agent reason over a short sequence of options, rather than a long sequence of primitive actions.

The Bellman equation thu

Options

The *flat* policy

Flat policy

challenges addressed cognitive neuroscience (botvinick)

$$
\begin{aligned}
\bar{T} &\sim \text{GEM}(...) \\
T(.|s) = T_s &\sim \text{DP}(..., \bar{T}) \\
\phi_s &\sim H \\
s_t &\sim T_{s_{t-1}} \\
o_t &\sim F(\phi_{s_t})
\end{aligned}
$$

1.3 hierarchical iHMM as HDP

$$
\begin{aligned}
\beta &\sim \text{GEM}(...) \\
z_s &\sim \text{Mult}(\beta) \\
\bar{T} &\sim \text{GEM}(...) \\
\bar{T}_c &\sim \text{DP}(..., \bar{T}) \\
T(.|s) = T_s &\sim \text{DP}(..., \bar{T}_{z_s}) \\
\phi_s &\sim H \\
s_t &\sim T_{s_{t-1}} \\
o_t &\sim F(\phi_{s_t})
\end{aligned}
$$

$$
\begin{aligned}
H_c &\sim ... \\
\phi_s &\sim H_{z_s}
\end{aligned}
$$

1.4 hierarchical iHMM as HDP, version 2

$$
\begin{aligned}
\bar{T} &\sim \text{GEM}(...) \\
T(.|c) = T_c &\sim \text{DP}(..., \bar{T}) \\
\bar{T}_{c,\cdot} &\sim \text{GEM}(...) \\
T(. \in c | s \in c) = T_{c,s} &\sim \text{DP}(..., \bar{T}_{c,\cdot}) \\
T_s(s') = T(s'|s) &= \begin{cases} T_{c,s}(s')\ T_c(c) & \text{if } s, s' \in c \\ \bar{T}_{c',\cdot}(s')\ T_c(c') & \text{if } s \in c, s' \in c' \end{cases} \\
\phi_s &\sim H \\
s_t &\sim T_{s_{t-1}} \\
o_t &\sim F(\phi_{s_t})
\end{aligned}
$$

1.5 hierarchical iHMM as HDP, version 3

$$
\begin{aligned}
\beta &\sim \text{GEM}(...) \\
z_c &\sim \text{Mult}(\beta) \\
\bar{T}_{g,\cdot} &\sim \text{GEM}(...) \\
T_{g,s} &\sim \text{DP}(..., \bar{T}_{g,\cdot}) \\
\bar{T} &\sim \text{GEM}(...) \\
T(.|c) = T_c &\sim \text{DP}(..., \bar{T}) \\
\bar{T}_{c,\cdot} &= \bar{T}_{g=z_c,\cdot} \\
T(. \in c | s \in c) = T_{c,s} &= T_{g=z_c,s} \\
T_s(s') = T(s'|s) &= \begin{cases} T_{c,s}(s')\ T_c(c) & \text{if } s, s' \in c \\ \bar{T}_{c',\cdot}(s')\ T_c(c') & \text{if } s \in c, s' \in c' \end{cases} \\
\phi_s &\sim H \\
s_t &\sim T_{s_{t-1}} \\
o_t &\sim F(\phi_{s_t})
\end{aligned}
$$

Bibliography & References Cited

[1] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[2] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

[3] Ray J Dolan and Peter Dayan. Goals and habits in the brain. *Neuron*, 80(2):312–325, 2013.

[4] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181 – 211, 1999.