Reinforcement Learning

Hanxiao Liu
Carnegie Mellon University
hanxiaol@cs.cmu.edu

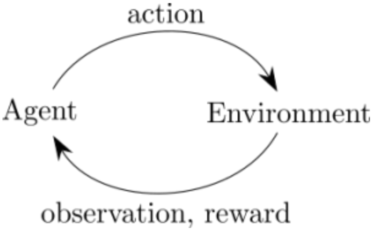
September 20, 2016

[0] Based on David Silver's lectures on RL.

Outline

Examples

1. Helicopter Control
2. Atari Games
3. Learning Simple Algorithms



Basic Setups

History: Agent's experience

$$H_t = O_1, R_1, A_1, O_2, R_2, A_2, \ldots, A_{t-1}, O_t, R_t \tag{1}$$

State: A summary of the history

$$S_t = f(H_t) \tag{2}$$

Markov Property

$$\Pr\left[S_{t+1} | S_t\right] = P\left[S_{t+1} | S_1, \ldots, S_t\right] \tag{3}$$

Basic Setups

Key components of a RL agent

- Policy: Behavior of the agent

$$a = \pi(s) \tag{4}$$

$$\pi(a|s) = \Pr\left[A_t = a|S_t = s\right] \tag{5}$$

- Value Function: A prediction of future reward

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s\right] \tag{6}$$

- Model: Agent's representation of the environment.
  - We will primarily focus on model-free RL.

Basic Setups

Fundamental sequential decision making problems:

- Planning: Fully observed environment.
- RL: The environment is initially unknown.
  - Exploration and exploitation.

Tasks

- Prediction: Evaluate $v_\pi(s)$ given π.
- Control: Optimize $v_*(s)$ by refining π.

Outline

Introduction

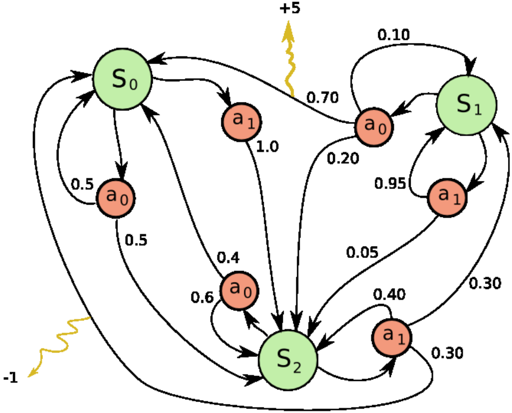
Markov Decision Process

Model-Free Prediction

Model-Free Control

Function Approximation

Markov Decision Process



Markov Decision Process

A MDP $\mathcal{M}$ is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

$$\mathcal{P}_{ss'}^{a} = \Pr\left[S_{t+1} = s' | S_t = s, A_t = a\right] \tag{7}$$

$$\mathcal{R}_{s}^{a} = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right] \tag{8}$$

- Here we assume the environment has been fully observed—$\mathcal{P}$ and $\mathcal{R}$ are known.
- For any fixed policy, $(\mathcal{S}, \mathcal{P}^{\pi})$ defines a Markov Process.

$$\mathcal{P}_{ss'}^{\pi} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^{a} \tag{9}$$

Value Functions

Goal: Refining π to maximize future returns

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \tag{10}$$

Value functions

- State-value function

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right] \tag{11}$$

- Action-value function

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] \tag{12}$$

Bellman Expectation Equation

Q: How to evaluate any given π (obtain v_π)?

Value = immediate reward + discounted successor value

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right] \quad (13)$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right] \quad (14)$$

$$v_\pi(s) = \sum_a \pi(a|s) \Big[\underbrace{\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_\pi(s')}_{q_\pi(s,a)} \Big] \quad (15)$$

$\Longrightarrow$ v_π can be obtained by solving a linear system.

Bellman Optimality Equation

Q: How do we know if π is already the optimal?

Recall for any fixed π:

$$v_\pi(s) = \sum_a \pi(a|s)\Big[\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_\pi(s')\Big] \tag{16}$$

For the optimal π^*:

$$v_{\pi^*}(s) = \max_a \;\; \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_{\pi^*}(s') \tag{17}$$

No closed-form solution, but iterative solvers are available.

Control

Q: How to improve π?

Policy iteration

(a) Policy evaluation: compute $v_\pi(s)$ given π.

(b) Getting an improved policy $\pi' = \text{greedy}(v_\pi)$

- $v_\pi(s) \implies q_\pi(s, a)$
- $\pi'(s) = \text{argmax}_a \; q_\pi(s, a)$
- Theorem: $\pi' \succeq \pi$.

Alternative approach: Value iteration

(a) Obtain $v^*(s)$ by solving Bellman optimality equation.

(b) π^* is implied by v^*.

Outline

Model-Free Prediction

So far we've been assuming $\mathcal{M}$ is fully observed.

- We are informed about $\mathcal{P}$ and $\mathcal{R}$, so estimating $v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right]$ is easy.

Can $v_\pi(s)$ still be estimated if $\mathcal{M}$ is partially observed?

- Taking the full exception is not possible.
- However, we can sample from the environment.

Sampling Approaches

Monte Carlo (MC)

1. Estimate G_t by sampling from $\mathcal{M}$ following π.
   - We are sampling from a Markov (Reward) Process.
2. $v_\pi(S_t) \leftarrow v_\pi(S_t) + \alpha(G_t - v_\pi(S_t))$

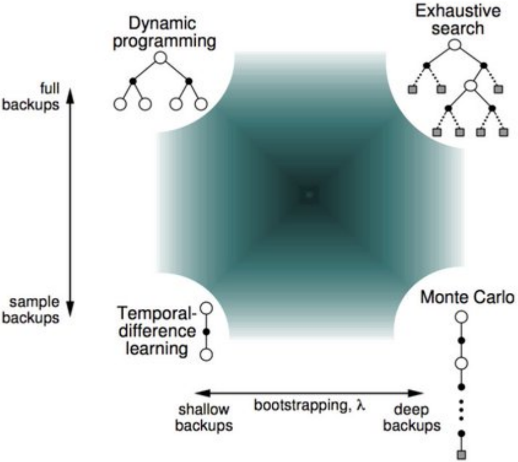
Temporal Difference (TD)

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \tag{18}$$

Advantages of TD

- Lower variance.
- More efficient (by exploit the Markov property).
- Handle incomplete sequences.

A Unified View



[0] http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MC-TD.pdf

Outline

Model-Free Control

Policy Iteration

1. Policy evaluation: compute $Q_\pi(s, a)$ given π.
   - using TD or MC.
2. Policy refinement: $\pi' = \epsilon\text{-greedy}(Q_\pi)$
   - $\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & a^* = \text{argmax}_a \ Q(s, a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$
   - Pure greedy is a bad idea—no exploration.

Outline

Large-scale RL

Some real-world problems:

- Go: 10^{170} states
- Robot control: continuous (infinite) action space

Size of $v(s)$ and/or $q(s, a)$ becomes intractable.

Solution: Function approximation

$$\hat{v}(s, w) \sim v_\pi(s) \tag{19}$$

$$\hat{q}(s, a, w) \sim q_\pi(s, a) \tag{20}$$

Ideally $\hat{v}$ and $\hat{q}$ are both differentiable and expressive

- deep neural networks

Optimization

Recall in TD:

$$\Delta v = \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t)) \tag{21}$$

With function approximation:

$$\Delta w = \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)) \nabla_w \hat{v}(S_t, w) \tag{22}$$

$\approx$ supervised learning using stochastic gradient descent

- Experience reply (in DQN): cache and reuse historical training examples to refine w.

Policy-based RL

Alternatively, we can parameterize π instead of v_π (or q_π)

$$\pi_\theta(s, a) = \Pr\left[a | s, \theta\right] \tag{23}$$

Then optimize θ w.r.t. some objective $J(\theta)$.

Advantage: no need to carry out $\text{argmax}_a q(s, a)$

- More efficient for high-dimensional/continuous $\mathcal{A}$.

Policy Gradient

Consider a one-step MDP starting from $s \sim d(s)$

$$J(\theta) := \mathbb{E}_{\pi_\theta}[r] = \sum_s d(s) \sum_a \pi_\theta(s,a)\mathcal{R}_s^a \tag{24}$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_s d(s) \sum_a \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\mathcal{R}_s^a & (25)\\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) r\right] & (26)\end{aligned}$$

The above allows us to access stochastic policy gradient without knowing the environment.

Actor-Critic Models

Policy gradient in more generic cases

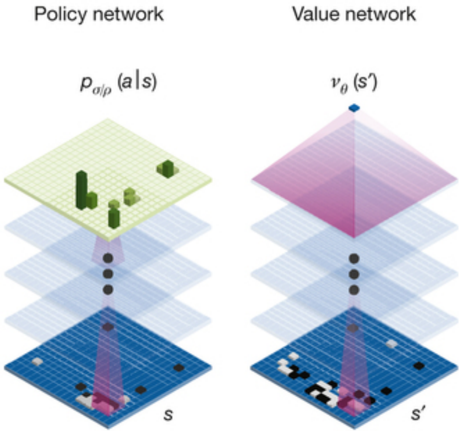
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi(s,a) q_{\pi_\theta}(s,a)\right] \tag{27}$$

$q_{\pi_\theta}(s,a)$ can be approximated by $\hat{q}(s,a,w)$.

Critic updates w.

Actor updates θ based on critic's suggestion.

Similar ideas in AlphaGo



The End

Other interesting topics

- Convergence.
- Exploration v.s. Exploitation.
- Credit assignment.
- Off-Policy RL.