Reinforcement Learning Solutions

Minsuk Chang

January 8, 2022

Chapter 1

Introduction

Exercise 1.1

Probability will cause them to learn differently, because every movement they choose will be different by time to time.

Exercise 1.2

Reducing the number of stages will be very beneficial, because number of stages will be reduced.

Exercise 1.3

It max maximize temporal rewards, but will work poorly in long terms.

Exercise 1.4

Exploratory moves make the model learn and converge fast to become optimal, but remains suboptimal because of the probability of exploration.

Exercise 1.5

Learn from games which are already played by professional players.

Chapter 2

Multi Armed Bandits

Exercise 2.1

$P = (0.5 * P(greedy)) + (0.5 * P(Random)) = (0.5 * 1) + (0.5 * 0.5) = 0.75$

Exercise 2.2

A	1	2	2	2	3
R	-1	1	-2	2	0

Must have occurred: After 3, 4
Might have occurred : All intervals

Exercise 2.3

$\epsilon = 0.01$
If infinite time passes, 99%of actions chosen will be optimal.

Exercise 2.4

Chapter 3

Finite Markov Decision Processes

Exercise 3.1

Chess, Go, Tic-Tac-To

Exercise 3.2

Games like slot machine which is uniform random and each state isn't dependent on previous states.

Exercise 3.3

1. Accelerator, steering wheel, and brake.
2. The boundary is drawn where the agent's absolute control reaches.
3. It may be free choice, but closer to the boundary, more efficient the learning process would be.

Exercise 3.4

Exercise 3.5

$S^+ \neq S$ in this notation.
$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1$, for all $s \in S, a \in A(s)$

probability is 0 to move to other states from the terminal state.
$p(s', r|s, a) = 0$, for all $s \in S, a \in A(s), s' \in S^+ - S$

Chapter 4

Dynamic Programming

Exercise 4.4

1. Initialization

   $V(s) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S; V(terminal) = 0$

2. Policy Evaluation

   Loop:

   $\Delta \leftarrow 0$

   Loop for each $s \in S$:

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$

   $\Delta \leftarrow max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$(a small positive number determining the accuracy of estimation)

3. Policy Improvement

   policy-stable $\leftarrow$ *true*

   For each $s \in S$:

   old-action $\leftarrow \pi(s)$

   $\pi(s) \leftarrow argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

   $maxval \leftarrow max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

   If *old-action* $\neq \pi(s), maxval > V(s)$, then *policy-stable* $\leftarrow false$

   If *policy-stable*, then stop and return $V \approx v_* and \pi \approx \pi_*$; else go to 2

Exercise 4.5

1. Initialization

   $Q(s, a) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S; Q(terminal, a) = 0$ for all $a \in A(terminal)$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in S$:

Loop for each $a \in A(a)$:

$q \leftarrow Q(s,a)$

$Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))[r + \gamma \sum_{a'} \pi(a'|s')Q(s',a')]$

$\Delta \leftarrow max(\Delta, |q - Q(s,a)|)$

until $\Delta < \theta$(a small positive number determining the accuracy of estimation)

3. Policy Improvement

policy-stable $\leftarrow true$

For each $s \in S$:

For each $a \in A(s)$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow argmax_a Q(s,a)$

$maxval \leftarrow max_a Q(s,a)$

If *old-action* $\neq \pi(s), maxval > Q(s, \textit{old-action})$, then *policy-stable* $\leftarrow false$

If *policy-stable*, then stop and return $Q \approx q_*$ and $\pi \approx \pi_*$; else go to 2

Exercise 4.6

3 : The actions extracted by the policy would be depending on probability. This means that to improve the policy, we need to adjust the probability of actions to be selected on such state, but maintaining its minimum. When one of the action's probability reaches the minimum, then the algorithm needs to end.

2: To reach convergence, $\pi(s)$ needs to be replaced to $argmax_a \pi(s)$

1: $\pi(s)$ is no longer an element but the probability of the actions of $A(s)$.