Reinforcement Learning Solutions

Minsuk Chang

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 $s \in S, a \in A(s), s' \in S^+ - S$

Exercise 3.6

1. Always return 0 except last state which returns 1.
2. The expected return is always -1.

Exercise 3.7

The robot won't get any penalty for staying long in maze.
No, solving the maze in the fastest time must be the goal.

Exercise 3.8

$G_5 = 0$, $G_4 = 1$ Others are pure calculation.

Exercise 3.9

$G_1 = 70$, $G_0 = 65$

Exercise 3.10

The formula is from the sum of geometric series.

Exercise 3.11

$\sum_{r,s'} \sum_a rp(s'r|S_t, a)\pi(a|S_t)$

Exercise 3.12

$v_\pi(s) = \sum_a \pi(a|s)q_\pi(s)$

Exercise 3.13

$$\begin{aligned} q_\pi(s,a) &= E_\pi[G_t|S_t = s, A_t = a] \\ &= \sum_{r,s'} p(s'r|s,a)(r + \gamma v_\pi(s')) \end{aligned}$$

Exercise 3.14

$$\frac{1}{4} * 0.9 * (2.3 + 0.4 + 0.4 + 0.7) = \frac{27}{40} = 0.675 \approx 0.7$$

Exercise 3.15

$$G'_t = G_t + \frac{c}{1-\gamma}$$
$$V_c = \frac{c}{1-\gamma}$$

Exercise 3.16

For the maze problem, traveling back and forth might maximize the reward, because the reward for reaching any position in the maze may be positive. This may lead to the same problem as Example 3.7

Exercise 3.17

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma \sum_{a'} \pi(a'|s')q(s',a'))$$

Exercise 3.18

$$\begin{aligned} v_\pi(s) &= E[q(S_t+1,a)|S_t = s] \\ &= \sum_a \pi(a|s)q(s,a) \end{aligned}$$

Exercise 3.19

$$\begin{aligned} q_\pi(s,a) &= E[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \\ &= \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s')) \end{aligned}$$

Exercise 3.20

0 : hole
-1 : all over the green land
-2 : range(driver) far from -1 (including bunker)
-3 : range(driver) far from -2
...

Exercise 3.21

0 : hole
-1 : all over the green land
-2 : range(putter) far from -1 (excluding bunker)

-3 : range(driver) far from -2 (including bunker)
...

Exercise 3.22

$\gamma = 0$: left $\gamma = 0.9$: right $\gamma = 0.5$: both same

Exercise 3.23

Exercise 3.24

$$10 * (1 + 0 + 0 + 0 + 0 + 0.9^5 + 0 + 0 + 0 + 0 + 0.9^{10} + ...)$$
$$= \frac{10}{1 - 0.9^5}$$
$$= 24.41942... \approx 24.4$$

Exercise 3.25

$v_*(s) = max_a q_*(s, a)$

Exercise 3.26

$q_*(s, a) = max_{s'} v_*(s')$ for $p(s', r|s, a) > 0$

Exercise 3.27

$\pi_*(a|s) = 1$ if $a = argmax_a q_*(a, s)$ or else 0

Exercise 3.28

$\pi_*(a|s) = 1$ if $a = argmax_a v_*(s')$ for $p(s', r|s, a) > 0$ or else 0

Exercise 3.29

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')]$$

$$q_\pi(s,a) = \sum_{s'} p(s'|s,a)(r(s,a) + \gamma \sum_{a'} \pi(a'|s') q(s',a'))$$

$$v_* i(s) = max_a \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_*(s')]$$

$$q_*(s,a) = \sum_{s'} p(s'|s,a)(r(s,a) + \gamma max_a q_*(s',a'))$$

Chapter 4

Dynamic Programming

Exercise 4.1

$q_\pi(11, down) = -1 + 0 = 0$
$q_\pi(7, down) = -1 + v_\pi(11) = -15$

Exercise 4.2

$$v_\pi(15) = \frac{1}{4}(v_\pi(15) - 22 - 20 - 14) - 1 \tag{4.1}$$

$$\begin{aligned} v_\pi(15) &= \frac{1}{4}(v_\pi(15) + v_\pi(13) - 20 - 14) - 1 \\ v_\pi(13) &= \frac{1}{4}(v_\pi(15) - 56) - 1 \\ v_\pi(15) &= -20 \end{aligned} \tag{4.2}$$

Exercise 4.3

$$q_{k+1}(s, a) = \sum_{s', r} p(s', r|s, a)(r + \gamma \sum_{a'} \pi(a'|s') q_k(s', a'))$$

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

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

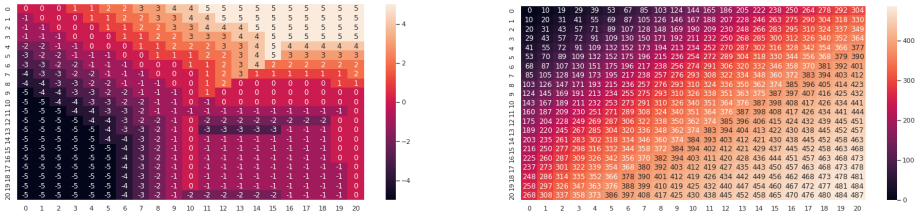
Exercise 4.6

3 : Policy is not $\pi(s)$ but $\pi(a|s)$. So, the update of $\pi(a|s)$ should be done by approximating to $P = softmax(Q(s,a))$ for all $a \in A(s)$

2: The formula to calculate $V(s)$ must be changed to
$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,\pi(s))[r + \gamma V(s')]$ for $a \in A(s)$

1: $\pi(a|s) = 1/|A(s)|$

Exercise 4.7



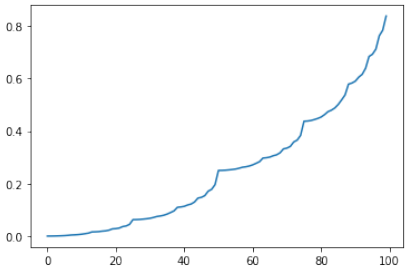
(a) Optimal Policy (b) Value Function

Figure 4.1: visualized by seaborn

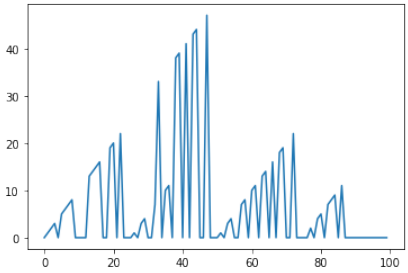
Exercise 4.8

$q(51,49) = 0.4 * 1 + 0.6 * v(2)$
$q(51,1) = 0.4 * v(52) + 0.6 * 0.4$ $q(51,49) - q(51,1) = 0.6v(2) - 0.4(v(52) - v(50)) > 0$

Exercise 4.9

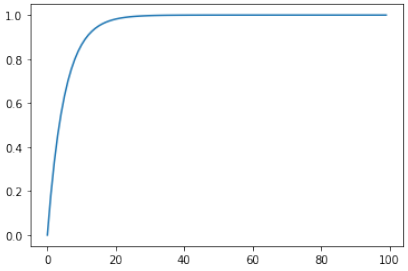


(a) Value function

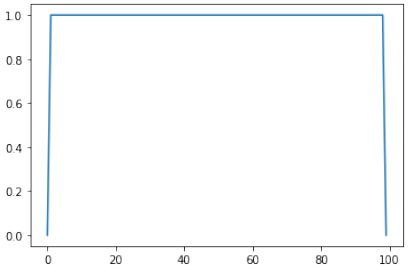


(b) Optimal Policy

Figure 4.2: Policy of p=0.25



(a) Value function



(b) Optimal Policy

Figure 4.3: Policy of p=0.55

Exercise 4.10

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma max_a q_k(s',a')]$$

Chapter 5

Monte Carlo Methods

Exercise 5.1

Last two rows : When we stick on 20, 21 then those two cases have very high probability to win. However from 19, another hit will likely to go bust.

Left row :If the dealer has an Ace, then it can be usable and would give more chances to hit.

Front most : If our ace is usable, then we can keep hit until the sum exceeds 21. And we can treat that ace to be 1 and hit again. This means that we have a second chance before going bust.

Exercise 5.2

It won't be very different, because even though same number appears several times in single episode, those cases can be treated as independently and act as distinct cases.

Exercise 5.3

I : Initial state
F : Final state
S : Intermediate states a : action

(I$\rightarrow$a) $\rightarrow$ (S$\rightarrow$a) $\rightarrow$ ... $\rightarrow$ (S$\rightarrow$a) $\rightarrow$ F

Exercise 5.4

Unless the pair S_t, A_t appears in $S_0, A_0, ...S_{t-1}, A_{t-1}$:
$Returns(S_t, A_t) += G$
$Counts(S_t, A_t) += 1$
$Q(s_t, A_t) \leftarrow Returns(S_t, A_t)/Counts(S_t, A_t)$
$\pi(S_t) \leftarrow argmax_a(S_t, a)$

Exercise 5.5

First visit : $10/1 = 10$
Second visit : $(1 + 2 + ... + 10)/10 = 5.5$

Exercise 5.6

$$Q(s,a) = \frac{\sum_{t \in J(s,a)} \rho_{t+1:T(t)-1} G_t}{\sum_{t \in J(s,a)} \rho_{t+1:T(t)-1}}$$

Exercise 5.7

The term with the biggest importance dominates the state's value. As its return(G) is based on the other fixed policy, its bias instantly increases the error.

Exercise 5.8

An episode with single states visited several times can be split into many episodes with first visit cases. In this perspective, translation to first visit simply increases the number of samples and its variance remains to be infinite.

Exercise 5.9

Unless S_t appears in $S_0, S_1, ..., S_{t-1}$:
$V(S_t) = V(S_t) + \frac{1}{C(S_t)}(G - V(S_t))$
$C(S_t) = C(S_t) + 1$

Exercise 5.10

$$
\begin{aligned}
V_{n+1} &= \frac{\sum_k^n W_k G_k}{\sum_k^n W_k} = \frac{\sum_k^{n-1} W_k G_k + W_n G_n}{\sum_k^n W_k} \\
&= \frac{(\sum_k^{n-1} W_k) V_n + W_n G_n}{\sum_k^n W_k} \\
&= \frac{(\sum_k^n W_k) V_n - W_n V_n + W_n G_n}{\sum_k^n W_k} \\
&= V_n - \frac{W_n}{C_n}(G_n - V_n)
\end{aligned}
$$

Exercise 5.11

If $A_t = \pi(s_t)$ then $\pi(A_t|S_t) = 1$ else : $\pi(A_t|S_t) = 0$ and breaking loop is more efficient.

Exercise 5.12

Programming Task

Exercise 5.13

$$
\begin{aligned}
& E[_{t:T-1} R_{t+1}] \\
=& E\left[\frac{\pi(A_t|S_t)}{b(A_t|S_t)} \ldots \frac{\pi(A_{T-1}|S_{T-1})}{b(A_{T-1}|S_{T-1})} R_{t+1}\right] = E[A_{T-1}] \\
=& E[A_{T-2}] * \sum_a b(a|S_{T-1}) \frac{\pi(a|S_{T-1})}{b(a|S_{T-1})} = E[A_{T-2}] \qquad by(5.13) \\
& \therefore E[S_{T-1}] = E[S_t] = E\left[\frac{\pi(a|S_t)}{b(a|S_t)} R_{t+1}\right]
\end{aligned}
$$

Exercise 5.14

Skipped

Chapter 6

Temporal-Difference Learning

Exercise 6.1

$$\begin{aligned} G_t - V_t(S_t) &= \delta_t + \gamma(G_{t+1} - V_t(S_{t+1})) \\ &= \delta_t + \gamma(V_{t+1}(S_{t+1}) - V_t(S_{t+1})) + \gamma(G_{t+1} - V_{t+1}(S_{t+1})) \\ &= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k + \sum_{k=t}^{T-1} \gamma^{k-t+1}(V_{k+1}(S_{k+1}) - V_k(S_{k+1}))) \end{aligned}$$

Exercise 6.2

Monte Carlo method averages every case which each states are visited in episodes. So newly generated episode is not likely to impact the change of state value. However, TD instantly updates each state value proportional to alpha, so the convergence to optimal is faster.

Exercise 6.3

$$V(A) = V(A) + 0.1(R + V(T) - V(A)) \quad = 0.5 + 0.1(0 + 0 - 0.5) = 0.45$$

Exercise 6.4

Even though wide range of alpha is used, bigger alpha leads to faster convergence and to less optimal converged value, and for small alpha, exactly opposite attribute appears.

Exercise 6.5

If alpha is big, then updated amount per step is too big to safely converge to optimal value, and keeps bouncing.

Exercise 6.6

Applying DP or solving equation would be possible solution. In this case, the number of state is small, so solving equation would be possible and could lead to exact value.

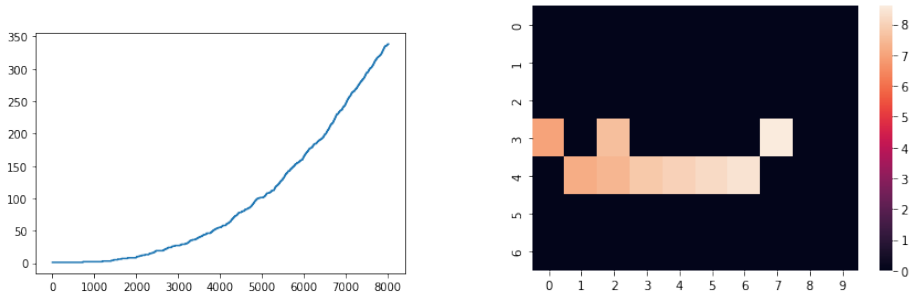
Exercise 6.7

$$V(S_t) = V(S_t) + \alpha \left[\rho_{t:t} R_{t+1} + (S_{t+1}) - V(s_t)\right]$$

Exercise 6.8

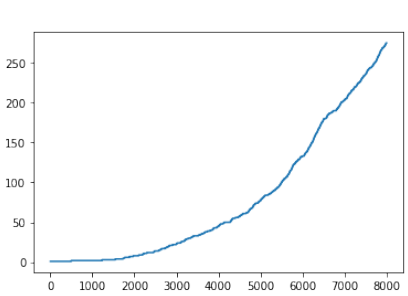
$$\begin{aligned}
\delta_t &= R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \\
G_t - Q(S_t, A_t) &= R_{t+1} + \gamma G_{t+1} - Q(S_t, A_t) + \gamma Q(S_{t+1}, A_{t+1}) - \gamma Q(S_{t+1}, A_{t+1}) \\
&= \delta_t + \gamma(G_{t+1} + Q(S_{t+1}, A_{t+1})) \\
&= \delta_t + \gamma\delta_{t+1} + ... + \gamma^{T-t}(G_T - Q(T, A_T)) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\gamma_k
\end{aligned}$$

Exercise 6.9

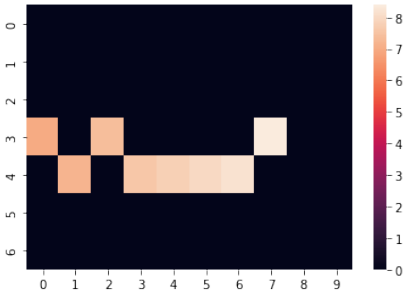


(a) Episodes/Timestamps (b) Trace of movement

Figure 6.1: Only King's movement



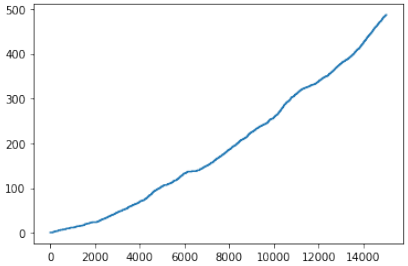
(a) Episodes/Timestamps



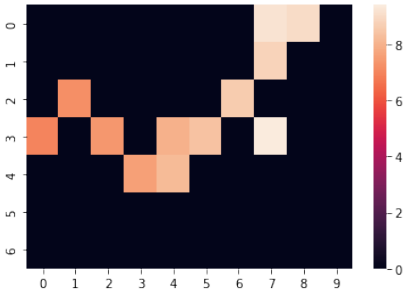
(b) Trace of movement

Figure 6.2: King's movement + Not moving

Exercise 6.10



(a) Episodes/Timestamps



(b) Trace of movement

Figure 6.3: Stochastic Grid World

Exercise 6.11

Q-learning is an off-policy method because it doesn't follow e-greedy method in update term $max_a Q(S', a)$ which is actually greedy.

Exercise 6.12

No. If the state returns to S_0 after initial action A_0, then the next action A_1 of both methods will be chosen based on different Q values. Sarsa would select its next action based on Q function before update and Q-learning would select its next action based on updated Q function.

Exercise 6.13

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_2(S_{t+1}, a) - Q_1(S_t, A_t) \right]$$

From the equation above, $\pi(a|S_{t+1})$ can be represented explicitly as below if an e-greedy policy is used.

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) +$$
$$\alpha \left[R_{t+1} + \gamma \left(\frac{\epsilon}{|A(a)|} \sum_a Q_2(S_{t+1}, a) + (1 - \epsilon) max_a Q_2(S_{t+1}, a) \right) - Q_1(S_t, A_t) \right]$$

Exercise 6.14

We can create an afterstate of cars right after moving cars, because the action of moving cars instantly determine the afterstate. In this way, we can effectively reduce the number of states and action pairs into single afterstate which will increase the speed of convergence, considering there would be less state values to update.

Chapter 7

n-step Bootstrapping

Exercise 7.1

$$\begin{aligned} G_{t:t+n} - V(S_t) &= R_{t+1} + \gamma R_{t+1} + ... + \gamma^n V(S_{t+n}) - V(S_t) \\ &= \delta_t + \gamma(R_{t+1} - \gamma V(S_{t+1})) + ... \\ &= \delta_t + \gamma\delta_{t+1} + ... + \gamma^{n-1}\delta_{t+n-1} + \gamma^n(V(S_{t+n}) - V(S_{t+n})) \\ &= \sum_{k=0}^{n-1} \gamma^k \delta_{t+k} \end{aligned}$$

Exercise 7.3

1. Larger random walks would show the differences between the parameters significantly.
2. Yes. Case with smaller walks would have smaller optimal number of n.
3. No. The optimal state value and the slope of the graph would change, but the best parameters for the algorithm would be same.

Exercise 7.4

$$\begin{aligned} G_{t:t+n} &= Q_{t-1}(S_t, A_t) + \sum_{k=t}^{min(t+n,T)-1} \gamma^{k-t} \left[R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1} - Q_{k-1}(S_k, A_k)\right] \\ &= R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) + \sum_{k=t+1}^{min(t+n,T)-1} \gamma^{k-t} \left[R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1} - Q_{k-1}(S_k, A_k)\right] \\ &= R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) + ... + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}) \end{aligned}$$

Exercise 7.5

pass

Exercise 7.6

The expected value of $\rho_{t+1} = 1$ independently to estimates (sec 5.9)

Exercise 7.7

pass

Exercise 7.8

$$G_{h-1:h}\rho_{h-1}(R_h + \gamma V(S_h)) + (1 - \rho_{h-1})V(S_{h-1}) = \rho_{h-1}\delta_{h-1} + V(S_{h-1})$$

...

$$G_{t:h} = \rho_t\delta_t + ... + \gamma^{h-1}\rho_t...\rho_{h-1}\delta_{h-1} + V(S_t)$$

$$G_{t:h} - V(S_t) = \sum_{k=t}^{h-1} \delta_k \gamma^{k-t} \Pi_{m=t}^{k} \rho_m$$

Exercise 7.9

$$G_{h-1:h} = R_h + \gamma\rho_h(Q(S_h, A_h) - Q(S_h, A_h)) + \gamma V(S_h) = \delta_{h-1} + V(S_{h-1})$$

...

$$G_{t:h} = \delta_t + \gamma\rho_{t+1}\delta_{t+1} + ... + \gamma^{h-t-1}\rho_{t+1}...\rho h - 1\delta_{h-1} + V(S_t)$$

$$G_{t:h} - V(S_t) = \delta_t + \sum_{k=t+1}^{h-1} \gamma^{k-t}\delta_k \Pi_{m=t+1}^{k} \rho_m$$

$$R + t + 1 + \gamma\bar{V}(S_{t+1}) - Q(S_t, A_t)$$

$$= \delta_t + \hat{V}(S_t) - Q(S_t, A_t) = \delta_t(\because \hat{V}(S_t) - Q(S_t, A_t) \approx 0)$$

Exercise 7.10

pass

Exercise 7.11

$G_{t+n-1:t+n} = R_{t+n} + \gamma V_{t+n-1}(S_{t+n}) - Q(S_{t+n-1}, A_{t+n-1}) + Q(S_{t+n-1}, A_{t+n-1})$
$= \delta_{t+n-1} + Q(S_{t+n-1}, A_{t+n-1})$
$G_{t+n-2:t+n} = \delta_{t+n-2} + \gamma\pi(S_{t+n-1}|A_{t+n-1})\delta_{t+n-1} + Q(S_{t+n-2}, A_{t+n-2})$
$G_{t:t+n} = Q(S_t, A_t) + \sum_{k=t}^{min(t+n-1,T-1)} \delta_k \prod_{i=k+1}^{k} \gamma\pi(A_i|S_i)$

Chapter 8

n-step Bootstrapping

Exercise 8.1

If the model seems to be deterministic, then n-DynaQ replays single episode n times, so its more efficient.

Exercise 8.2

The additional reward $k\sqrt{\tau}$ makes the accumulated reward bigger than DynaQ.

Exercise 8.3

As the time step increases, the disadvantage of exploration gets bigger, and reduces the difference between both methods.

Exercise 8.4

pass

Exercise 8.5

Stochastic → Store records pick one in random.

Changing environment problem → Environment's probability of transition changes, so it takes a long time before the averaged number converges to correct number.

Modification → Consider storing only recent K(fixed constant) cases of every

(S_t, A_t) pair which would efficiently reduces both the time needed for convergence and space required for storing the numbers

Exercise 8.6

Even though the probability is skewed, the samples are also likely to follow the skewed distribution, so the simulated episodes will effectively represent the episodes played on the original environment.

Exercise 8.7

In the uniform case, every states are updated in a fixed and independent sequence without regarding to the episode's path. So, if the update sequence resembles some particular episode's path, then the convergence speed should be instantly high, but if not, would be significantly slow. This difference of convergence speed regarding to the episode makes the irregular scalloped shape on the graph.