Please refer to the "read_me.txt" for running the code.

Answer for q0:

(a)(,b) I agree to share code. (I would argue that the shared code should be given bonus)

Answer for q1:

1、(a) $V_*(s) = \max_a q_*(s,a)$

(b) $q_*(s,a) = \sum_{s',r} p(s',r \mid s,a)\,[r + \gamma V_*(s')]$

(c) $\pi_*(s) = \arg\max_a q_*(s,a)$

(d) $\pi_*(s) = \arg\max_a \sum_{s',r} p(s',r \mid s,a)\,[r + \gamma V_*(s')]$

(e) $V_\pi(s) = \sum_a \pi(a|s)\left(r(s,a) + \gamma \cdot \sum_{s'} p(s' \mid s,a) \cdot V_\pi(s')\right)$

$q_\pi(s,a) = r(s,a) + \sum_{s'} P(s' \mid s,a) \cdot \gamma \cdot \sum_{a'} \pi(a'|s') \cdot q_\pi(s',a')$

$V_*(s) = \sum_a \pi_*(a|s)\left(r(s,a) + \sum_{s'} p(s'|s,a) \cdot \gamma \cdot V_*(s')\right)$

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

$V_*(s)$ and $q_*(s,a)$ could be also written as follows.

$V_*(s) = \max_a \left(r(s,a) + \sum_{s'} p(s'|s,a) \cdot \gamma \cdot V_*(s')\right)$

$q_*(s,a) = r(s,a) + \sum_{s'} p(s' \mid s,a) \cdot \gamma \cdot \max_{a'} q_*(s',a')$

Answer for q2:

I prefer the method1 in 2(a) due to that np.argmax() would return the first best value's index and I use this method in q5(c) to output the optimal policy.

2. (a) Method 1: give all the ~~distribution~~ probability to the first best action, in this case, the agent would always select the first best action in each state.

Method 2: Due to that the best actions have the same values, we could use best action's values to check ~~wether~~ whether the two policy are the same.

Pseudocode for Method 1:

Keep actions in $A(s)$ having fixed sequence.

$\pi(s) = \text{first\_argmax}_a \sum_{s',r} p(s',r|s,a)[r+\gamma V(s')]$

Pseudocode for Method 2:

For each $s \in S$:

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

$V_1 = \sum p(s',r|s, \text{old-action})[r+\gamma V(s')]$

$\pi(s) \leftarrow \arg\max_a \sum p(s',r|s,a)[r+\gamma V(s')]$

$V_2 = \sum p(s',r|s,\pi(s))[r+\gamma V(s')]$

if $V_1 \neq V_2$, policy-stable $\leftarrow$ false

(b) Value functions desn't suffer from the issue: The best actions have the same values

Answer for q3:

3.a

1. Initialization

$Q(s,a) \in \mathbb{R}$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

2. Policy Evaluation

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in S$:

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

$q \leftarrow Q(s,a)$

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

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

until $\Delta < \theta$

3. Policy Improvement

policy-stable $\leftarrow$ true

For each $s \in S$:

old_action $= \pi(s)$

$\underbrace{\pi(s) = \arg\max_a Q(s,a)}$

If old_action $\neq \pi(s)$, then policy-stable $\leftarrow$ false

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

b.

$$q_{k+1}(s,a) \doteq \sum_{s',r} p(s',r|s,a)\left[r + \gamma \cdot \max_{a'} q(s',a)\right]$$

Answer for q4:

4. (a) state x prefer to take action c

state y may prefer to take action b

(b) Initial value $V(x)=0$, $V(y)=0$, $V(z)=0$
Initial policy $\pi(x)=c$, $\pi(y)=c$

policy evaluation: $\pi(x)=c$, $\pi(y)=c$

$$\begin{cases} V(x) = 0.1\,(-1+V(z)) + 0.9\,(-1+V(x)) \\ V(y) = 0.1\,(-2+V(z)) + 0.9\,(-2+V(y)) \end{cases}$$

$$\begin{cases} V(x) = -10 \\ V(y) = -20 \end{cases}$$

policy improvement:

$q(x,c) = -10$
$q(x,b) = 0.8\,(-1+V(y)) + 0.2\,(-1+V(x))$
$= -19$
$\pi(x) = \arg\max q(x,a) = c$
$q(y,c) = -20$
$q(y,b) = 0.8\,(-2+V(x)) + 0.2\,(-2+V(y))$
$= -14$
$\pi(y) = \arg\max q(y,a) = b$

Policy evaluation: $\pi(x) = c$, $\pi(y) = b$

$$\begin{cases} V(x) = 0.1(-1+V(z)) + 0.9(-1+V(x)) \\ V(y) = 0.2(-2+V(y)) + 0.8(-2+V(x)) \end{cases}$$

$$\begin{cases} V(x) = -10 \\ V(y) = -12.5 \end{cases}$$

policy improvement:

$$q(x,b) = 0.2(-1+V(x)) + 0.8(-1+V(y))$$
$$= -13$$
$$q(x,c) = -10$$

$$\pi(x) = \arg\max q(x,a) = c$$

$$q(y,b) = -12.5$$

$$q(y,c) = 0.1(-2+V(z)) + 0.9(-2+V(y))$$

$$= -13.25$$
$$\pi(y) = \arg\max q(y,a) = b$$

Policy optimal!

The optimal action: $\pi(x) = c$; $\pi(y) = b$
The optimal value: $V(x) = -10$; $V(y) = -12.5$

4 (c). Initial value $V(X)=0$, $V(y)=0$, $V(Z)=0$

Initial policy $\pi(X)=b$, $\pi(y)=b$.

Policy evaluation: (Initial policy)

$$\begin{cases} V(X) = 0.2(-1+V(X)) + 0.8(-1+V(y)) & ① \\ V(y) = 0.2(-2+V(y)) + 0.8(-2+V(X)) & ② \end{cases}$$

$$\begin{cases} 0.8\,V(X) = -1 + 0.8\,V(y) \\ 0.8\,V(y) = -2 + 0.8\,V(X) \end{cases} \rightarrow \underbrace{0.8\,V(X) = -1 - 2 + 0.8\,V(X)}\ ③$$

there is no solution for ③, and thus it couldn't converge when doing evaluation step by step.

Using discount: $0<\gamma<1$

policy evaluation: (Initial policy)

$$\begin{cases} V(X) = 0.2(-1+\gamma\cdot V(X)) + 0.8(-1+\gamma\cdot V(y)) \\ V(y) = 0.2(-2+\gamma\cdot V(y)) + 0.8(-1+\gamma\cdot V(X)) \end{cases}$$

$$\begin{cases} V(X) = \dfrac{-1-1.4\gamma}{(1-0.2\gamma)^2-(0.8\gamma)^2} \\ V(y) = \dfrac{-2-0.4\gamma}{(1-0.2\gamma)^2-(0.8\gamma)^2} \end{cases}$$

$\because\ 0<\gamma<1 \rightarrow \begin{cases} (1-0.2\gamma)^2-(0.8\gamma)^2 > 0 \\ -1-1.4\gamma > -2-0.4\gamma \end{cases} \rightarrow V(X) > V(y)$

$\therefore$ discount helps!

policy improvement:

$$Q(X,b) = \frac{-1-1.4\gamma}{(1-0.2\gamma)^2-(0.8\gamma)^2}$$

$$q(x,c) = 0.9(-1+\gamma\cdot V(x)) + 0.1(-1+\gamma\cdot V(z))$$

$$= -1 + 0.9\gamma\cdot\frac{-1-1.4\gamma}{(1-0.2\gamma)^2-(0.8\gamma)^2}$$

$$q(y,b) = \frac{-2-0.4\gamma}{(1-0.2\gamma)^2-(0.8\gamma)^2}$$

$$q(y,c) = 0.9(-2+\gamma\cdot V(y)) + 0.1(-2+\gamma\cdot V(z))$$

$$= -2 + 0.9\gamma\cdot\frac{-2-0.4\gamma}{(1-0.2\gamma)^2-(0.8\gamma)^2}$$

For $\pi(x) = \arg\max_a q(x,a)$

$\pi(y) = \arg\max_a q(y,a)$

and each q value involve γ. So the optimal value function depends on discount factor.

For example: If $\gamma = 0$, $q_*(y,b) = q_*(y,c) = -2 \rightarrow \pi_*(y) = b$ or c

If $\gamma = 1$ without the action b in initial policy for both state, $\pi_*(y) = b$.

due to the fact the optimal policy doesn't depend on initial policy.

We could conclude that optimal policy depend on γ.

Answer for q5:

a).

run python value_estimation.py to get the value function for the equiprobable random policy in the figures below

3.314	8.793	4.431	5.326	1.496
1.526	2.996	2.253	1.911	0.55
0.055	0.742	0.676	0.361	-0.4
-0.97	-0.432	-0.352	-0.583	-1.18
-1.854	-1.342	-1.226	-1.42	-1.972

b) run value_iteration.py to get the optimal value function and the optimal policy in the figures below

21.976	24.419	21.977	19.419	17.477
19.779	21.977	19.779	17.801	16.021
17.801	19.779	17.801	16.021	14.419
16.021	17.801	16.021	14.419	12.977
14.419	16.021	14.419	12.977	11.679

right,	left,right,up,down,	left,	left,right,up,down,	left,
right,up,	up,	left,up,	left,	left,
right,up,	up,	left,up,	left,up,	left,up,
right,up,	up,	left,up,	left,up,	left,up,
right,up,	up,	left,up,	left,up,	left,up,

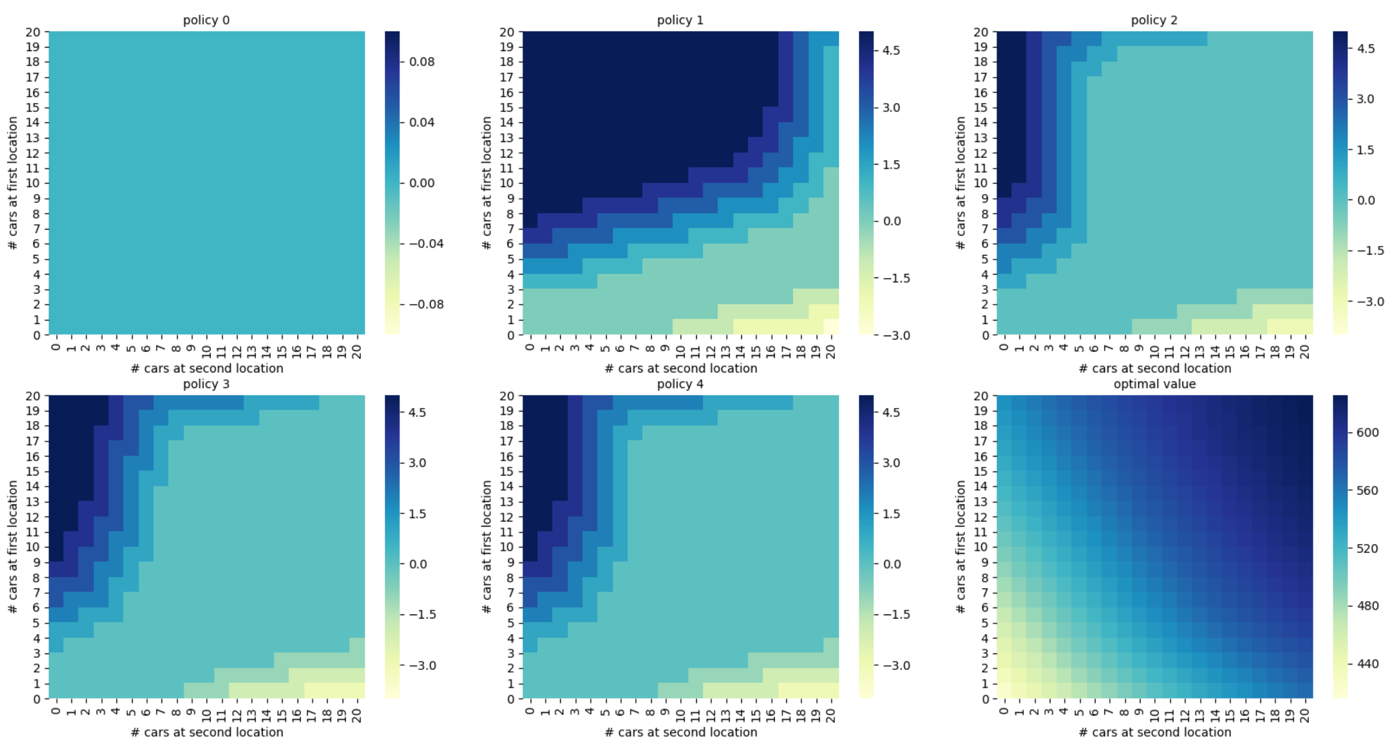
c) run policy_iteration.py to get the optimal value function and the optimal policy in the figures below

21.977	24.419	21.977	19.419	17.477
19.78	21.977	19.78	17.802	16.022
17.802	19.78	17.802	16.022	14.419
16.022	17.802	16.022	14.419	12.977
14.419	16.022	14.419	12.977	11.68

right	right	left	right	left
right	up	left	left	left
right	up	left	left	left
right	up	left	left	left
right	up	left	left	left

Answer for q6:

a) Run python jack_car_a.py --constant_return to get the optimal value function and the optimal policy in the figures below



Some details:

Constant return is used to simplify the model: (You could also run python jack_car_a.py to get the accurate model)

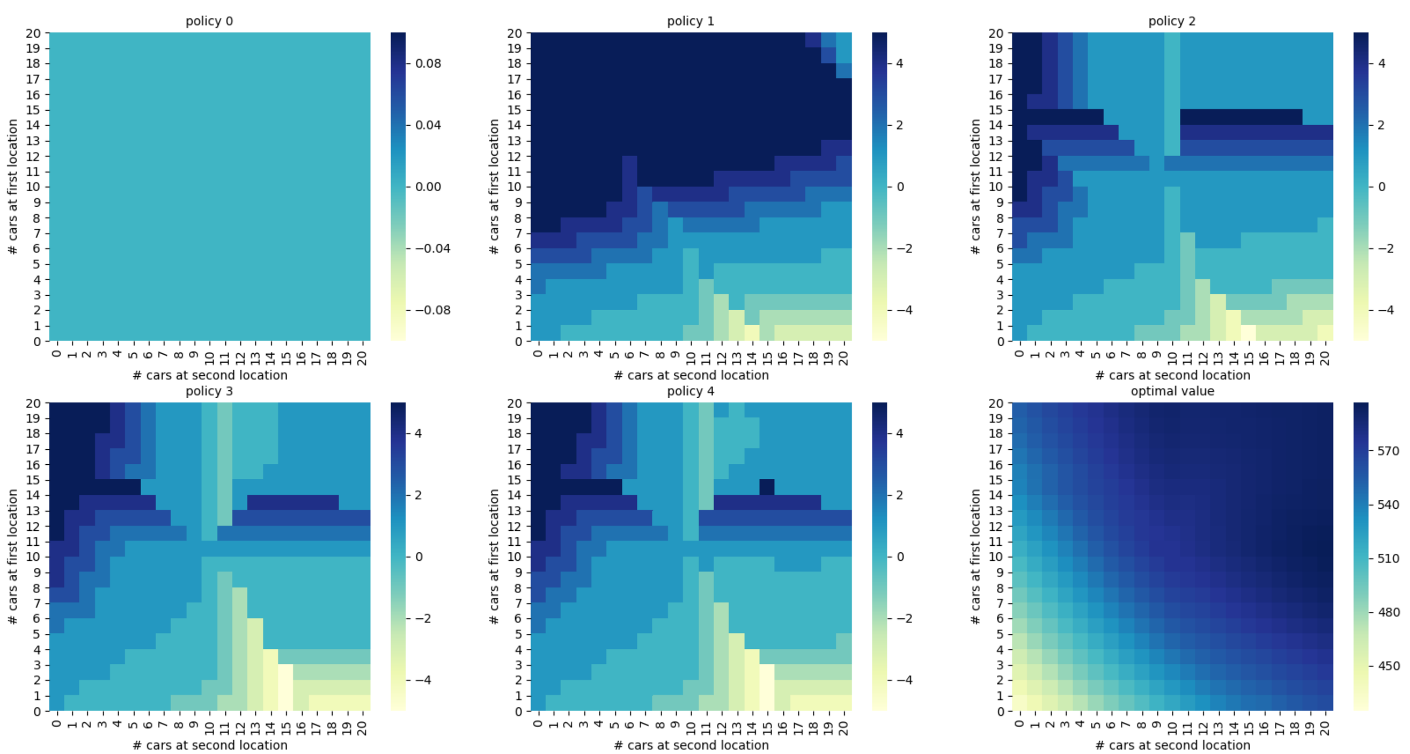
In order to check the output quickly, constant returns instead of the random returns from poisson distribution would be more efficient. Specifically, the expectation of poisson distribution is lambda. In this case, the expectation of returns in the first location is 3 and the expectation returns in the second location is 2. It would leave the optimal policy and value almost the same.

b):

dynamic function changes:

- the employee's help could save 2 cost if the action is moving one or more cars from the first location to the second location and thus increase reward compared with the previous dynamic function
- the parking would cost 4 for each location if the parking number bigger than 10 and thus would decrease the reward compared with the previous dynamic function

run jack_rental_car_b.py --constant_return to get the optimal value function and the optimal policy in the figures



Details:

You could also run python jack_car_b.py to use the accurate the model:

In order to check the output quickly, constant returns instead of the random returns from poisson distribution would be more efficient. Specifically, the expectation of poisson distribution is lambda. In this case, the expectation of returns in the first location is 3 and the expectation returns in the second location is 2. It would leave the optimal policy and value almost the same.

Policy difference:

- States that have 10 more cars in location1 would like to move more cars to location 2 to avoid parking fee and use the employee's help

- States that have 10 more cars in one location and less than 10 cars in the another location would like to move cars (won't excess 2) to the location that has less cars to avoid parking fee.

Answer for q7:

7. (a). Prove $\left|\max_a f(a) - \max_a g(a)\right| \le \max_a |f(a) - g(a)|$

given any function f, g:

$$|f(a) - g(a)| \ge f(a) - g(a)$$
$$|f(a) - g(a)| + g(a) \ge f(a) \quad ①$$

$$\max\left(|f(a) - g(a)| + g(a)\right) \ge \max f(a) \quad ②$$

$\because$ $\max|f(a) - g(a)| + \max g(a) \ge \max\left(|f(a) - g(a)| + g(a)\right)$

$\therefore$ $\max|f(a) - g(a)| + \max g(a) \ge \max f(a) \quad ③$

$$\max|f(a) - g(a)| \ge \max f(a) - \max g(a) \quad ④$$

It's easy to use the symmetry trick to replace $f(a)$, $g(a)$:

$$\max|f(a) - g(a)| \ge \max g(a) - \max f(a) \quad ⑤$$

From ④, ⑤,

$$\max|f(a) - g(a)| \ge |\max f(a) - \max g(a)|$$

7. (b) $V_i = \begin{bmatrix} V_i(s_1) \\ V_i(s_2) \\ \vdots \\ V_i(s_n) \end{bmatrix}$ $\quad V_i' = \begin{bmatrix} V_i'(s_1) \\ V_i'(s_2) \\ \vdots \\ V_i'(s_n) \end{bmatrix}$

$$\|BV_i - BV_i'\| = \left\{ \begin{matrix} \max_a \sum_{s_1', r} p(s_1', r \mid s_1, a)(r + \gamma V_i(s_1')) - \max_a \sum_{s_1', r} p(\ \mid\ )(r + \gamma V_i'(s_1')) \\ \max_a \sum_{s_2', r} p(s_2', r \mid s_2, a)(r + \gamma V_i(s_2')) - \max_a \sum_{s_2', r} p(\ \mid\ )(r + \gamma V_i'(s_2')) \\ \vdots \qquad\qquad\qquad \vdots \\ \max_a \sum_{s_n', r} p(s_n', r \mid s_n, a)(r + \gamma V_i(s_n')) - \max_a \sum_{s_n', r} (\ \mid\ )(r + \gamma V_i'(s_n')) \end{matrix} \right\}$$

$$\le \left\{ \begin{matrix} \max_a \left| \sum_{s_1', r} p(s_1', r \mid s_1, a) \gamma (V_i(s_1') - V_i'(s_1')) \right| \\ \max_a \left| \sum_{s_2', r} p(s_2', r \mid s_2, a) \cdot \gamma (V_i(s_2') - V_i'(s_2')) \right| \\ \vdots \\ \max_a \left| \sum_{s_n', r} p(s_n', r \mid s_n, a) \cdot \gamma (V_i(s_n') - V_i'(s_n')) \right| \end{matrix} \right\} \quad ②$$

From ②, we could get

$$\|BV_i - BV_i'\|_\infty \le \underbrace{\left\| \begin{matrix} \max_a \left| \sum_{s_1', r} p(s_1', r \mid s_1, a) \gamma (V_i(s_1') - V_i'(s_1')) \right| \\ \max_a \left| \sum_{s_2', r} p(s_2', r \mid s_2, a) \cdot \gamma (V_i(s_2') - V_i'(s_2')) \right| \\ \vdots \\ \max_a \left| \sum_{s_n', r} p(s_n', r \mid s_n, a) \cdot \gamma \cdot (V_i(s_n') - V_i'(s_n')) \right| \end{matrix} \right\|_\infty}_{③}$$

$\because \sum p(s, r \mid s, a) = 1$

$\therefore ③ \le \gamma \| V_i - V_i' \|_\infty$

$\therefore \| BV_i - BV_i' \|_\infty \le \gamma \| V_i - V_i' \|_\infty$

7 (c) Method 1:

Assume that $\{V_0, V_1, V_2, V_3 \cdots V_n\}$ is sequence of values.

$V_n = BV_{n-1}$, and $\Delta = \|V_0 - V_1\|_\infty$:

from 7(b), Bellman backup operator is a ~~backup~~ contraction mapping.

$\|BV_0 - BV_1\|_\infty \leq \gamma \|V_0 - V_1\|_\infty = \gamma \cdot \Delta$

$\therefore \|BV_n - BV_{n+1}\|_\infty \leq \gamma^{n+1} \|V_0 - V_1\|_\infty = \gamma^{n+1} \cdot \Delta$

$\lim_{n \to \infty} \|BV_n - BV_{n+1}\|_\infty \leq \lim_{n \to \infty} \gamma^{n+1} \cdot \Delta$, where $0 < \gamma < 1$, Δ is a constant.

$\therefore \lim_{n \to \infty} \|BV_n - BV_{n+1}\|_\infty = 0$ ①

$\because V_n = BV_{n-1} \rightarrow BV_n = V_{n+1}$ ②

$\rightarrow \lim_{n \to \infty} BV_n = V_n$

Method 2: use cauche sequence, metric space to prove the fixed points

(c) A is the sequence contains $x_0, x_1 \cdots x_n$, where $x_n = B(x_{n-1})$

d is a metric function: $d(x_n, x_m) = \|x_n - x_m\|_\infty$

$d(x,z) \le d(x,y) + d(y,z)$ triangle inequality

In this case (A, d) is a metric space ①

From b, we could get

$|Bx_1 - Bx_0|_\infty \le \gamma |x_1 - x_0|_\infty \rightarrow d(x_2, x_1) \le \gamma \cdot d(x_1, x_0)$ ②

It's easy to find $d(x_{n+1}, x_n) \le q^n d(x_1, x_0)$, where $q = \gamma$

For Bellman operator is a contraction mapping, we can show $\{x_n\}$ is a ~~the~~ Cauchy sequence.

let $m, n \in N$ and $m > n$,

$$d(x_m, x_n) \le d(x_m, x_{m-1}) + d(x_{m-1}, x_{m-2}) + \cdots + d(x_{n+1}, x_n) \quad \begin{pmatrix}\text{triangle}\\ \text{inequality}\end{pmatrix}$$

$$\le q^{m-1} d(x_1, x_0) + q^{m-2} d(x_1, x_0) + \cdots + q^n d(x_1, x_0)$$

$$= q^n d(x_1, x_0) \sum_{k=0}^{m-n-1} q^k$$

$$\le q^n d(x_1, x_0) \sum_{k=0}^{\infty} q^k = \underbrace{q^n d(x_1, x_0)\left(\frac{1}{1-q}\right)}_{\triangle} \quad ③$$

let $\varepsilon > 0$ be arbitrary, since $q = \gamma \in [0, 1)$, there would be a large $C \in N$,

$$q^C < \frac{\varepsilon(1-q)}{d(x_1, x_0)} \quad ④$$

If m, n larger than C $\longrightarrow q^n < q^C$ ⑤

$$\therefore d(x_m, x_n) \le q^n d(x_1, x_0)\left(\frac{1}{1-q}\right) < \frac{\varepsilon(1-q)}{d(x_1, x_0)} \cdot d(x_1, x_0)\left(\frac{1}{1-q}\right) = \varepsilon$$

So for every positive real number ε, there is a positive integer N

Such that for all natural numbers $m, n > N$

$d(x_m - x_n) < \varepsilon$, the sequence is cauche sequance.

In this case, the sequance $\{x_n\}$ has a limit $x^* \in X$

$$x^* = \lim_{n \to \infty} x_n = \lim_{n \to \infty} B(x_{n-1}) = B(\lim_{n \to \infty} x_{n-1}) = B(x^*)$$

, which shows value iteration ~~cove~~ converges to a fix point. (converge)

If there are two fixed points P_1 and P_2:

$$\| B(P_1) - B(P_2) \|_{L-\infty} \le \gamma \cdot \| P_1 - P_2 \|_\infty$$

$\because B(P_1) = P_1$, $B(P_2) = P_2$ (fixed points' property)

$\therefore \| P_1 - P_2 \|_{L-\infty} \le \gamma \cdot \| P_1 - P_2 \|_\infty$ ①

$\because 0 < \gamma < 1$

$\therefore$ equation ① would be correct if only $P_1 = P_2$

, which shows the fixed point is unique. (unique)

When $B(x^*) = x^*$, $x^*(s) = \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma x^*(s')]$

$$= \max_a E_{\pi_*} [R_{t+1} + \gamma X_*(S_{t+1}) \mid S_t = s, A_t = a]$$

$$= \max_a E_{\pi_*} [R_{t+1} + \gamma \cdot G_{t+1} \mid S_t = s, A_t = a]$$

$$= \max_a E_{\pi_*} [G_t \mid S_t = s, A_t = a]$$

$$X^*(s) = \max_a E_{\pi^*}[G_t \mid S_t = s, A_t = a]$$

$$= \underbrace{\max_a q_{\pi^*}(s,a)}$$

Which shows $X^*(s)$ is the max q_* value and thus ~~would~~ is optimal value. (optimal value function)

Reference:

https://github.com/ShangtongZhang/reinforcement-learning-an-introduction

https://en.wikipedia.org/wiki/Banach_fixed-point_theorem