CS5180 / Reinforcement Learning
Homework 3 – Solutions

Teoman Kaman

Question 1

(a) Give an equation for v^* in terms of q^*.

By definition, the optimal state-value is the best action-value:

$$v^*(s) = \max_a q^*(s, a).$$

(b) Give an equation for q^* in terms of v^* and p.

Starting from (s, a), the next state is distributed by $p(s' \mid s, a)$, and the expected one-step reward is $r(s, a)$. Therefore:

$$q^*(s, a) = r(s, a) + \gamma \sum_{s'} p(s' \mid s, a)\, v^*(s').$$

(c) Give an equation for π^* in terms of q^*.

An optimal policy chooses an action that maximises the optimal action-value:

$$\pi^*(s) = \arg\max_a q^*(s, a).$$

(If there are ties, any maximising action is optimal.)

(d) Give an equation for π^* in terms of v^* and p.

Using the Bellman optimality equation written with $p(s' \mid s, a)$ and $r(s, a)$, the optimal policy chooses an action that maximises the expected return:

$$\pi^*(s) = \arg\max_a \left[r(s, a) + \gamma \sum_{s'} p(s' \mid s, a)\, v^*(s') \right].$$

Question 2 – Policy Iteration by Hand (Undiscounted)

We have an undiscounted MDP ($\gamma = 1$) with states x, y, z, where z is terminal and $V(z) = 0$. Actions are b and c in states x and y.

Given dynamics

- In state x, action b: $x \to y$ with prob 0.7, $x \to x$ with prob 0.3.
- In state y, action b: $y \to x$ with prob 0.7, $y \to y$ with prob 0.3.
- In either x or y, action c: go to z with prob 0.2, stay in same state with prob 0.8.

Given rewards

- In state x, reward is always -1.
- In state y, reward is always -2.
- Terminal state z has $V(z) = 0$.

(a) Qualitative reasoning about the optimal policy

State y gives a worse (more negative) reward per step than x (-2 vs -1). So, intuitively:

- In state y, it is usually better to try to move to x using action b, because spending time in x is less costly than spending time in y.
- In state x, it is attractive to take action c, because with probability 0.2 it terminates at z, which stops future negative rewards.

So we expect something like $\pi^*(x) = c$ and $\pi^*(y) = b$.

(b) Policy iteration starting from $\pi_0(x) = c,\ \pi_0(y) = c$

Step 1: Policy evaluation of π_0

Bellman expectation equation (undiscounted, $\gamma = 1$):

$$V^\pi(s) = \sum_{s'} p(s' \mid s, \pi(s)) \big[r(s) + V^\pi(s') \big],$$

where $r(x) = -1$ and $r(y) = -2$.

State x under action c. From x, action c goes to z with prob 0.2 and stays at x with prob 0.8.

$$V^{\pi_0}(x) = -1 + 0.8\, V^{\pi_0}(x) + 0.2\, V^{\pi_0}(z).$$

Since $V^{\pi_0}(z) = 0$,

$$V^{\pi_0}(x) = -1 + 0.8 V^{\pi_0}(x).$$

Move terms:

$$V^{\pi_0}(x) - 0.8 V^{\pi_0}(x) = -1 \quad \Rightarrow \quad 0.2 V^{\pi_0}(x) = -1 \quad \Rightarrow \quad V^{\pi_0}(x) = -5.$$

State y under action c. Similarly,

$$V^{\pi_0}(y) = -2 + 0.8\,V^{\pi_0}(y) + 0.2\,V^{\pi_0}(z) = -2 + 0.8V^{\pi_0}(y).$$

Thus

$$0.2V^{\pi_0}(y) = -2 \quad \Rightarrow \quad V^{\pi_0}(y) = -10.$$

So the evaluated values are:

$$V^{\pi_0}(x) = -5, \qquad V^{\pi_0}(y) = -10, \qquad V^{\pi_0}(z) = 0.$$

Step 2: Policy improvement from π_0

We compute action-values using:

$$Q^{\pi_0}(s,a) = r(s) + \sum_{s'} p(s' \mid s,a) V^{\pi_0}(s').$$

Improve state x.

$$Q^{\pi_0}(x,c) = -1 + 0.8V^{\pi_0}(x) + 0.2V^{\pi_0}(z) = -1 + 0.8(-5) + 0 = -5.$$

$$Q^{\pi_0}(x,b) = -1 + 0.7V^{\pi_0}(y) + 0.3V^{\pi_0}(x) = -1 + 0.7(-10) + 0.3(-5) = -1 - 7 - 1.5 = -9.5.$$

Since $-5 > -9.5$, choose c in x.

Improve state y.

$$Q^{\pi_0}(y,c) = -2 + 0.8V^{\pi_0}(y) + 0.2V^{\pi_0}(z) = -2 + 0.8(-10) = -10.$$

$$Q^{\pi_0}(y,b) = -2 + 0.7V^{\pi_0}(x) + 0.3V^{\pi_0}(y) = -2 + 0.7(-5) + 0.3(-10) = -2 - 3.5 - 3 = -8.5.$$

Since $-8.5 > -10$, choose b in y.

So the improved policy is:

$$\pi_1(x) = c, \qquad \pi_1(y) = b.$$

Step 3: Policy evaluation of π_1

Now solve for $V^{\pi_1}(x)$ and $V^{\pi_1}(y)$.

State x under c. This is the same equation as before:

$$V^{\pi_1}(x) = -1 + 0.8V^{\pi_1}(x) + 0.2 \cdot 0 \quad \Rightarrow \quad V^{\pi_1}(x) = -5.$$

State y under b. From y, action b goes to x with prob 0.7 and stays at y with prob 0.3:

$$V^{\pi_1}(y) = -2 + 0.7V^{\pi_1}(x) + 0.3V^{\pi_1}(y).$$

Substitute $V^{\pi_1}(x) = -5$:

$$V^{\pi_1}(y) = -2 + 0.7(-5) + 0.3V^{\pi_1}(y) = -2 - 3.5 + 0.3V^{\pi_1}(y) = -5.5 + 0.3V^{\pi_1}(y).$$

Move terms:

$$V^{\pi_1}(y) - 0.3V^{\pi_1}(y) = -5.5 \quad \Rightarrow \quad 0.7V^{\pi_1}(y) = -5.5 \quad \Rightarrow \quad V^{\pi_1}(y) = -\frac{5.5}{0.7} = -\frac{55}{7} \approx -7.857.$$

So:

$$V^{\pi_1}(x) = -5, \qquad V^{\pi_1}(y) = -\frac{55}{7}.$$

Step 4: Policy improvement from π_1

Check if any action changes.

State x.

$$Q^{\pi_1}(x,c) = -1 + 0.8V^{\pi_1}(x) = -1 + 0.8(-5) = -5.$$

$$Q^{\pi_1}(x,b) = -1 + 0.7V^{\pi_1}(y) + 0.3V^{\pi_1}(x) = -1 + 0.7\left(-\frac{55}{7}\right) + 0.3(-5) = -1 - 5.5 - 1.5 = -8.$$

So c remains better in x.

State y.

$$Q^{\pi_1}(y,b) = V^{\pi_1}(y) = -\frac{55}{7} \approx -7.857.$$

$$Q^{\pi_1}(y,c) = -2 + 0.8V^{\pi_1}(y) = -2 + 0.8\left(-\frac{55}{7}\right) = -2 - \frac{44}{7} = -\frac{58}{7} \approx -8.286.$$

So b remains better in y.

No action changes, so the policy is stable and therefore optimal.

Final answer for (b)

$$\boxed{\pi^*(x) = c, \qquad \pi^*(y) = b}$$

and

$$\boxed{V^*(x) = -5, \qquad V^*(y) = -\frac{55}{7} \approx -7.857, \qquad V^*(z) = 0.}$$

(c) What if the initial policy is b in both states? Does discounting help?

If the initial policy is

$$\pi_0(x) = b, \qquad \pi_0(y) = b,$$

then the agent never chooses action c, so it can never reach the terminal state z. Because rewards are negative at every step and $\gamma = 1$, the return is an infinite sum of negative numbers. Therefore, the value function is not finite (it diverges to $-\infty$), and policy evaluation does not properly converge.

If we use discounting (e.g. $\gamma = 0.9$), then the infinite sum is discounted and becomes finite. So policy evaluation converges, and policy iteration works normally.

In this MDP, the optimal policy does not change when using a discount factor in $(0, 1]$: it remains $\pi^*(x) = c$ and $\pi^*(y) = b$.

Question 3

(a) Value Iteration

Value iteration is performed using the update:

$$V_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V_k(s')\right],$$

with discount factor $\gamma = 0.8$ and threshold $\theta = 10^{-3}$.

The algorithm converges when the maximum change in value over all states is less than θ.
Implementation: The full implementation and all outputs (final values and greedy policy) are provided in the submitted `.ipynb` file.

(b) Policy Iteration

Given a fixed policy, we compute its value function using iterative policy evaluation:

$$V_{k+1}(s) = \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V_k(s') \right].$$

Implementation: The policy evaluation procedure and the resulting value function are provided in the submitted `.ipynb` file.

Question 4

(a) Fixing the non-termination bug in policy iteration

The bug happens in the *policy improvement* step. If there are ties (two or more actions give the same maximal value), the policy can switch between equally good actions forever, so `policy-stable` may never become true.

A simple fix is: **do not change the action if the current action is still optimal**. Formally, for each state s, compute the set of greedy actions

$$A^*(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s') \big].$$

Then update the policy as follows:

- If the old action $\pi(s) \in A^*(s)$, keep it (no switch on ties).
- Otherwise, choose an action from $A^*(s)$ using a deterministic tie-break rule (e.g., smallest action index).

This guarantees convergence because the policy will stop changing when it is already greedy, even if multiple greedy actions exist.

(b) Is there an analogous bug in value iteration?

No, value iteration does not have the same non-termination issue. Value iteration terminates using a threshold on the value change (e.g., stop when $\Delta < \theta$), not by checking whether a greedy policy is stable.

Even if multiple actions are tied during the $\arg\max$, the *value function* still converges (for $\gamma < 1$) to v^* because the Bellman optimality operator is a contraction. At worst, the *extracted greedy policy* could differ due to ties, but this does not prevent the value estimates from converging.

(If we want a deterministic final policy from value iteration, we can apply the same tie-breaking rule when extracting $\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$.)

Question 5

(a) Original Reward Function

Policy iteration is applied using the original reward model described in the problem.

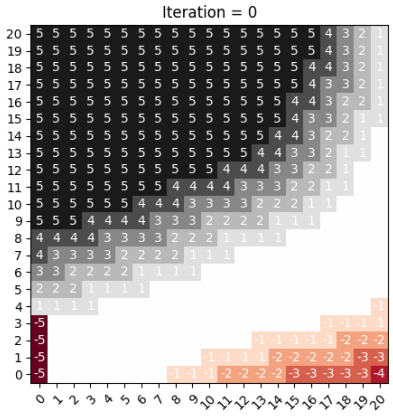

Figure 1: Iteration 0

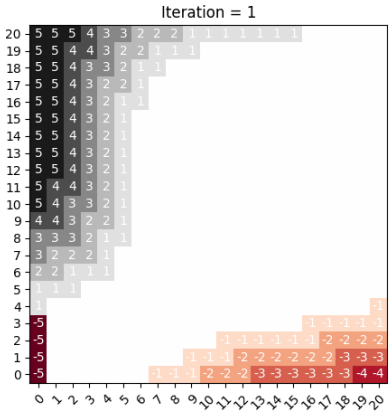

Figure 2: Iteration 1

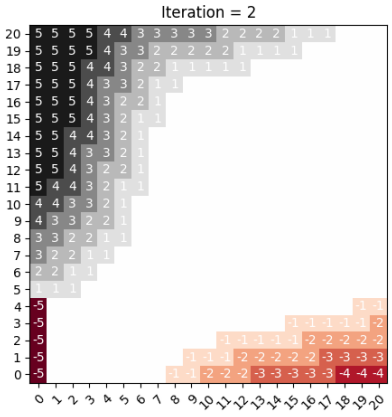

Figure 3: Iteration 2

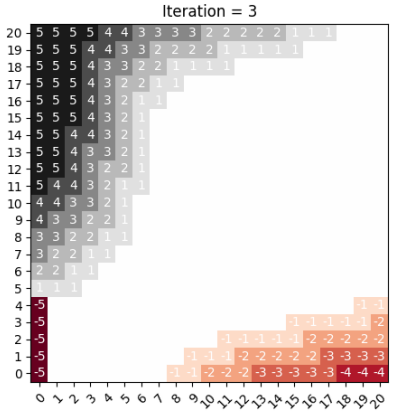

Figure 4: Iteration 3

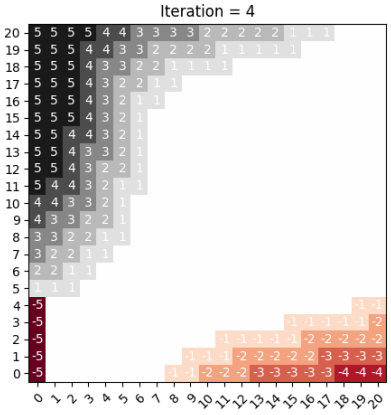

Figure 5: Iteration 4

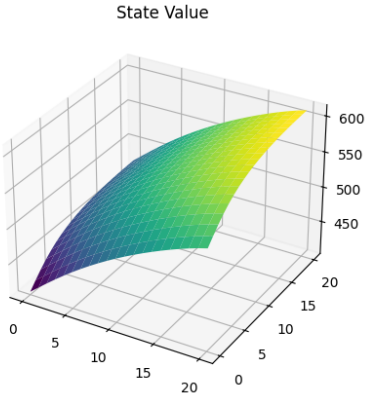

Figure 6: State value

(b) Modified Reward Function

The reward function is modified as follows:

- The first car moved from location 1 to 2 is free.
- All other car movements cost $2 per car.
- Parking more than 10 cars at any location costs $4.

Policy iteration is rerun using this modified reward function.

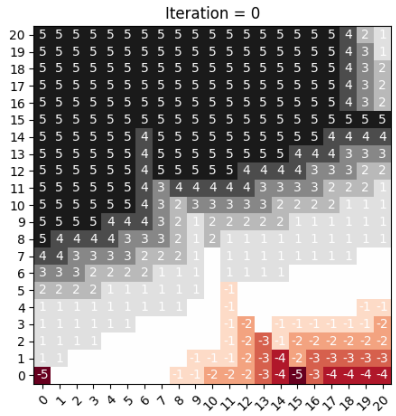

Figure 7: Iteration 0

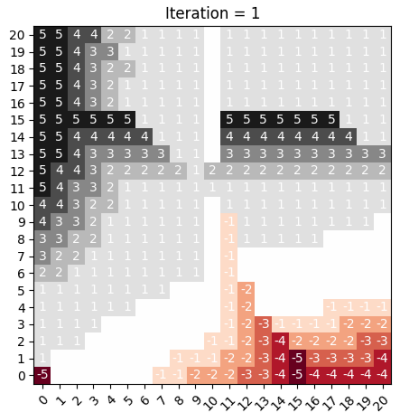

Figure 8: Iteration 1

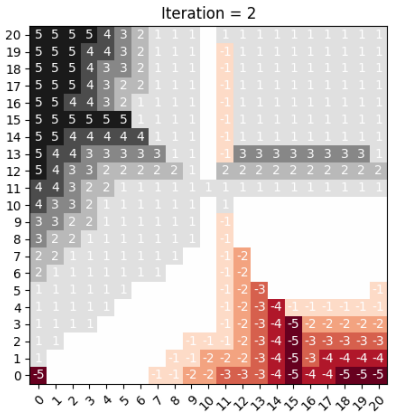

Figure 9: Iteration 2

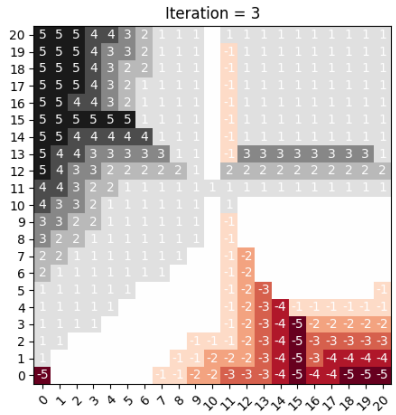

Figure 10: Iteration 3

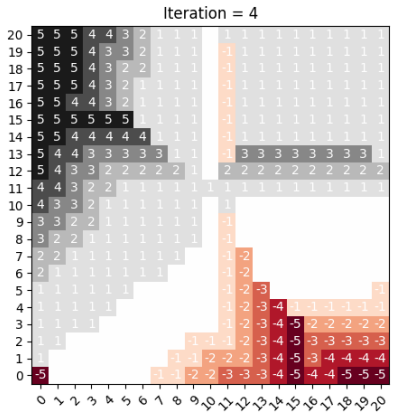

Figure 11: Iteration 4

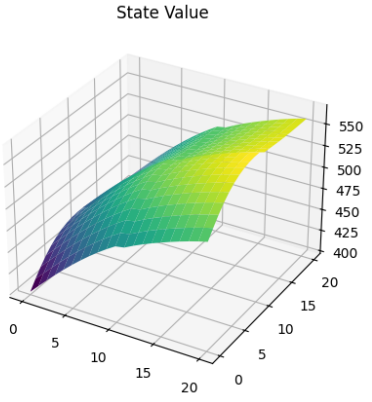

Figure 12: State value

Q5(b) Policy Comparison. The optimal policy in Q5(b) differs from Q5(a) in two important ways. First, it moves more cars from location 1 to location 2 because the first car in this direction can be moved for free, which reduces the transfer cost. Second, the policy tries to keep both locations at or below 10 cars in order to avoid the $4 parking fee.

Overall, the Q5(b) policy is more active than the Q5(a) policy. It takes advantage of the free move and avoids the parking penalty whenever the long-term value justifies the additional $2 moving cost for other cars.

Implementation: The full policy iteration implementation, including all figures and plots for both Q5(a) and Q5(b), is provided in the submitted `.ipynb` file.