SI252 Reinforcement Learning: Homework #05

Due on April 30, 2025 at 11:59 p.m.(CST)

Name: **Zhou Shouchen**
Student ID: 2021533042

Problem 1

Proofs
(a) Reproduce the proofs of Bellman Expectation Equations for Markov Decision Processes (MDPs).
(b) Reproduce the proofs of Bellman Optimality Equations for Markov Decision Processes (MDPs).
(c) Reproduce the proofs of Policy Improvement by acting greedily.
Solution
(a) The Bellman Expectation Equations for MDPs have 2 Expectation forms, and 4 summation forms.
G_t is the future accumulated reward: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$, $v_{\pi(s)}$ is the value function under the policy π, $q_\pi(s,a)$ is the value-state function under the policy π.
1.1 The state value function's expectation form: From the definition, we can get that

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots |S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1}|S_t = s] + \gamma \mathbb{E}_\pi[G_{t+1}|S_t = s] \end{aligned}$$

Then we need to focus on $\mathbb{E}[G_{t+1}|S_t = s]$. According to Adam's law with extra conditioning, we have

$$\mathbb{E}[\mathbb{E}(Y|X,Z)|Z] = \mathbb{E}[Y|Z]$$

Let $Y = G_{t+1}, X = S_{t+1}$ and $Z = S_t$, where S_{t+1}, S_t are random variables, thus we have:

$$\mathbb{E}_\pi[\mathbb{E}_\pi(G_{t+1}|S_{t+1}, S_t)|S_t] = \mathbb{E}_\pi[G_{t+1}|S_t]$$

According to the Markov property, we can also get that

$$\mathbb{E}_\pi(G_{t+1}|S_{t+1}, S_t) = \mathbb{E}_\pi(G_{t+1}|S_{t+1}) = v_\pi(S_{t+1}) \Rightarrow \mathbb{E}_\pi[\mathbb{E}_\pi(G_{t+1}|S_{t+1}, S_t)|S_t] = \mathbb{E}_\pi[\mathbb{E}_\pi(G_{t+1}|S_{t+1})|S_t] = \mathbb{E}_\pi[v_\pi(S_{t+1})|S_t]$$

Combine the above two equations, we can get that

$$\mathbb{E}_\pi[G_{t+1}|S_t] = \mathbb{E}_\pi[v_\pi(S_{t+1})|S_t] \Rightarrow \quad \forall s \in \mathcal{S},\ \mathbb{E}_\pi[G_{t+1}|S_t = s] = \mathbb{E}_\pi[v_\pi(S_{t+1})|S_t = s]$$

Put it into the original eqution, we can get that

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[R_{t+1}|S_t = s] + \gamma \mathbb{E}_\pi[G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1}|S_t = s] + \gamma \mathbb{E}_\pi[v_\pi(S_{t+1})|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned}$$

1.2 The state-action value function's expectation form: From the definition, we can get that

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

Change S_t, A_t into random variables, we have:

$$\begin{gathered} q_\pi(S_t, A_t) = \mathbb{E}_\pi[G_t|S_t, A_t] \\ q_\pi(S_t, A_{t+1}) = \mathbb{E}_\pi[G_{t+1}|S_{t+1}, A_{t+1}] \end{gathered}$$

Then again, use the Adam's law with extra conditioning, let $Y = G_{t+1}$, $Z = (S_t, A_t)$ and $X = (S_{t+1}, A_{t+1})$:

$$\begin{aligned} & \mathbb{E}_\pi[G_{t+1}|S_t, A_t] \\ \text{(Adam's law)} \quad &= \mathbb{E}_\pi[\mathbb{E}_\pi[G_{t+1}|S_{t+1}, A_{t+1}, S_t, A_t]|S_t, A_t] \\ \text{(Markov property)} \quad &= \mathbb{E}_\pi[\mathbb{E}_\pi[G_{t+1}|S_{t+1}, A_{t+1}]|S_t, A_t] \\ &= \mathbb{E}_\pi[q_\pi(S_{t+1}, A_{t+1})|S_t, A_t] \\ \Rightarrow \quad \forall a \in \mathcal{A}, s \in \mathcal{S}, \mathbb{E}_\pi[G_{t+1}|S_t = s, A_t = a] &= \mathbb{E}_\pi[q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \end{aligned}$$

Problem 1 continued on next page...

Put it into the original eqution, we can get that

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi[G_t|S_t=s,A_t=a] \\
&= \mathbb{E}_\pi[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\ldots|S_t=s,A_t=a] \\
&= \mathbb{E}_\pi[R_{t+1}+\gamma(R_{t+2}+\gamma R_{t+3}+\ldots)|S_t=s,A_t=a] \\
&= \mathbb{E}_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s,A_t=a] \\
&= \mathbb{E}_\pi[R_{t+1}|S_t=s,A_t=a]+\gamma\mathbb{E}_\pi[G_{t+1}|S_t=s,A_t=a] \\
\text{(last proof)} \quad &= \mathbb{E}_\pi[R_{t+1}|S_t=s,A_t=a]+\gamma\mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a] \\
&= \mathbb{E}_\pi[R_{t+1}+\gamma q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a]
\end{aligned}
$$

2.1. Summation form, $q \to v$:
Directly using LOTE to the definition of state value function, we can get that

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t=s] \\
\text{(LOTE)} \quad &= \sum_{a\in\mathcal{A}} \mathbb{E}_\pi[G_t|S_t=s,A_t=a]P(A_t=a|S_t=s) \\
&= \sum_{a\in\mathcal{A}} q_\pi(s,a)\pi(a|s)
\end{aligned}
$$

2.2. Summation form, $v \to q$:

$$
\begin{aligned}
&\mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_{t+1}=s',S_t=s,A_t=a] \\
\text{(LOTE)} \quad &= \sum_{a'\in\mathcal{A}} \mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_{t+1}=s',A_{t+1}=a',S_t=s,A_t=a]P(A_{t+1}=a'|S_{t+1}=s',S_t=s,A_t=a) \\
\text{(Markov property)} \quad &= \sum_{a'\in\mathcal{A}} \mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_{t+1}=s',A_{t+1}=a']P(A_{t+1}=a'|S_{t+1}=s') \\
&= \sum_{a'\in\mathcal{A}} a_\pi(s',a')\pi(a'|s') \\
\text{(Conclusion in 2.1)} \quad &= v^\pi(s')
\end{aligned}
$$

Using LOTE, put the above conclusion, we can get that

$$
\begin{aligned}
&\mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a] \\
\text{(LOTE)} \quad &= \sum_{s'\in\mathcal{S}} \mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_{t+1}=s',A_t=a,S_t=s]P(A_{t+1}=a|S_t=s,A_t=a) \\
\text{(Last proof)} \quad &= \sum_{s'\in\mathcal{S}} v_\pi(s')\mathcal{P}^a_{s,s'}
\end{aligned}
$$

The put it into the state-action value function's expectation form:

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi[R_{t+1}+\gamma q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a] \\
&= \mathbb{E}_\pi[R_{t+1}|S_t=s,A_t=a]+\gamma\mathbb{E}_\pi[q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a] \\
&= \mathcal{R}^a_s+\gamma\sum_{s'\in\mathcal{S}} v_\pi(s')\mathcal{P}^a_{s,s'}
\end{aligned}
$$

2.3. Summation form, $v \to v$:
Combining 2.1 and 2.2, we can get that

$$
\begin{aligned}
v_\pi(s) &= \sum_{a\in\mathcal{A}} \pi(a|s)q_\pi(s,a) \\
&= \sum_{a\in\mathcal{A}} \pi(a|s)\left(\mathcal{R}^a_s+\gamma\sum_{s'\in\mathcal{S}} \mathcal{P}^a_{s,s'}v_\pi(s')\right)
\end{aligned}
$$

Problem 1 continued on next page...

2.4. Summation form, $q \to q$:
Combining 2.2 and 2.1, we can get that

$$\begin{aligned} q_\pi(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_\pi(s') \\ &= \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a \left(\sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a') \right) \end{aligned}$$

So above all, we have proved the exceptation forms and summation forms of Bellman Expectation Equations for MDPs:

$$\begin{cases} v_\pi(s) & = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\ q_\pi(s,a) & = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \end{cases} \Rightarrow \begin{cases} v_\pi(s) & = \sum\limits_{a \in \mathcal{A}} q_\pi(s,a)\pi(a|s) \\ q_\pi(s,a) & = \mathcal{R}_s^a + \gamma \sum\limits_{s' \in \mathcal{S}} v_\pi(s') \mathcal{P}_{s,s'}^a \\ v_\pi(s) & = \sum\limits_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum\limits_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_\pi(s') \right) \\ q_\pi(s,a) & = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a \left(\sum\limits_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a') \right) \end{cases}$$

(b) Similarly to the Bellman Exceptation Equation, the Bellman Optimality Equation also has 4 forms.
1. $q_* \to v_*$:
According to the definition of optimal, and conclusions from (a), we can get that:

$$\begin{aligned} v_*(s) &= \max_\pi v_\pi(s) \\ \text{((a) 2.1)} \quad &= \max_\pi \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) \\ \text{(Linear Programming)} \quad &= \max_\pi \max_a q_\pi(s,a) \\ &= \max_a \max_\pi q_\pi(s,a) \\ \text{(Definition of } q_*) \quad &= \max_a q_*(s,a) \end{aligned}$$

The third equation holds due to $\sum\limits_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$ is the convex combination of $q_\pi(s,a)$, and it's maximum value is obviously the maximum value of $q_\pi(s,a)$ with the conclusion in the convex optimization.
2. $v_* \to q_*$:
From (a) 2.2, we can get that

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} v_\pi(s') \mathcal{P}_{s,s'}^a$$

According to the definition of q_*, we can get that

$$q_*(s,a) = \max_\pi q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a \left(\max_\pi v_\pi(s') \right) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s')$$

Also, according to the definition of reward functions, we can get that

$$\mathcal{R}_s^a = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$$

And since when $S_{t+1} = s'$ is given, the expectation becomes a constant, i.e. $\mathbb{E}[v_*(S_{t+1}) | S_{t+1} = s'] = v_*(s')$, so

$$\begin{aligned} & \mathbb{E}[v_*(S_{t+1}) | S_t = s, A_t = a] \\ \text{(LOTE)} \quad &= \sum_{s' \in \mathcal{S}} \mathbb{E}[v_*(S_{t+1}) | S_{t+1} = s', S_t = s, A_t = a] \cdot P(S_{t+1} = s' | S_t = s, A_t = a) \\ &= \sum_{s' \in \mathcal{S}} v_*(s') \mathcal{P}_{s,s'}^a \end{aligned}$$

Problem 1 continued on next page...

Thus, combine all infomation above, we can get that

$$\begin{aligned} q_*(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \\ &= \mathbb{E}(R_{t+1}|S_t = s, A_t = a) + \gamma\mathbb{E}[v_*(S_{t+1})|S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = t] \end{aligned}$$

So above all, we have proved that

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \Leftrightarrow q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a]$$

3. $v_* \to v_*$:
Combine 1. and 2., we can get that

$$\begin{aligned} v_*(s) &= \max_a q_*(s,a) \\ \text{(From 2.)} \quad &= \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s')\right) \\ \text{(From 2.)} \quad &= \max_a \left(\mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a]\right) \end{aligned}$$

4. $q_* \to q_*$: Combine 2. and 1., we can get that

$$\begin{aligned} q_*(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \\ \text{(From 1.)} \quad &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a \max_{a'} q_*(s',a') \end{aligned}$$

Also, according to the definition of reward functions, we can get that

$$\mathcal{R}_s^a = \mathbb{E}(R_{t+1}|S_t = s, A_t = a)$$

And since when $S_{t+1} = s'$ is given, the expectation becomes a constant, i.e.

$$\mathbb{E}[\max_{a'} q_*(S_{t+1},a')|S_{t+1} = s'] = \max_{a'} q_*(s',a')$$

Use this property, we can get that

$$\begin{aligned} &\mathbb{E}[\max_{a'} q_*(S_{t+1},a')|S_t = s, A_t = a] \\ \text{(LOTE)} \quad &= \sum_{s'\in\mathcal{S}} \mathbb{E}[\max_{a'} q_*(S_{t+1},a')|S_{t+1} = s', S_t = s, A_t = a] \cdot P(S_{t+1} = s'|S_t = s, A_t = a) \\ &= \sum_{s'\in\mathcal{S}} \max_{a'} q_*(s',a')\mathcal{P}_{s,s'}^a \end{aligned}$$

Thus, combine all infomation above, we can get that

$$\begin{aligned} q_*(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a \max_{a'} q_*(s',a') \\ &= \mathbb{E}(R_{t+1}|S_t = s, A_t = a) + \gamma\mathbb{E}[\max_{a'} q_*(S_{t+1},a')|S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a')|S_t = s, A_t = t] \end{aligned}$$

So above all, we have proved that

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a \max_{a'} q_*(s',a') \Leftrightarrow q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a')|S_t = s, A_t = t]$$

Problem 1 continued on next page...

So, we have proved the Bellman Optimality Equations for MDPs:

$$\begin{cases} v_*(s) = \max_a q_*(s,a) & \\ q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') & \Leftrightarrow q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ v_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \right) & \Leftrightarrow v_*(s) = \max_a \left(\mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \right) \\ q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a \max_{a'} q_*(s',a') & \Leftrightarrow q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a') | S_t = s, A_t = t] \end{cases}$$

(c) What we need to prove is that for a policy $\pi(s)$, applying Policy Improvement by acting greedily to generate

$$\pi'(s) = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, q_\pi(s,a)$$

Then both the state value function and the state-action value function will be both improved. Here we consider the ppolicy is deterministic, i.e. $a = \pi(s)$.
1. state-action value function:
According to the definition of policy improvement, we can get that $\forall s \in \mathcal{S}$:

$$\begin{aligned} q_\pi(s,a) &= q_\pi(s,\pi'(s)) \\ &= \max_{a\in\mathcal{A}} q_\pi(s,a) \quad \text{(Update rule)} \\ &\geq q_\pi(s,a)\ \forall a \in \mathcal{A} \\ &\geq q_\pi(s,\pi(s)) \\ &= v_\pi(s) \end{aligned}$$

2. action value function:
Here we define $\mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$ representing that the expected discounted reward when starting in state s, choosing action accoding to policy π' for the next step, and then choos actions **according to policy π thereafter**! In other words, the next one step is using π', and the following steps are using π. Thus:

$$\begin{aligned} & v_\pi(s) \leq q_\pi(s,\pi'(s)) \quad \text{(Last proof)} \\ \text{(Definition of } q_\pi(s,\pi'(s))) \quad & = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\ \text{(Definition above)} \quad & = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\ \text{(Last proof extends to } S_{t+1}) \quad & \leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1},\pi(S_{t+1})) | S_t = s] \\ & = \mathbb{E}_{\pi'}[R_{t+1} | S_t = s] + \gamma \mathbb{E}_{\pi'}[q_\pi(S_{t+1},\pi(S_{t+1})) | S_t = s] \end{aligned}$$

According to the definition, we have

$$\begin{aligned} & q_\pi(S_t,\pi'(S_t)) = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t, A_t = \pi'(S_t)] \\ \Rightarrow \quad & q_\pi(S_{t+1},\pi'(S_{t+1})) = \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_t = \pi'(S_{t+1})] \end{aligned}$$

Thus, we have

$$\begin{aligned} & \mathbb{E}_{\pi'}[q_\pi(S_{t+1},\pi(S_{t+1})) | S_t = s] \\ \text{(above definition)} & = \mathbb{E}_{\pi'}[\mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_t = \pi'(S_{t+1})] | S_t = s] \\ \text{(Markov property)} & = \mathbb{E}_{\pi'}[\mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_t = \pi'(S_{t+1}), S_t = s] | S_t = s] \\ \text{(Adam's law with extra conditioning)} & = \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_t = s] \end{aligned}$$

Problem 1 continued on next page...

Put this into the origin proof, we can get that

$$
\begin{aligned}
v_\pi(s) &\leq \mathbb{E}_{\pi'}[R_{t+1}|S_t = s] + \gamma\mathbb{E}_{\pi'}[q_\pi(S_{t+1}, \pi(S_{t+1}))|S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1}|S_t = s] + \gamma\mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2})|S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma v_\pi(S_{t+2})|S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma v_\pi(S_{t+3})|S_t = s] \\
&\qquad\vdots \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+3} + \cdots|S_t = s] \\
&= \mathbb{E}_{\pi'}[G_t|S_t = s] \\
&= v_{\pi'}(s)
\end{aligned}
$$

So, we have proved that

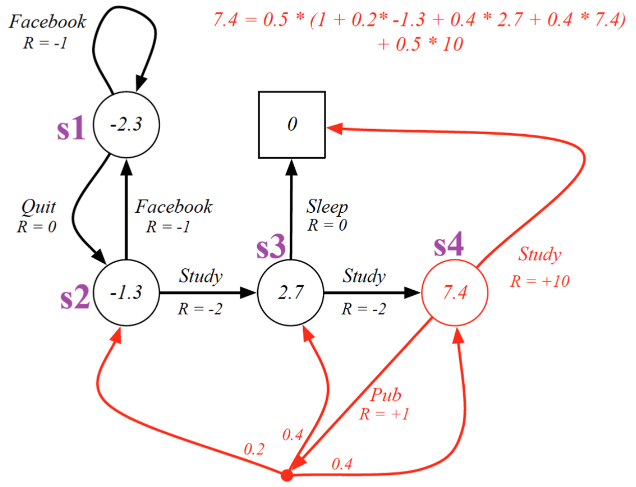
$$v_\pi(s) \leq v_{\pi'}(s)$$

Which means that the Policy Improvement by acting greedily will improve the state-action value function.

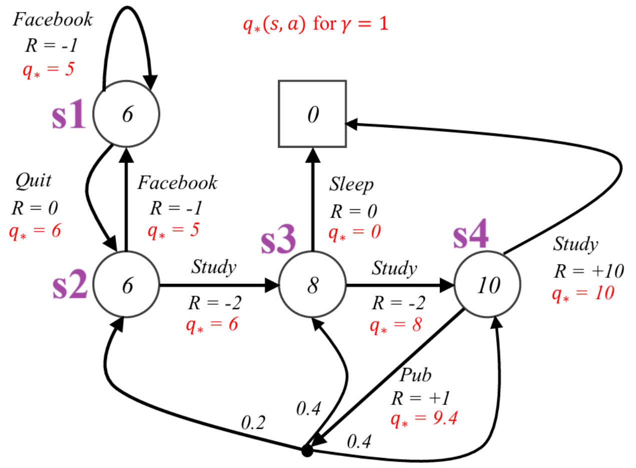
Problem 2

Student MDPs

(a) Given the equal option policy, reproduce the state values & state-action values for student MDP with both theoretical method and iterative policy evaluation method. Then discuss the pros and cons of each method.



(b) Reproduce the optimal state values, optimal state-action values, and optimal policy for student MDP with theoretical method, policy iteration method and value iteration method. Then discuss the pros and cons of each method.



Solution

We can number the states for convience, written on the figure with purple $s_1, \cdots, s_4$(The terminal state is ignored, whose state value and action-state values are all 0). And the action space is $\mathcal{A} =${Facebook, Quit, Study, Sleep, Pub}. And the rewards and transition probabilities are given in the figure. And the discount factor is $\gamma = 1$. The codes for (a), (b) could be found in the 'hw5_code.ipynb' file.

(a) From the Bellman Expectation Equations for MDPs, we can get that

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_\pi(s') \right)$$
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_\pi(s')$$

Problem 2 continued on next page...

1. Theoretical method:
We can use the inplace method to solve the state values $v_\pi(s)$, i.e. solve the equations:

$$\begin{cases} v_\pi(s_1) = 0.5 * (\mathcal{R}_{s_1}^{\text{Facebook}} + 1 * 1 * v_\pi(s_1)) + 0.5 * (\mathcal{R}_{s_1}^{\text{Quit}} + 1 * 1 * v_\pi(s_2)) \\ v_\pi(s_2) = 0.5 * (\mathcal{R}_{s_2}^{\text{Facebook}} + 1 * 1 * v_\pi(s_1)) + 0.5 * (\mathcal{R}_{s_2}^{\text{Study}} + 1 * 1 * v_\pi(s_3)) \\ v_\pi(s_3) = 0.5 * (\mathcal{R}_{s_3}^{\text{Study}} + 1 * 1 * v_\pi(s_4)) + 0.5 * (\mathcal{R}_{s_3}^{\text{Sleep}} + 1 * 1 * 0) \\ v_\pi(s_4) = 0.5 * (\mathcal{R}_{s_4}^{\text{Study}} + 1 * 1 * 0) + 0.5 * \left(\mathcal{R}_{s_4}^{\text{Pub}} + 1 * (0.2 * v_\pi(s_2) + 0.4 * v_\pi(s_3) + 0.4 * v_\pi(s_4))\right) \end{cases}$$

$$\Rightarrow \begin{cases} v_\pi(s_1) - v_\pi(s_2) = -1 \\ -v_\pi(s_1) + 2v_\pi(s_2) - v_\pi(s_3) = -3 \\ 2v_\pi(s_3) - v_\pi(s_4) = -2 \\ -v_\pi(s_2) - 2v_\pi(s_3) + 8v_\pi(s_4) = 55 \end{cases}$$

$$\Rightarrow \begin{cases} v_\pi(s_1) = -2.3076923076923066 \\ v_\pi(s_2) = -1.3076923076923066 \\ v_\pi(s_3) = 2.6923076923076934 \\ v_\pi(s_4) = 7.384615384615385 \end{cases} \approx \begin{cases} v_\pi(s_1) = 2.3 \\ v_\pi(s_2) = -1.3 \\ v_\pi(s_3) = 2.7 \\ v_\pi(s_4) = 7.4 \end{cases}$$

With calculated state values, we can get the state-action values:

$$\begin{cases} q_\pi(s_1, \text{Facebook}) & = -3.3076923076923066 \\ q_\pi(s_1, \text{Quit}) & = -1.3076923076923066 \\ q_\pi(s_2, \text{Facebook}) & = -3.3076923076923066 \\ q_\pi(s_2, \text{Study}) & = 0.6923076923076934 \\ q_\pi(s_3, \text{Study}) & = 5.384615384615385 \\ q_\pi(s_3, \text{Sleep}) & = 0 \\ q_\pi(s_4, \text{Study}) & = 10 \\ q_\pi(s_4, \text{Pub}) & = 4.76923076923077 \end{cases} \approx \begin{cases} q_\pi(s_1, \text{Facebook}) & = -3.3 \\ q_\pi(s_1, \text{Quit}) & = -1.3 \\ q_\pi(s_2, \text{Facebook}) & = -3.3 \\ q_\pi(s_2, \text{Study}) & = 0.7 \\ q_\pi(s_3, \text{Study}) & = 5.4 \\ q_\pi(s_3, \text{Sleep}) & = 0 \\ q_\pi(s_4, \text{Study}) & = 10 \\ q_\pi(s_4, \text{Pub}) & = 4.8 \end{cases}$$

2. Iterative Policy Evaluation Method:
The threshold is set to be $\epsilon = 10^{-3}$, after 26 iterations, the state values convergence, and the results are

$$\begin{cases} v_\pi(s_1) = -2.3116692858874535 \\ v_\pi(s_2) = -1.3102871136591983 \\ v_\pi(s_3) = 2.6919968668115835 \\ v_\pi(s_4) = 7.384101760095685 \end{cases} \approx \begin{cases} v_\pi(s_1) = 2.3 \\ v_\pi(s_2) = -1.3 \\ v_\pi(s_3) = 2.7 \\ v_\pi(s_4) = 7.4 \end{cases}$$

The correspondence state-action values are

$$\begin{cases} q_\pi(s_1, \text{Facebook}) & = -3.3116692858874535 \\ q_\pi(s_1, \text{Quit}) & = -1.3102871136591983 \\ q_\pi(s_2, \text{Facebook}) & = -3.3116692858874535 \\ q_\pi(s_2, \text{Study}) & = 0.6919968668115835 \\ q_\pi(s_3, \text{Study}) & = 5.384101760095685 \\ q_\pi(s_3, \text{Sleep}) & = 0 \\ q_\pi(s_4, \text{Study}) & = 10 \\ q_\pi(s_4, \text{Pub}) & = 4.768382028031068 \end{cases} \approx \begin{cases} q_\pi(s_1, \text{Facebook}) & = -3.3 \\ q_\pi(s_1, \text{Quit}) & = -1.3 \\ q_\pi(s_2, \text{Facebook}) & = -3.3 \\ q_\pi(s_2, \text{Study}) & = 0.7 \\ q_\pi(s_3, \text{Study}) & = 5.4 \\ q_\pi(s_3, \text{Sleep}) & = 0 \\ q_\pi(s_4, \text{Study}) & = 10 \\ q_\pi(s_4, \text{Pub}) & = 4.8 \end{cases}$$

Problem 2 continued on next page...

3. Pros and cons:
Theoretical method calculates the optimal state values $v_*(s)$ by solving a linear system, and its advantage lies in providing an exact solution in one go. It constructs the matrix $(I-\gamma\mathcal{P}^\pi)^{-1}\mathcal{R}^\pi$ to solve for the optimal state values, avoiding the need for iterative processes. The result is accurate, and when the model is relatively small in scale, the complexity is relatively low.
However, the disadvantage of this method is that getting inverse matrix requires $O(n^3)$ time complexity, where n is the number of states. When the state space and action space are large, the theoretical method may become computationally expensive and inefficient.

Iterative policy evaluation method iteratively updates the state value function to approximate the optimal solution. It can gradually converge to the optimal solution it can still effectively perform evaluations when the spaces are large. It may not be the accurate solution, but close to the accurate one and computational friendly.
However, its disadvantage is that it may have cases that converges slowly. When chasing for a more accurate solution, i.e. setting ϵ to be smaller, it may converge much slower.

(b) Let $v_*(s)$ be the optimal state value function, and $q_*(s,a)$ be the optimal state-action value function. The Bellman Optimality Equation is:

$$\begin{aligned} v_*(s) &= \max_{a\in\mathcal{A}}\left(\mathcal{R}_s^a+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s,s'}^a v_*(s')\right)\\ q_*(s,a) &= \mathcal{R}_s^a+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s,s'}^a v_*(s')\\ \pi_*(a|s) &= \begin{cases}1 & \text{if } a\in\operatorname*{argmax}_{a\in\mathcal{A}} q_*(s,a)\\ 0 & \text{otherwise}\end{cases}\end{aligned}$$

Thus, we could get the optimal state value function $v_*(s)$ with different methods, then get optimal state-action value function $q_*(s,a)$ using $v_*(s)$ and final get the optimal policy $\pi^*(a|s)$ using $q_*(s,a)$.
1. Theoretical method:
We can re-write the definition of $v_*(s)$ into the an inequality form:

$$\begin{aligned} v_*(s)=\max_{a\in\mathcal{A}}\left(\mathcal{R}_s^a+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s,s'}^a v_*(s')\right) &\Rightarrow v_*(s)\geq \mathcal{R}_s^a+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s,s'}^a v_*(s')\quad \forall a\in A\\ &\Rightarrow -v_*(s)+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s,s'}^a v_*(s')\leq -\mathcal{R}_s^a\quad \forall a\in A\end{aligned}$$

So the Bellman Optimality Equation could be re-written into a set of inequality constrains. As we known of the basic knowledge of convex optimization, the optimal solution must be lying on the inconstraint set for the Linear Programming problem, thus we can add a linear objective function to ensure the inequality constrains try to be equal. So the final LP problem is:

$$\begin{aligned} \min_v \quad & \sum_{s\in\mathcal{S}} v(s)\\ \text{s.t.} \quad & -v(s)+\gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s,s'}^a v(s')\leq -\mathcal{R}_s^a\quad \forall s\in\mathcal{S}, a\in\mathcal{A}\end{aligned}$$

Thus, solve the LP problem with optimizers, we can get the theoritical optimal state values, optimal state-

Problem 2 continued on next page...

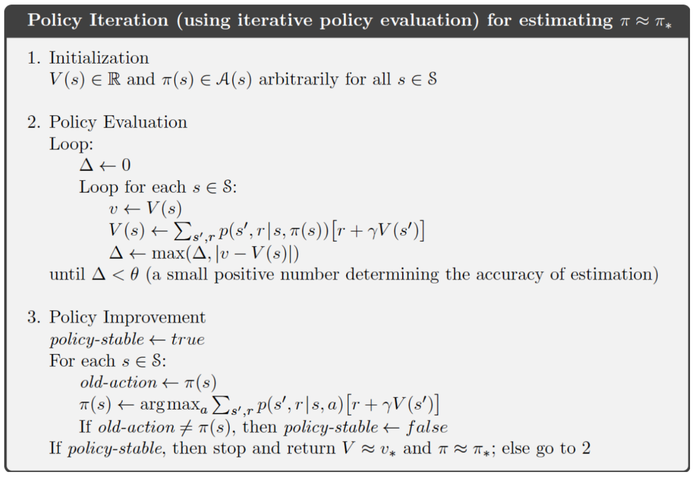
action values and optimal policy. The constructed LP is:

$$
\begin{aligned}
\min_{v} \quad & \sum_{s\in\mathcal{S}} v(s) \\
\text{s.t.} \quad & \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0.2 & 0.4 & -0.6 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} v(s_1) \\ v(s_2) \\ v(s_3) \\ v(s_4) \end{pmatrix} \leq \begin{pmatrix} 1 \\ 0 \\ 1 \\ 2 \\ 0 \\ 2 \\ -1 \\ -10 \end{pmatrix}
\end{aligned}
$$

The result is:

$$
\begin{cases} v_*(s_1) = 6 \\ v_*(s_2) = 6 \\ v_*(s_3) = 8 \\ v_*(s_4) = 10 \end{cases} \Rightarrow \begin{cases} q_*(s_1, \text{Facebook}) & = 5 \\ q_*(s_1, \text{Quit}) & = 6 \\ q_*(s_2, \text{Facebook}) & = 5 \\ q_*(s_2, \text{Study}) & = 6 \\ q_*(s_3, \text{Study}) & = 8 \\ q_*(s_3, \text{Sleep}) & = 0 \\ q_*(s_4, \text{Study}) & = 10 \\ q_*(s_4, \text{Pub}) & = 9.4 \end{cases} \Rightarrow \begin{cases} \pi_*(s_1) = \text{Quit} \\ \pi_*(s_2) = \text{Study} \\ \pi_*(s_3) = \text{Study} \\ \pi_*(s_4) = \text{Study} \end{cases}
$$

2. Policy Iteration Method:
We can applying policy iteration method through the following pseudocode to get $v_*(s)$ and $\pi_*(s)$, then get $q_*(s,a)$ by Bellman Optimality Equation using $v_*(s)$:
The result is:



Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ true
   For each $s \in \mathcal{S}$:
       old-action $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If old-action $\neq \pi(s)$, then policy-stable $\leftarrow$ false
   If policy-stable, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

$$
\begin{cases} v_*(s_1) = 6 \\ v_*(s_2) = 6 \\ v_*(s_3) = 8 \\ v_*(s_4) = 10 \end{cases} \Rightarrow \begin{cases} q_*(s_1, \text{Facebook}) & = 5 \\ q_*(s_1, \text{Quit}) & = 6 \\ q_*(s_2, \text{Facebook}) & = 5 \\ q_*(s_2, \text{Study}) & = 6 \\ q_*(s_3, \text{Study}) & = 8 \\ q_*(s_3, \text{Sleep}) & = 0 \\ q_*(s_4, \text{Study}) & = 10 \\ q_*(s_4, \text{Pub}) & = 9.4 \end{cases} \Rightarrow \begin{cases} \pi_*(s_1) = \text{Quit} \\ \pi_*(s_2) = \text{Study} \\ \pi_*(s_3) = \text{Study} \\ \pi_*(s_4) = \text{Study} \end{cases}
$$

Problem 2 continued on next page...

It totally run policy iteration 2 times, and the policy evaluation used 24+4=30 iterations.
3. Value Iteration Method:
We can applying policy iteration method through the following pseudocode to get $v_*(s)$ and $q_*(s,a)$, then get $\pi_*(s)$ through $q_*(s,a)$:

Value Iteration, for estimating $\pi \approx \pi_*$

```
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
|  $\Delta \leftarrow 0$
|  Loop for each $s \in \mathcal{S}$:
|      $v \leftarrow V(s)$
|      $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
|      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

It totally run value iteration 4 times. The result is:

$$\begin{cases} v_*(s_1) = 6 \\ v_*(s_2) = 6 \\ v_*(s_3) = 8 \\ v_*(s_4) = 10 \end{cases} \Rightarrow \begin{cases} q_*(s_1, \text{Facebook}) & = 5 \\ q_*(s_1, \text{Quit}) & = 6 \\ q_*(s_2, \text{Facebook}) & = 5 \\ q_*(s_2, \text{Study}) & = 6 \\ q_*(s_3, \text{Study}) & = 8 \\ q_*(s_3, \text{Sleep}) & = 0 \\ q_*(s_4, \text{Study}) & = 10 \\ q_*(s_4, \text{Pub}) & = 9.4 \end{cases} \Rightarrow \begin{cases} \pi_*(s_1) = \text{Quit} \\ \pi_*(s_2) = \text{Study} \\ \pi_*(s_3) = \text{Study} \\ \pi_*(s_4) = \text{Study} \end{cases}$$

4. Pros and cons:
Theoretical method:
Advantage: It can get an exact and optimal solution by formulating the Bellman Optimality Equation as a set of linear inequalities, then solving the problem using linear programming. This method guarantees that the resulting state values and policies are optimal because it directly solves the problem mathematically without iteration. It is highly efficient for small-scale problems.
Disadvantages: It is not suitable for large-scale problems as solving the optimization problem requires a much more time, which is not efficient.

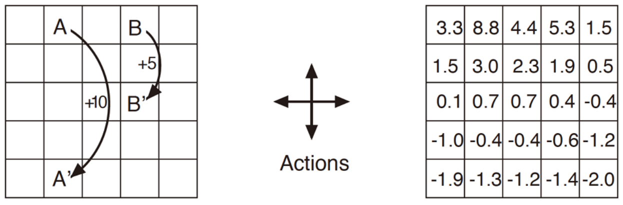
Policy Iteration:
Advantage: By alternating between policy evaluation and policy improvement, the optimal policy can be found and the corresponding optimal state value $V_x(s)$can be calculated, resulting in high accuracy for policy evaluation.
Disadvantages: Requires repeated strategy evaluation and improvement, slow convergence speed, especially when the state space is large.

Value Iteration:
Advantage: By continuously updating the state value function, it is possible to directly approximate the optimal state value $V_x(s)$. It is based on greedy selection, calculating the optimal action only at last.
Disadvantage: Due to updating the values of all states at each iteration, it may require more iterations to converge when the state space is large.

Problem 3

5x5 Grid World(Example 3.5 Gridworld in the book "Reinforcement Learning: An Introduction" , the second edition)

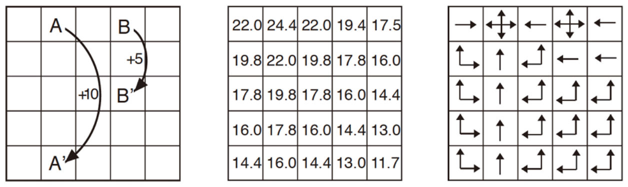
(a) Find the state values under the uniform random policy with both theoretical method and iterative policy evaluation method.



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(a) (b)

(b) Reproduce the optimal state values, optimal state-action values, and optimal policy with theoretical method, policy iteration method and value iteration method.



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

a) gridworld b) v_* c) π_*

Solution

In the book "Reinforcement Learning: An Introduction" , the second edition, Example 3.5 Gridworld. The setting is that in each cell, four actions are possible: north, south, east and west. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of -1. Other actions result in a reward of 0, except those that move the agent out of the special states A and B. From state A, all four actions yield a reward of +10 and take the agent to A'. From state B, all actions yield a reward of +5 and take the agent to B'.

The discount factor is set to be $\gamma = 0.9$. Let the state value function under policy π be $v_\pi(s)$. The action space is $\mathcal{A} = \{\text{north, south, east, west}\}$, and the state space is ordered by the position of the cells, from left to right, from up to down: $\mathcal{S} = \{s_1, \cdots, s_{25}\}$, additionally, $s_2 = A, s_4 = B, s_{14} = B', s_{22} = A'$. The reward $\mathcal{R}$ is defined above, and the transition probability $\mathcal{P}$ is deterministic, i.e. 1 if it suit the transition policy, 0 otherwise. So when know s, a, we can get the unique s', thus we can get the reward:

$$\mathcal{R}_s^a = \begin{cases} -1 & \text{if } s = s' \\ 10 & \text{if } s = A, s' = A' \\ 5 & \text{if } s = B, s' = B' \\ 0 & \text{otherwise} \end{cases}$$

The codes for (a), (b) could be found in the 'hw5_code.ipynb' file.

(a) From what we have learned, the Bellman Expectation Equation for v_π is

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a v_\pi(s) \right)$$

We can re-write is into the matrix form:

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

Problem 3 continued on next page...

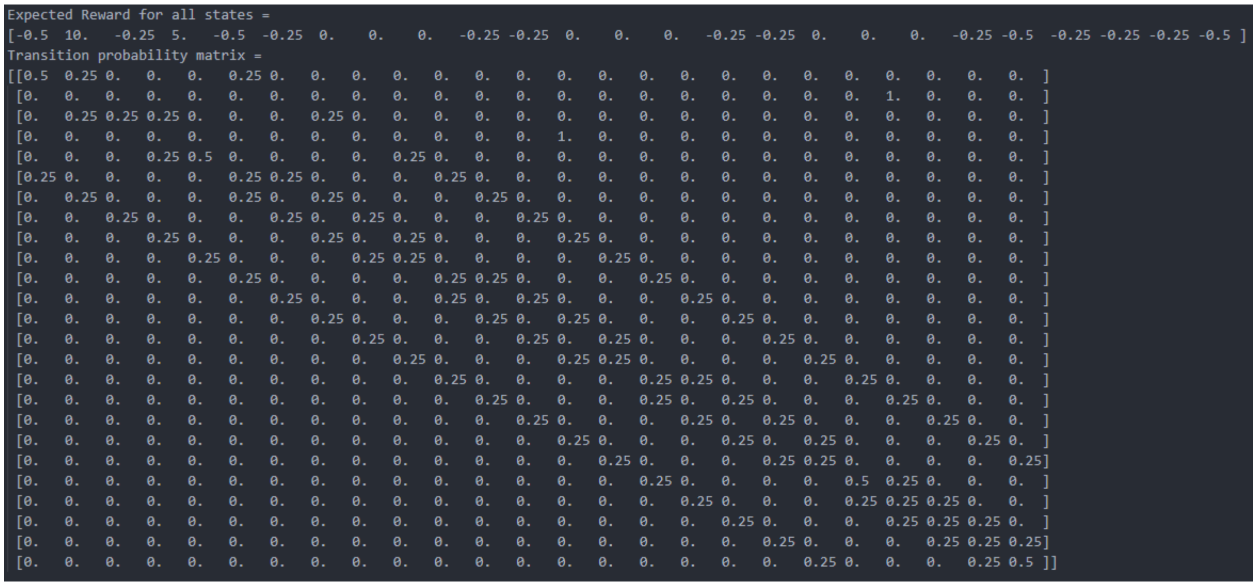
So from the definition

$$\mathcal{R}_s^\pi = \mathbb{E}\left[R_{t+1} | S_t = s\right] = \sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a \mathcal{R}_s^a = \frac{1}{4}\sum_{a\in\mathcal{A}} \mathcal{R}_s^a$$

So we can get

$$\mathcal{R}^\pi = \left(\mathcal{R}_{s_1}^\pi, \cdots, \mathcal{R}_{s_{25}}^\pi\right)^\top$$
$$\mathcal{P}^\pi = \left(\sum_{a\in\mathcal{A}} \mathcal{P}_s^a\right)_{s,s'\text{: if } s \text{ could reach } s' \text{ through } a}$$

And we can get $\mathcal{R}^\pi, \mathcal{P}^\pi$ through coding, the numerical results are shown below.

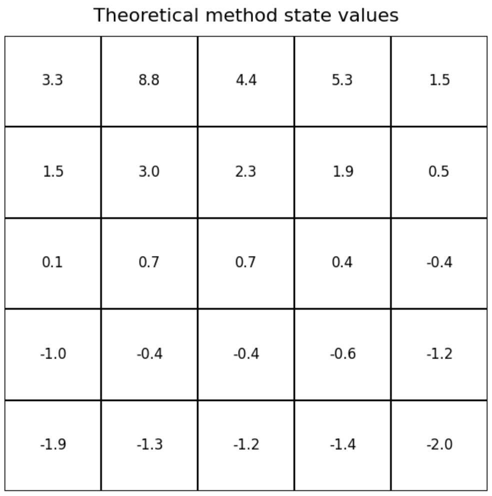


```
Expected Reward for all states =
[-0.5  10.   -0.25  5.   -0.5  -0.25  0.    0.    0.   -0.25 -0.25  0.    0.    0.   -0.25 -0.25  0.    0.    0.   -0.25 -0.5  -0.25 -0.25 -0.25 -0.5 ]
Transition probability matrix =
[[0.5  0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.   0.  ]
 [0.   0.25 0.25 0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.25 0.5  0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.25 0.   0.   0.   0.   0.25 0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.25 0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.25 0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.25 0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.25 0.25 0.   0.   0.   0.25 0.   0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.   0.25 0.   0.   0.   0.25 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.25 0.   0.   0.   0.   0.25]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.   0.5  0.25 0.   0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.25 0.25 0.   0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.25 0.25 0.  ]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.25 0.25]
 [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.5 ]]
```

1. Theoretical method:
Using the matrices above, we can use the matrix form to get the theoretical state values by

$$v_\pi = \left(I - \gamma\mathcal{P}^\pi\right)^{-1}\mathcal{R}^\pi$$

The result is shown as follows:



Theoretical method state values

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

2. Iterative policy evaluation method:
Directly apply the Bellman Expectation Equation iteratively, and use the assignment operator to update the state values. Here we regard the update using synchronous backups, i.e.

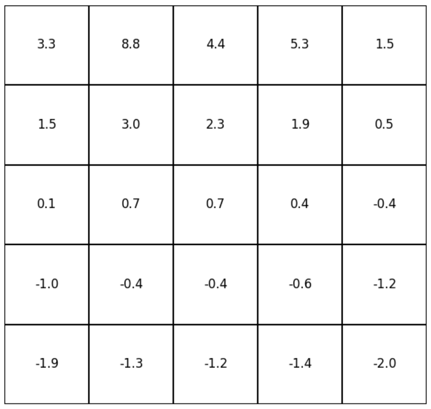
$$v_\pi^{k+1} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi^k$$

Define Δ as the maximum difference between the two iterations, i.e.

$$\Delta = \max_{s\in\mathcal{S}} |v_\pi^{k+1}(s) - v_\pi^k(s)|$$

Initially, we set $v_\pi^0 = \mathbf{0}$. And we set a threshold $\epsilon = 10^{-3}$, and stop the iteration when $\Delta < \epsilon$. The result is shown below. Which totally used 26 iterations to converge.

Iterative policy evaluation method state values



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b) Let $v_*(s)$ be the optimal state value function, and $q_*(s,a)$ be the optimal state-action value function. The Bellman Optimality Equation is:

$$v_*(s) = \max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \right)$$
$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s')$$

Where $\mathcal{P}_{s,s'}^a$ is deterministic, as a valid transition has probability 1, and 0 otherwise. And $\mathcal{R}_s^a$ is defined above. So we can use the Bellman Optimality Equations to get the optimal state values $v_*(s)$ and the optimal state-action values $q_*(s,a)$. Since there might has more than one optimal action, we can define the optimal action set $\mathcal{A}(s) = \operatorname*{argmax}_{a\in\mathcal{A}} q_*(s,a)$. So the optimal policy π^* is defined as:

$$\pi_*(a|s) = \begin{cases} \dfrac{1}{|\mathcal{A}(s)|} & \text{if } a \in \mathcal{A}(s) \\ 0 & \text{otherwise} \end{cases}$$

Thus, we could get the optimal state value function $v_*(s)$ with different methods, then get optimal state-action value function $q_*(s,a)$ using $v_*(s)$ and final get the optimal policy $\pi^*(a|s)$ using $q_*(s,a)$.
1. Theoretical method:
We can re-write the definition of $v_*(s)$ into the an inequality form:

$$v_*(s) = \max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \right) \Rightarrow v_*(s) \geq \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \;\; \forall a \in A$$
$$\Rightarrow -v_*(s) + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a v_*(s') \leq -\mathcal{R}_s^a \;\; \forall a \in A$$

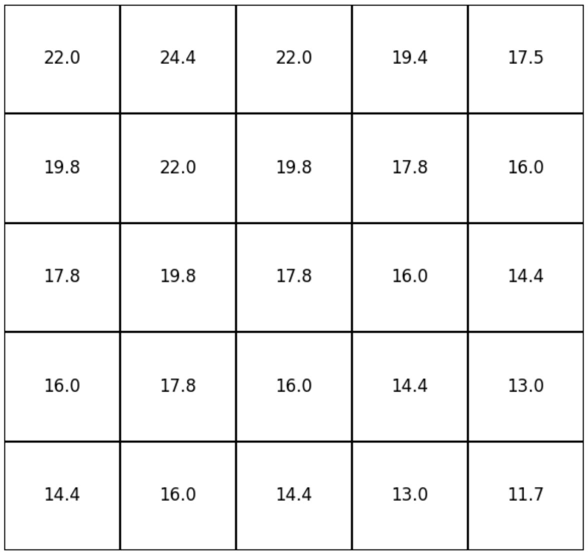
Problem 3 continued on next page...

So the Bellman Optimality Equation could be re-written into a set of inequality constrains. As we known of the basic knowledge of convex optimization, the optimal solution must be lying on the inconstraint set for the Linear Programming problem, thus we can add a linear objective function to ensure the inequality constrains try to be equal. So the final LP problem is:

$$
\begin{aligned}
\min_{v} \quad & \sum_{s \in \mathcal{S}} v(s) \\
\text{s.t.} \quad & -v(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{s,s'} v(s') \leq -\mathcal{R}^{a}_{s} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}
\end{aligned}
$$

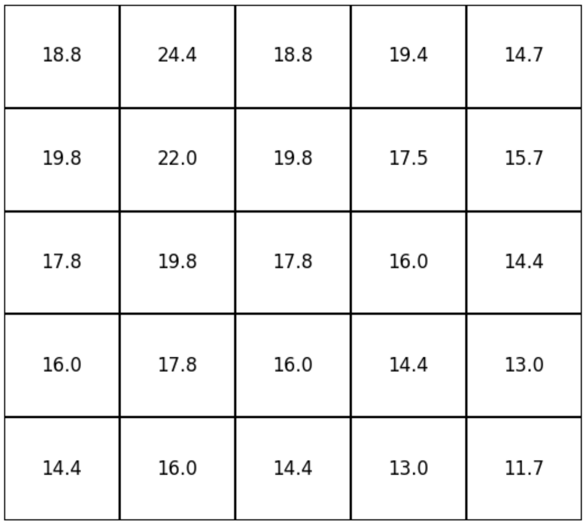
Thus, solve the LP problem with optimizers, we can get the theoritical optimal state values, optimal state-action values and optimal policy. The result is shown below:

Theoritical method optimal state values



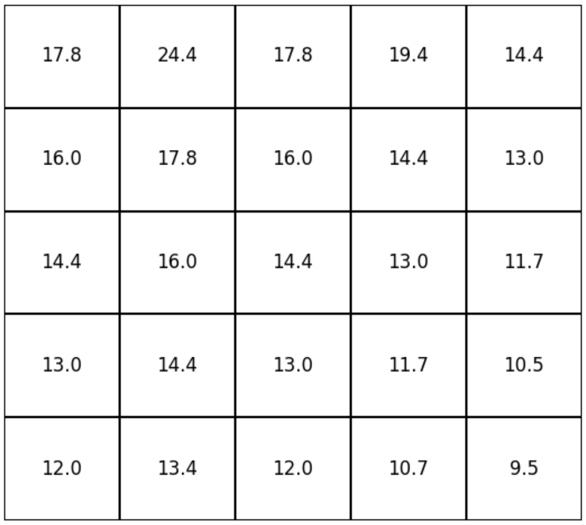
22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Theoretical method optimal state-action values for action north



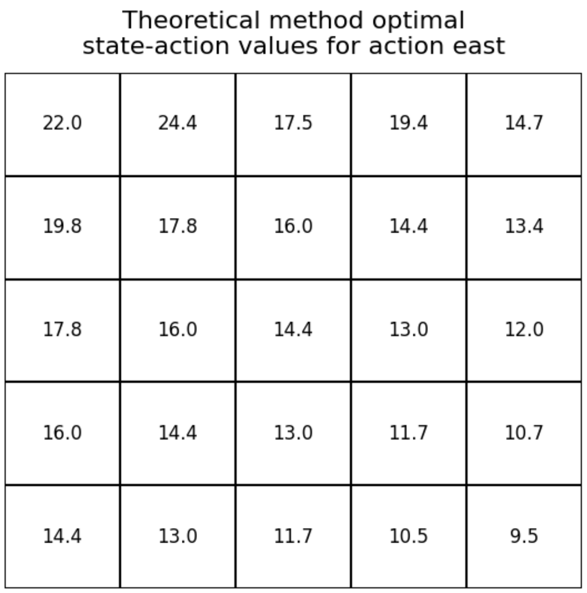
18.8	24.4	18.8	19.4	14.7
19.8	22.0	19.8	17.5	15.7
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Theoretical method optimal state-action values for action south



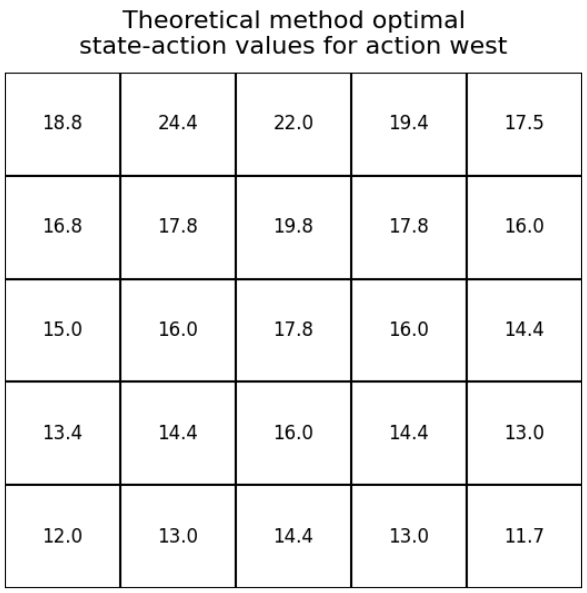
17.8	24.4	17.8	19.4	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7
13.0	14.4	13.0	11.7	10.5
12.0	13.4	12.0	10.7	9.5

Problem 3 continued on next page...



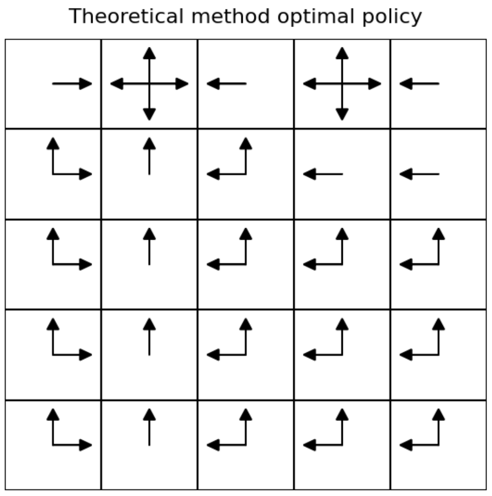
Theoretical method optimal state-action values for action east

22.0	24.4	17.5	19.4	14.7
19.8	17.8	16.0	14.4	13.4
17.8	16.0	14.4	13.0	12.0
16.0	14.4	13.0	11.7	10.7
14.4	13.0	11.7	10.5	9.5

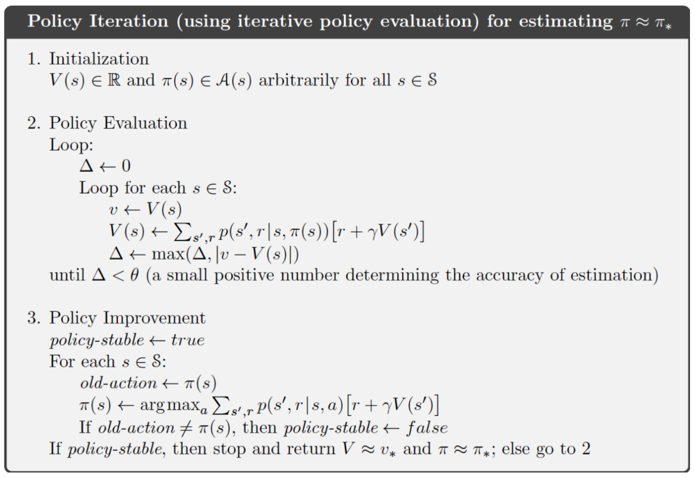


Theoretical method optimal state-action values for action west

18.8	24.4	22.0	19.4	17.5
16.8	17.8	19.8	17.8	16.0
15.0	16.0	17.8	16.0	14.4
13.4	14.4	16.0	14.4	13.0
12.0	13.0	14.4	13.0	11.7



2. Policy iteration method:
We can applying policy iteration method through the following pseudocode to get $v_*(s)$ and $\pi_*(s)$, then get $q_*(s,a)$ by Bellman Optimality Equation using $v_*(s)$:

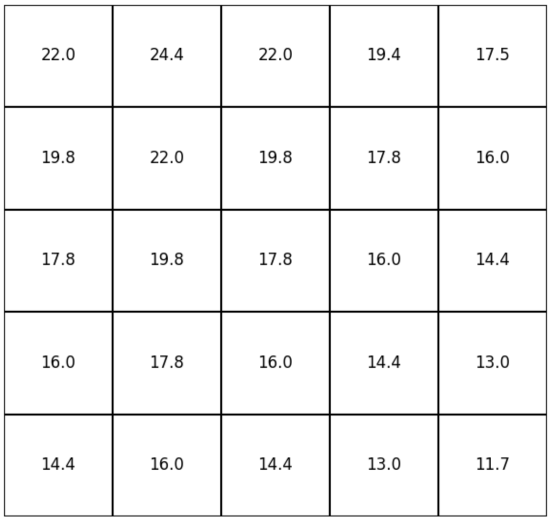


Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

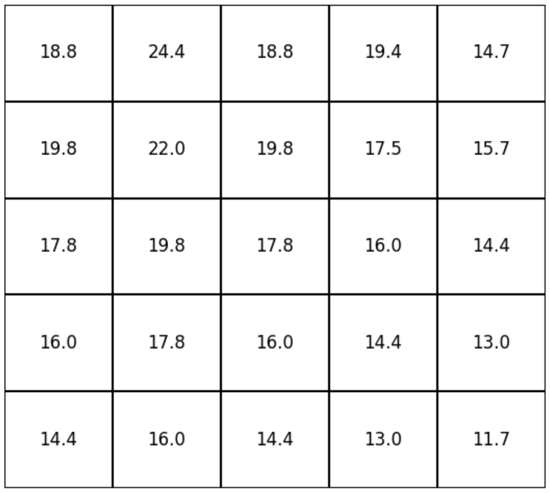
```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ true
   For each $s \in \mathcal{S}$:
       old-action $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If old-action $\neq \pi(s)$, then policy-stable $\leftarrow$ false
   If policy-stable, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

Problem 3 continued on next page...

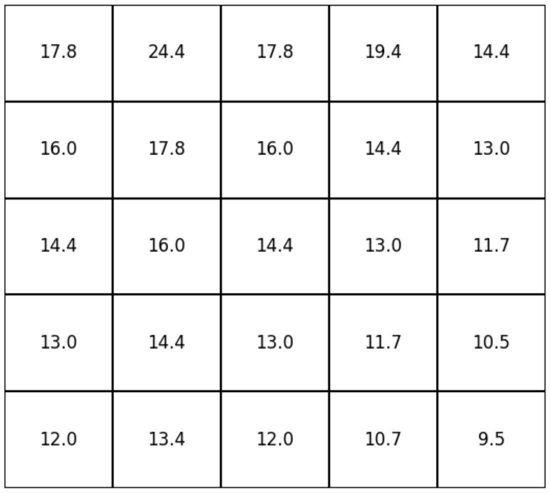
The results are shown as follows:


Policy iteration method optimal state values

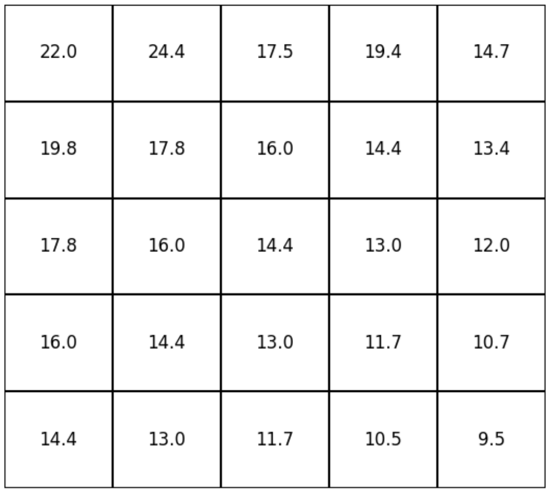
22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7


Policy iteration method optimal state-action values for action north

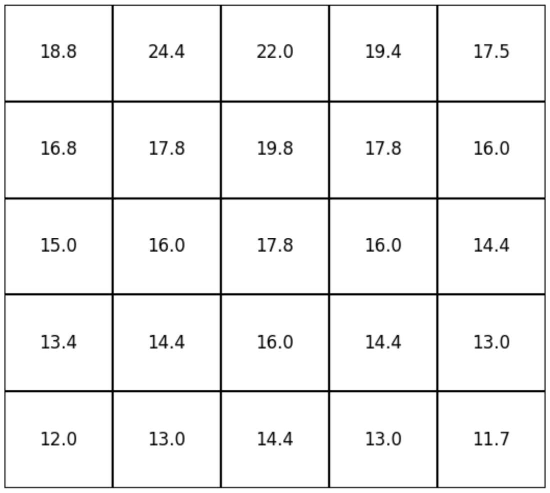
18.8	24.4	18.8	19.4	14.7
19.8	22.0	19.8	17.5	15.7
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7


Policy iteration method optimal state-action values for action south

17.8	24.4	17.8	19.4	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7
13.0	14.4	13.0	11.7	10.5
12.0	13.4	12.0	10.7	9.5

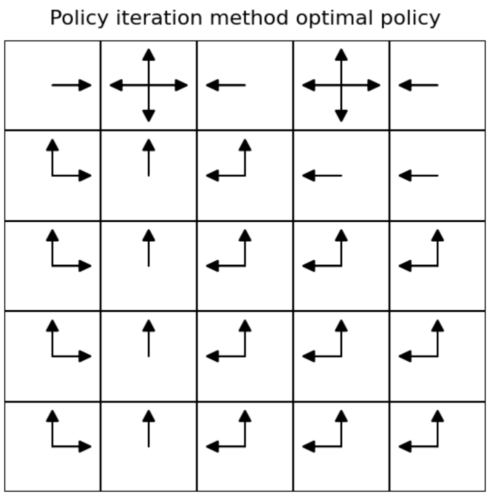

Policy iteration method optimal state-action values for action east

22.0	24.4	17.5	19.4	14.7
19.8	17.8	16.0	14.4	13.4
17.8	16.0	14.4	13.0	12.0
16.0	14.4	13.0	11.7	10.7
14.4	13.0	11.7	10.5	9.5

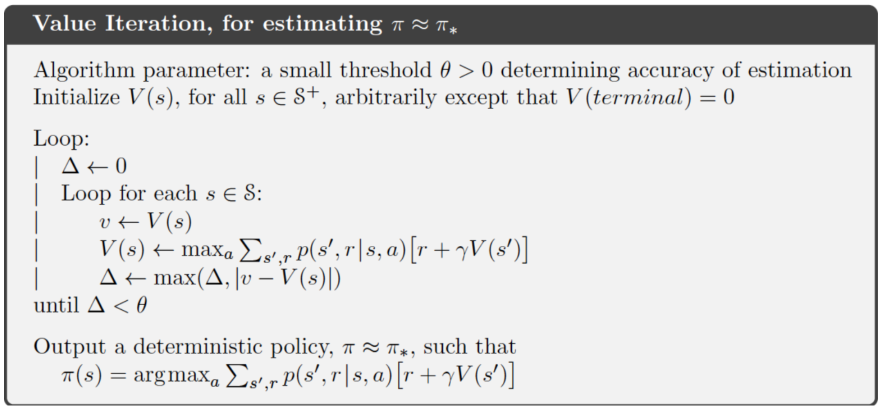

Policy iteration method optimal state-action values for action west

18.8	24.4	22.0	19.4	17.5
16.8	17.8	19.8	17.8	16.0
15.0	16.0	17.8	16.0	14.4
13.4	14.4	16.0	14.4	13.0
12.0	13.0	14.4	13.0	11.7

Problem 3 continued on next page...



3. Value iteration method:
We can applying policy iteration method through the following pseudocode to get $v_*(s)$ and $q_*(s,a)$, then get $\pi_*(s)$ through $q_*(s,a)$:

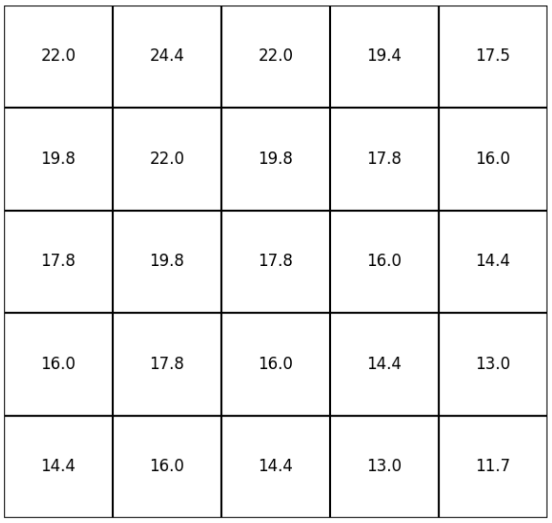


Value Iteration, for estimating $\pi \approx \pi_*$

```
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
|   $\Delta \leftarrow 0$
|   Loop for each $s \in \mathcal{S}$:
|       $v \leftarrow V(s)$
|       $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
|       $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

The results are shown as follows:

Policy iteration method optimal state values



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Policy iteration method optimal
state-action values for action north

18.8	24.4	18.8	19.4	14.7
19.8	22.0	19.8	17.5	15.7
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Policy iteration method optimal
state-action values for action south

17.8	24.4	17.8	19.4	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7
13.0	14.4	13.0	11.7	10.5
12.0	13.4	12.0	10.7	9.5

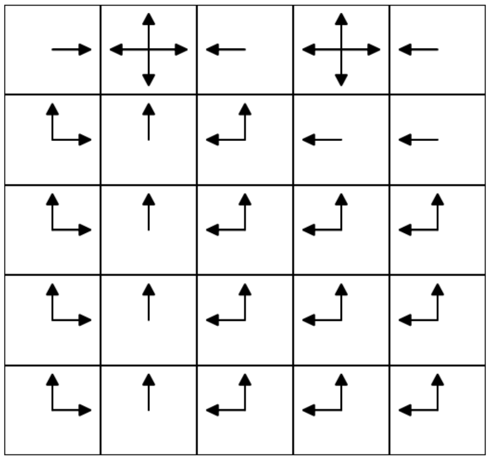
Policy iteration method optimal
state-action values for action east

22.0	24.4	17.5	19.4	14.7
19.8	17.8	16.0	14.4	13.4
17.8	16.0	14.4	13.0	12.0
16.0	14.4	13.0	11.7	10.7
14.4	13.0	11.7	10.5	9.5

Policy iteration method optimal
state-action values for action west

18.8	24.4	22.0	19.4	17.5
16.8	17.8	19.8	17.8	16.0
15.0	16.0	17.8	16.0	14.4
13.4	14.4	16.0	14.4	13.0
12.0	13.0	14.4	13.0	11.7

Policy iteration method optimal policy



Problem 4

Optimal Policy for Simple MDP
Consider the simple n-state MDP shown in Figure 1. Starting from state s_1, the agent can move to the right (a_0) or left (a_1) from any state s_i. Actions are deterministic and always succeed (e.g. going left from state s_2 goes to state s_1, and going left from state s_1 transitions to itself). Rewards are given upon taking an action from the state. Taking any action from the goal state G earns a reward of $r = +1$ and the agent stays in state G. Otherwise, each move has zero reward ($r = 0$). Assume a discount factor $\gamma < 1$.

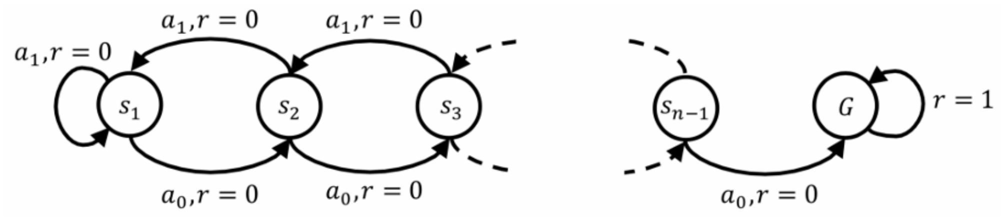


Figure 1: n-state MDP

(a) The optimal action from any state s_i is taking a_0 (right) until the agent reaches the goal state G. Find the optimal value function for all states s_i and the goal state G.
(b) Does the optimal policy depend on the value of the discount factor γ? Explain your answer.
(c) Consider adding a constant c to all rewards (i.e. taking any action from states s_i has reward c and any action from the goal state G has reward $1+c$). Find the new optimal value function for all states s_i and the goal state G. Does adding a constant reward c change the optimal policy? Explain your answer.
(d) After adding a constant c to all rewards now consider scaling all the rewards by a constant a (i.e.$r_{\text{new}} = a\left(c + r_{\text{old}}\right)$). Find the new optimal value function for all states s_i and the goal state G. Does that change the optimal policy? Explain your answer, If yes, give an example of α and c that changes the optimal policy.
Solution
(a) Firstly, consider the state value function of G, since it has only one action:

$$\begin{aligned} v_*(G) &= \max_{a\in\mathcal{A}}\left(\mathcal{R}_G^a + \gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{G,s'}^a v_*(s')\right) \\ &= 1 + \gamma v_*(G) \\ \Rightarrow v_*(G) &= \frac{1}{1-\gamma} \end{aligned}$$

Since the optimal policy is obvious, as it is given above, so we can get the state value function of all states using Bellman Optimality Equation. Define $G = s_n$, then $\forall i = 1, 2, \ldots, n-1$, we have:

$$\begin{aligned} v_*(s_i) &= \max_{a\in\mathcal{A}}\left(\mathcal{R}_{s_i}^a + \gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{s_i,s'}^a v_*(s')\right) \\ &= 0 + \gamma\cdot 1\cdot v_*(s_{i+1}) \\ &= \gamma v_*(s_{i+1}) \end{aligned}$$

So above all, all states' value functions are:

$$\begin{aligned} v_*(G) &= \frac{1}{1-\gamma} \\ v_*(s_i) &= \frac{\gamma^{n-i}}{1-\gamma} \qquad \forall i = 1, 2, \ldots, n-1 \end{aligned}$$

(b) When $\gamma > 0$, the value of γ does not change the relative order of the value functions of states, since

$$q_*(s,a) = \gamma v_*(s')$$

So the optimal policy will not change. i.e. $\forall i = 1, \cdots, n-1, \pi(s_i) = a_0$. However, when $\gamma = 0$, no matter what action(a_0 or a_1) we take, the state-action value $q_*(s,a)$ will always be 0, which are the same, so both a_0 and a_1 are the optimal actions.
(c) For any policy π, suppose the state value function is $v_\pi(s_i)$. If we fixed policy, but only change the reward, then the state value functions becomes $v'_\pi(s_i)$. Similarly, suppose the reward is $r'_t = r_t + c$. Then $\forall i = 1, \cdots, n$, we have:

$$\begin{aligned}
v'_\pi(s_i) &= \mathbb{E}_\pi[G_t|S_t = s_i] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r'_t\gamma^t|S_t = s_i\right] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} (r_t + c)\gamma^t|S_t = s_i\right] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r_t\gamma^t|S_t = s_i\right] + \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} c\gamma^t|S_t = s_i\right] \\
&= v_\pi(s_i) + \frac{c}{1-\gamma}
\end{aligned}$$

Similarly,

$$q'_\pi(s,a) = \gamma v'_\pi(s') = \gamma v_\pi(s') + \frac{c\cdot\gamma}{1-\gamma}$$

Thus, the relative order of value functions of states will not change, and the optimal policy will not changed.
(d) Similarly with (c), but the new reward $r'_t = a(c + r_t)$, the value functions will be:

$$\begin{aligned}
v'_\pi(s_i) &= \mathbb{E}_\pi[G_t|S_t = s_i] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r'_t\gamma^t|S_t = s_i\right] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} a(r_t + c)\gamma^t|S_t = s_i\right] \\
&= a\cdot\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r_t\gamma^t|S_t = s_i\right] + a\cdot\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} c\gamma^t|S_t = s_i\right] \\
&= a\cdot v_\pi(s_i) + \frac{ac}{1-\gamma}
\end{aligned}$$

Similarly,

$$q'_\pi(s,a) = \gamma v'_\pi(s') = a\cdot\left(\gamma v_\pi(s') + \frac{c\cdot\gamma}{1-\gamma}\right)$$

So above all, if $a > 0$, the relative order of $v(s_i)$ will not change, so the optimal policy will not change.
If $a = 0$, then all states will have $q_*(s,a) = 0$, so any policy is the optimal policy.
If $a < 0$, then reaching G would have the worst reward, so the optimal policy is to never reach G.

Problem 5

Running Time of Value Iteration

In this problem we construct an example to bound the number of steps it will take to find the optimal policy using value iteration. Consider the infinite MDP with discount factor $\gamma < 1$ illustrated in Figure 2. It consists of 3 states, and rewards are given upon taking an action from the state. From state s_0, action a_1 has zero immediate reward and causes a deterministic transition to state s_1 where there is reward $+1$ for every time step afterwards (regardless of action). From state s_0, action a_2 causes a deterministic transition to state s_2 with immediate reward of $\frac{\gamma^2}{1-\gamma}$ but state s_2 has zero reward for every time step afterwards (regardless of action).

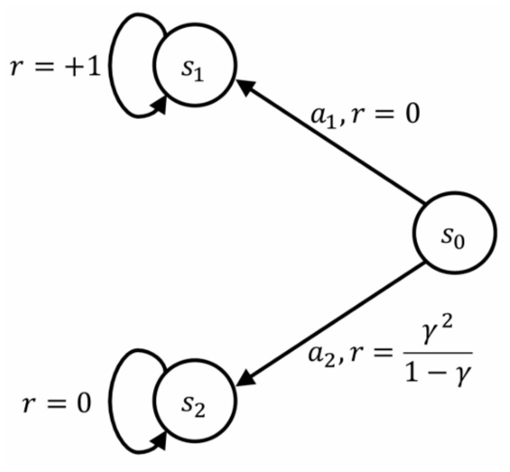


Figure 2: infinite 3-state MDP

(a) What is the total discounted return $\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$ of taking action a_1 from state s_0 at time step $t = 0$?

(b) What is the total discounted return $\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$ of taking action a_2 from state s_0 at time step $t = 0$? What is the optimal action?

(c) Assume we initialize value of each state to zero, (i.e. at iteration $n = 0, \forall s : V_{n=0}(s) = 0$). Show that value iteration continues to choose the sub-optimal action until iteration n^* where,

$$n^* \geq \frac{\log(1-\gamma)}{\log \gamma} \geq \frac{1}{2} \log\left(\frac{1}{1-\gamma}\right) \frac{1}{1-\gamma}$$

Thus, value iteration has a running time that grows faster than $\frac{1}{1-\gamma}$. (You just need to show the first inequality)

Solution

(a) The total discounted return of taking action a_1 is:

$$\sum_{t=0}^{\infty} r_t \gamma^t = 0 + \sum_{t=1}^{\infty} \gamma^t = \frac{\gamma}{1-\gamma}$$

(b) The total discounted return of taking action a_2 is:

$$\sum_{t=0}^{\infty} r_t \gamma^t = \frac{\gamma^2}{1-\gamma} + 0 = \frac{\gamma^2}{1-\gamma}$$

Since $\gamma < 1$, so it must have $\dfrac{\gamma}{1-\gamma} > \dfrac{\gamma^2}{1-\gamma}$ i.e. the optimal action is a_1.

Problem 5 continued on next page...

(c) Since state s_1 and s_2 are the absorbing states, thus directly apply Bellman Expectation Equation, we can get that

$$\begin{aligned} &V_{n+1}(s_1) = 1 + \gamma V_n(s_1) \Rightarrow V_n(s_1) = \frac{1-\gamma^n}{1-\gamma} \\ &V_{n+1}(s_2) = \gamma V_n(s_2) = 0 \end{aligned}$$

As for state s_0, we need to consider its actions, thus we need the state-action values:

$$\begin{aligned} Q_{n+1}(s_0, a_1) &= 0 + \gamma V_n(s_1) = \gamma \frac{1-\gamma^n}{1-\gamma} \\ Q_{n+1}(s_0, a_2) &= \frac{\gamma^2}{1-\gamma} + \gamma V_n(s_2) = \frac{\gamma^2}{1-\gamma} \end{aligned}$$

Thus when $Q_{n+1}(s_0, a_1) \geq Q_{n+1}(s_0, a_2)$, we will selection action a_1. Thus n^* is the minimal iteration number such that $Q_{n+1}(s_0, a_1) \geq Q_{n+1}(s_0, a_2)$. Thus we can get n^* by solving

$$\gamma \frac{1-\gamma^n}{1-\gamma} \geq \frac{\gamma^2}{1-\gamma}$$

Solve the inequality, we have

$$n^* \geq \frac{\log(1-\gamma)}{\log \gamma}$$

So above all, we have proved that the value iteration continues to choose the sub-optimal action until iteration n^* where

$$n^* \geq \frac{\log(1-\gamma)}{\log \gamma}$$

For the second inequality, which is easy to prove. The base for the log function should be 2. Since $0 < \gamma < 1$, thus what we need to prove is that

$$\begin{aligned} &\frac{\log(1-\gamma)}{\log \gamma} \geq \frac{1}{2} \log\left(\frac{1}{1-\gamma}\right) \frac{1}{1-\gamma} \\ &\Rightarrow \log \gamma + 2(1-\gamma) \leq 0 \end{aligned}$$

Let $f(\gamma) = \log \gamma + 2(1-\gamma) \Rightarrow f'(\gamma) = \log \gamma + 2(1-\gamma), f''(\gamma) = -\frac{1}{\gamma^2} < 0$, thus

$$f(\gamma) \leq f\left(\frac{1}{2}\right) = 1 - \log 2 = 0 \qquad \text{If base is 2}$$

So above all, we have proved that

$$n^* \geq \frac{\log(1-\gamma)}{\log \gamma} \geq \frac{1}{2} \log\left(\frac{1}{1-\gamma}\right) \frac{1}{1-\gamma}$$

Problem 6

Approximating the Optimal Value Function
Consider a finite MDP $M = \langle \mathcal{S}, \mathcal{A}, T, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ action space, T transition probabilities, $\mathcal{R}$ reward function and γ the discount factor. Define Q^* to be the optimal state-action value $Q^*(s,a) = Q_{\pi^*}(s,a)$ where π^* is the optimal policy. Assume we have an estimate $\tilde{Q}$ of Q^*, and $\tilde{Q}$ is bounded by L_∞ norm as follows:

$$\|\tilde{Q} - Q^*\|_\infty \le \epsilon$$

Where $\|x\|_\infty = \max\limits_{s,a} |x(s,a)|$.
Assume that we are following the greedy policy with respect to $\tilde{Q}, \pi(s) = \operatorname*{argmax}\limits_{a \in \mathcal{A}} \tilde{Q}(s,a)$. We want to show that the following holds:

$$V_\pi(s) \ge V^*(s) - \frac{2\epsilon}{1-\gamma}$$

Where $V_\pi(s)$ is the value function of the greedy policy π and $V^*(s) = \max\limits_{a\in\mathcal{A}} Q^*(s,a)$ is the optimal value function. This shows that if we compute an approximately optimal state-action value function and then extract the greedy policy for that approximate state-action value function, the resulting policy still does well in the real MDP.
(a) Let π^* be the optimal policy, V^* the optimal value function and as defined above $\pi(s) = \operatorname*{argmax}\limits_{a\in\mathcal{A}} \tilde{Q}(s,a)$. Show the following bound holds for all states $s \in \mathcal{S}$.

$$V^*(s) - Q^*(s, \pi(s)) \le 2\epsilon$$

(b) Using the results of part 1, prove that $V_\pi(s) \ge V^*(s) - \frac{2\epsilon}{1-\gamma}$.
Now we show that this bound is tight. Consider the 2-state MDP illustrated in Figure 3. State s_1 has two actions, "*stay*" self transition with reward 0 and "*go*" that goes to state s_2 with reward 2ϵ. State s_2 transitions to itself with reward 2ϵ for every time step afterwards.

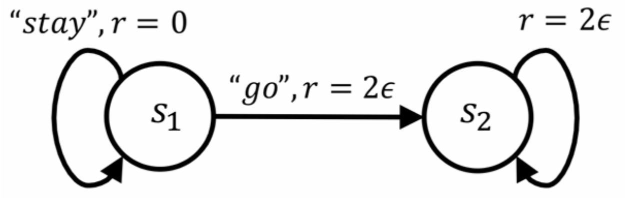


Figure 3: A 2-state MDP.

(c) Compute the optimal value function $V^*(s)$ for each state and the optimal stateaction value function $Q^*(s,a)$ for state s_1 and each action.
(d) Show that there exists an approximate state-action value function $\tilde{Q}$ with ϵ error (measured with L_∞ norm), such that $V_\pi(s_1) - V^*(s_1) = -\frac{2\epsilon}{1-\gamma}$, where $\pi(s) = \operatorname*{argmax}\limits_{a\in\mathcal{A}} \tilde{Q}(s,a)$. (You may need to define a consistent tie break rule)

Solution

(a) Since $\pi(s)$ is the optimal policy for $\tilde{Q}$, thus it has $\tilde{Q}(s,\pi(s)) \ge \tilde{Q}(s,\pi^*(s))$, so

$$\begin{aligned}
V^*(s) - Q^*(s,\pi(s)) &= V^*(s) - \tilde{Q}(s,\pi(s)) + \tilde{Q}(s,\pi(s)) - Q^*(s,\pi(s)) \\
(\text{Since } \|\tilde{Q} - Q^*\|_\infty \le \epsilon) \quad &\le V^*(s) - \tilde{Q}(s,\pi(s)) + \epsilon \\
(\text{Since } \tilde{Q}(s,\pi(s)) \ge \tilde{Q}(s,\pi^*(s))) \quad &\le V^*(s) - \tilde{Q}(s,\pi^*(s)) + \epsilon \\
&= Q^*(s,\pi^*(s)) - \tilde{Q}(s,\pi^*(s)) + \epsilon \\
(\text{Since } \|\tilde{Q} - Q^*\|_\infty \le \epsilon) \quad &\le 2\epsilon
\end{aligned}$$

So above all, we have proved that $V^*(s) - Q^*(s,\pi(s)) \le 2\epsilon$.

Problem 6 continued on next page...

(b) Using Bellman Exceptation Equation, we have

$$\begin{aligned}Q^*(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a V^*(s') \\ \tilde{Q}(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{s,s'}^a V_\pi(s')\end{aligned}$$

Combining them, we can get that

$$\begin{aligned}V^*(s) - V_\pi(s) &= V^*(s) - Q^*(s,\pi(s)) + Q^*(s,\pi(s)) - V_\pi(s) \\ \text{(Conclusion of (a))} \quad &\leq 2\epsilon + Q^*(s,\pi(s)) - Q(s,\pi(s)) \\ &= 2\epsilon + \left(\mathcal{R}_s^{\pi(s)} + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{\pi(s),s'}^a V^*(s')\right) - \left(\mathcal{R}_s^{\pi(s)} + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{\pi(s),s'}^a V_\pi(s')\right) \\ &= 2\epsilon + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{\pi(s),s'}^a \left(V^*(s') - V_\pi(s')\right)\end{aligned}$$

Suppose that the upper bound of $V*(s) - V_\pi(s)$ is U, i.e. $\forall s \in \mathcal{S}, V^*(s) - V_\pi(s) \leq U$, thus we have:

$$\begin{aligned}\forall s \in \mathcal{S}, \quad V^*(s) - V_\pi(s) &\leq 2\epsilon + \gamma U \\ \Rightarrow \quad U &\leq 2\epsilon + \gamma U \\ \Rightarrow \quad (1-\gamma)U &\leq 2\epsilon \\ \Rightarrow \quad U &\leq \frac{2\epsilon}{1-\gamma}\end{aligned}$$

So above all, we have proved:

$$V^*(s) - V_\pi(s) \leq \frac{2\epsilon}{1-\gamma} \Rightarrow V_\pi(s) \geq V^*(s) - \frac{2\epsilon}{1-\gamma}$$

(c) It is obvious that the optimal policy is $\pi_*(s_1) = \text{go}$, from the Bellman Optimility Equation, we have

$$\begin{aligned}V^*(s_2) = 2\epsilon + \gamma V^*(s_2) &\Rightarrow V^*(s_2) = \frac{2\epsilon}{1-\gamma} \\ V^*(s_1) = V^{\pi_*}(s_1) = 2\epsilon + \gamma V^*(s_2) &\Rightarrow V^*(s_1) = \frac{2\epsilon}{1-\gamma} \\ Q^*(s_1,\text{go}) = 2\epsilon + \gamma V^*(s_2) &\Rightarrow Q^*(s_1,\text{go}) = \frac{2\epsilon}{1-\gamma} \\ Q^*(s_1,\text{stay}) = 0 + \gamma V^*(s_1) &\Rightarrow Q^*(s_1,\text{stay}) = \frac{2\epsilon\gamma}{1-\gamma}\end{aligned}$$

(d) Since the approximate state-action value function allowed the ϵ error, and we could observe that the difference between $Q^*(s_1,\text{go})$ and $Q^*(s_1,\text{stay})$ is $\frac{2\epsilon}{1-\gamma} - \frac{2\epsilon\gamma}{1-\gamma} = 2\epsilon$. Thus we can set $\tilde{Q}$ to be

$$\begin{aligned}\tilde{Q}(s_1,\text{go}) = Q^*(s_1,\text{go}) - \epsilon &\Rightarrow \tilde{Q}(s_1,\text{go}) = \frac{\epsilon(1+\gamma)}{1-\gamma} \\ \tilde{Q}(s_1,\text{stay}) = Q^*(s_1,\text{stay}) + \epsilon &\Rightarrow \tilde{Q}(s_1,\text{stay}) = \frac{\epsilon(1+\gamma)}{1-\gamma}\end{aligned}$$

Since the estimated $\tilde{Q}$ for state s_1 and its two actions are tie, so we can define the policy for tie: $\pi(\text{stay}|s_1) = 1$. Thus $V_\pi(s_1) = 0 + \gamma V_\pi(s_1) \Rightarrow V_\pi(s_1) = 0$. Since $V^*(s_1) = \frac{2\epsilon}{1-\gamma}$, thus $V_\pi(s_1) - V^*(s_1) = -\frac{2\epsilon}{1-\gamma}$.
So above all, we have proved that the $V_\pi(s) \geq V^*(s) - \frac{2\epsilon}{1-\gamma}$ is a tight bound.

Problem 7

Bonus Problem: Accepting the Best Offer.
A senior undergraduate student of SIST in ShanghaiTech University have applied many top graduate programs around the world. He is now presented with n offers in a sequential order. After looking at an offer, he must decide whether to accept it (and terminate the process) or to refuse it. Once refuseed, an offer is lost. Suppose that the only information he has at any time is the relative rank of the present offer compared with previously ones. His objective is to maximize the probability of selecting the best offer when all $n!$ orderings of the offers are assumed to be equally likely. Please help help him to find the optimal policy and compute the corresponding probability.

Solution
The optimal solution is the following policy: refuse the first $\left\lfloor \dfrac{n}{e} \right\rfloor$ offers, and mark the best offer among them be '*Best*'. Then looking for the remaining offers sequentially, when meet the first offer that is better than '*Best*', accept it.
Proof:
1. Firstly, we need to show that this method is optimal that refusing the first m offers at first.
According to the settings, it is a normal thought that we would not recieve the i-th offer if it is not the best among the first i offers. So when the i-th offer comes($i = 1, \cdots, n-1$), we have 2 choices: accept or refuse. Let A_i donates the probability that accepting the i-th offer, and we recieved that best offer among all n offers. And R_i donates refuseing the i-th offer, then the maximum probability that we can recieves that best offer. So the probability that we can recieve the best offer is:

$$B_i = \max\{A_i, R_i\}$$

If we accept the i-th offer, which means that it is the best offer among the first i offers. Thus the probability that i-th offer is the best offer among all n offers is:

$$\begin{aligned} A_i &= P(i\text{-th offer is the best offer among all } n \text{ offers} \mid i\text{-th offer is the best offer among the first } i \text{ offers}) \\ &= \frac{\frac{(i-1)!}{i!}}{\frac{(n-1)!}{n!}} \\ &= \frac{i}{n} \end{aligned}$$

And from the definition of R_i: the maximal probability of accepting the best offer when we have refuseed the first i offers, which is obviously deceasing. And since A_i is increasing, thus there must exists a $m \in \{1, 2, \cdots, n-1\}$, s.t.

$$\begin{aligned} A_i \le R_i, &\quad i \le m \\ A_i > R_i, &\quad i > m \end{aligned}$$

Therefore, the optimal policy must be that: refuse the first m offers and then accept the first offer which satisfies it has higher value than any of its predecessors. Thus, we need to find the optimal m.
2. Find the optimal m:
Define B donates the event that getting the best offer, P_m donates using this policy, using LOTP with extra conditioning, we can get that:

$$\begin{aligned} P(B|P_m) &= \sum_{i=1}^{n-m} P(B_m|\text{accept } (i+m)\text{-th offer}, P_m) \cdot P(\text{accept } (i+m)\text{-th offer}|P_m) \\ &= \sum_{i=1}^{n-m} A_{i+m} \cdot P(\text{accept } (i+m)\text{-th offer}|P_m) \end{aligned}$$

Problem 7 continued on next page...

And if we use the policy P_m, and recieves the $(i+m)$-th offer, then it means that the $(i+m)$-th offer is the first offer that is better than the best offer among the first m-th offers. So the best offer among the first $(i+m-1)$-th offers should be in the first m offers, whose probability $\dfrac{m\cdot(i+m-2)!}{(i+m-1)!}=\dfrac{m}{i+m-1}$.
Also, it requires that the $(i+m)$-th offer is the best offer among the first $i+m$ offers, whose probability is $\dfrac{(i+m-1)!}{(i+m)!}=\dfrac{1}{i+m}$. So above all,

$$\begin{aligned}
P(B|P_m) &= \sum_{i=1}^{n-m} A_{i+m}\cdot P(\text{accept } (i+m)\text{-th offer}|P_m)\\
&= \sum_{i=1}^{n-m}\frac{i+m}{n}\cdot\frac{m}{i+m-1}\cdot\frac{1}{i+m}\\
&= \frac{m}{n}\sum_{i=1}^{n-m}\frac{1}{i+m-1}\\
&= \frac{m}{n}\sum_{i=m}^{n-1}\frac{1}{i}\\
(\text{As } n\to\infty)\qquad &\approx \frac{m}{n}\int_m^{n-1}\frac{1}{x}\mathrm{d}x\\
&= \frac{m}{n}\log\left(\frac{n-1}{m}\right)\\
(\text{As } n\to\infty)\qquad &= \frac{m}{n}\log\left(\frac{n}{m}\right)
\end{aligned}$$

Define

$$f(m)=\frac{m}{n}\log\left(\frac{n}{m}\right)$$

Thus

$$\begin{aligned}
f'(m) &= \frac{1}{n}\log\left(\frac{n}{m}\right)-\frac{1}{n}\\
f''(m) &= -\frac{1}{nm}<0\\
f'(m)=0 &\Rightarrow m=\frac{n}{e}
\end{aligned}$$

Since $f(m)$ is concave, so $f(m)_{\max}=f\left(\frac{n}{e}\right)=\frac{1}{e}$. And since m is a non-negative integer, so set $m=\left\lfloor\frac{n}{e}\right\rfloor$.
So, the optimal policy is selecting $m=\left\lfloor\frac{n}{e}\right\rfloor$ as n is large enough.
So above all, we have proved that as n is large enough, the policy we mentioned at the beginning is optimal.
And the probability of selecting the best offer is $\frac{1}{e}$.

Problem 8

Bonus Problem: Bayesian Bandit Process.

There are two arms which may be pulled repeatedly in any order. Each pull may result in either a success or a failure. The sequence of successes and failures which results from pulling arm i ($i = 1$ or 2) forms a Bernoulli process with unknown success probability θ_i. A success at the t^{th} pull yields a reward $\gamma^{t-1}(0 < \gamma < 1)$, while an unsuccessful pull yields a zero reward. At time zero, each θ_i has a beta prior distribution with two parameters α_i, β_i and these distributions are independent for different arms. These prior distributions are updated to successive posterior distributions as arms are pulled. Since the class of beta distributions is closed under Bernoulli sampling, these posterior distributions are all beta distributions. How should the arm to pull next at each stage be chosen so as to maximize the total expected reward from an infinite sequence of pulls?

(a) One intuitive policy suggests that at each stage we should pull the arm for which the current expected value of θ_i is the largest. This policy behaves very good in most cases. However, it is unfortunately not optimal. Please provide an example to show why such policy is not optimal.

(b) For the expected total reward under an optimal policy, show that the following recurrence equation holds:

$$\begin{aligned}
R_1(\alpha_1, \beta_1) &= \frac{\alpha_1}{\alpha_1 + \beta_1}\left[1 + \gamma R(\alpha_1 + 1, \beta_1, \alpha_2, \beta_2)\right] + \frac{\beta_1}{\alpha_1 + \beta_1}\left[\gamma R(\alpha_1, \beta_1 + 1, \alpha_2, \beta_2)\right] \\
R_2(\alpha_2, \beta_2) &= \frac{\alpha_2}{\alpha_2 + \beta_2}\left[1 + \gamma R(\alpha_1, \beta_1, \alpha_2 + 1, \beta_2)\right] + \frac{\beta_2}{\alpha_2 + \beta_2}\left[\gamma R(\alpha_1, \beta_1, \alpha_2, \beta_2 + 1)\right] \\
R(\alpha_1, \beta_1, \alpha_2, \beta_2) &= \max\left\{R_1(\alpha_1, \beta_1), R_2(\alpha_2, \beta_2)\right\}
\end{aligned}$$

(c) Discuss the computational complexity of solving the above equation. How to solve it approximately?

(d) Find the optimal policy. (Hint: Gittins index policy)

Solution

The codes for (a), (d) could be found in the 'hw5_code.ipynb' file.

(a) One situation is that when the 2 arms' probability comes to be relatively close, we have that this algorithm will fail to give an optimal solution. Given an situation where $p_1 = 0.8, p_2 = 0.7$, by simulation, we will found out that the final probability of success is only about 0.8 which is closer to the smaller probability p_2 instead of the larger probability p_1, which is not optimal. The priorior distributions for the two arms are all set to be $\text{Beta}(1, 1)$. The code simulate this situation, in the code, we estimate the expectation reward by giving reward$=\dfrac{p_i}{1-\gamma}$ when the simulation number is larger enough if we stick to one arm. Then by compare the reward from the algorithm, we will determine which case the current simulation falls into.

From the code simulation, we can find out that the among the 1000 times simulation, most cases fall into the non-ideal cases, which is actually: when we choose the arm with probability 0.8 and we make success, then the algothrim tends to give choice that the arm with probability 0.8 is more likely to win as another arm hasn't beed shaked so is probability 0.5 to win. Therefore we actually falls into a case that we stick to one arm with lower win probability 0.8.

So the number of getting the better arm is only 308 times, and the number of getting the worse arm is 692 times. So it chooses much more worse arm than the better arm, which is not optimal. So we could say that we have given a case to show that such policy is not optimal.

(b) At time zero, each arm has a prior $\hat{\theta}_i \sim \text{Beta}(\alpha_i, \beta_i)$ and these distributions are independent for different arms. Suppose that we pull the arm 1 at time zero:

<1> If it successes, we can recieve the reward $\gamma^0 = 1$. And the parameters become $\text{Beta}(\alpha_1 + 1, \beta_1, \alpha_2, \beta_2)$ due to the Beta-Binoimal conjugate. After the discount, the future's reward is $\gamma R(\alpha_1 + 1, \beta_1, \alpha_2, \beta_2)$. Also, since success happens with probability $\hat{\theta}_1$. So the total rewards when success at this time is

$$\hat{\theta}_1[1 + \gamma R(\alpha_1 + 1, \beta_1, \alpha_2, \beta_2)]$$

Problem 8 continued on next page...

<2> If it fails, we can recieve the reward 0. And the parameters become $(\alpha_1, \beta_1 + 1, \alpha_2, \beta_2)$ due to the Beta-Binoimal conjugate. After the discount, the future's reward is $0 + \gamma R(\alpha_1, \beta_1 + 1, \alpha_2, \beta_2)$. Also, since failure happens with probability $1 - \hat{\theta}_1$. So the total rewards when fails at this time is

$$(1-\hat{\theta}_1)[0+\gamma R(\alpha_1,\beta_1+1,\alpha_2,\beta_2)] = (1-\hat{\theta}_1)[\gamma R(\alpha_1,\beta_1+1,\alpha_2,\beta_2)]$$

So combine the two parts, we can get that the total expected rewards when pull the arm 1 is that

$$\begin{aligned} R_1(\alpha_1,\beta_1) &= \mathbb{E}\left[\hat{\theta}_1[1+\gamma R(\alpha_1+1,\beta_1,\alpha_2,\beta_2)] + (1-\hat{\theta}_1)[\gamma R(\alpha_1,\beta_1+1,\alpha_2,\beta_2)]\right] \\ &= \mathbb{E}\left(\hat{\theta}_1\right)[1+\gamma R(\alpha_1+1,\beta_1,\alpha_2,\beta_2)] + \mathbb{E}\left(1-\hat{\theta}_1\right)[\gamma R(\alpha_1,\beta_1+1,\alpha_2,\beta_2)] \\ \left(\hat{\theta}_1 \sim \text{Beta}(\alpha_1,\beta_1)\right) \quad &= \frac{\alpha_1}{\alpha_1+\beta_1}[1+\gamma R(\alpha_1+1,\beta_1,\alpha_2,\beta_2)] + \frac{\beta_1}{\alpha_1+\beta_1}[\gamma R(\alpha_1,\beta_1+1,\alpha_2,\beta_2)] \end{aligned}$$

And we can get the total expected rewards when pull the arm 2 in a exactly same method, and it is:

$$\begin{aligned} R_2(\alpha_2,\beta_2) &= \mathbb{E}\left[\hat{\theta}_2[1+\gamma R(\alpha_1,\beta_1,\alpha_2+1,\beta_2)] + (1-\hat{\theta}_2)[\gamma R(\alpha_1,\beta_1,\alpha_2,\beta_2+1)]\right] \\ \left(\hat{\theta}_2 \sim \text{Beta}(\alpha_2,\beta_2)\right) \quad &= \frac{\alpha_2}{\alpha_2+\beta_2}[1+\gamma R(\alpha_1,\beta_1,\alpha_2+1,\beta_2)] + \frac{\beta_2}{\alpha_2+\beta_2}[\gamma R(\alpha_2,\beta_1,\alpha_2,\beta_2+1)] \end{aligned}$$

And since for each step, we want to maximize the total reward, i.e. choose the arm that could recieves the maximum the expected future total rewards, so we can get that:

$$R(\alpha_1,\beta_1,\alpha_2,\beta_2) = \max\{R_1(\alpha_1,\beta_1), R_2(\alpha_2,\beta_2)\}$$

So above all, the following recurrence equation holds have been proven.

$$\begin{aligned} R_1(\alpha_1,\beta_1) &= \frac{\alpha_1}{\alpha_1+\beta_1}[1+\gamma R(\alpha_1+1,\beta_1,\alpha_2,\beta_2)] + \frac{\beta_1}{\alpha_1+\beta_1}[\gamma R(\alpha_1,\beta_1+1,\alpha_2,\beta_2)] \\ R_2(\alpha_2,\beta_2) &= \frac{\alpha_2}{\alpha_2+\beta_2}[1+\gamma R(\alpha_1,\beta_1,\alpha_2+1,\beta_2)] + \frac{\beta_2}{\alpha_2+\beta_2}[\gamma R(\alpha_1,\beta_1,\alpha_2,\beta_2+1)] \\ R(\alpha_1,\beta_1,\alpha_2,\beta_2) &= \max\{R_1(\alpha_1,\beta_1), R_2(\alpha_2,\beta_2)\} \end{aligned}$$

(c) Since to calculate $R(\alpha_1,\beta_1,\alpha_2,\beta_2)$, we need $R(\alpha_1+1,\beta_1,\alpha_2,\beta_2), R(\alpha_1,\beta_1+1,\alpha_2,\beta_2), R(\alpha_1,\beta_1,\alpha_2+1,\beta_2), R(\alpha_1,\beta_1,\alpha_2,\beta_2+1)$, which could goes infinitely far. Since the discount factor $\gamma \in (0,1)$, which means that we can regard after steps of iteration, the expected future rewards is γ^t, which is significantly small, so the iteration could stop. Suppose that the threshold of the number of iterations is N, i.e. after $t > N$, we stop the iteration.
To save the computational complexity, we can use the memorization technique, which is to store the results of previous calculations in a table. Thus the computational complexity is reduced to $O(N^4)$, and we requires the space complexity to $O(N^4)$.
To solve the approximate, we only need to consider the corner cases, i.e. when $t = N$, which is the iteration before stops. We could regard that there have a sufficient exploration, thus we can directly compare their expectation, i.e. $\frac{\alpha_1}{\alpha_1+\beta_1}$ and $\frac{\alpha_2}{\alpha_2+\beta_2}$.
(d) The optimal policy is that when the pulling turn is more than N times, we choose the arm with the bigger expection.
Otherwise, we can firstly calculate the R_1, R_2 of each arm, then choose the arm with the bigger R_i.
The implementation could be found in 'hw5_code.ipynb'.