3. Finite Markov Decision Processes

Notes

- MDPs are mathematical idealised form of the RL problem. It contains both evaluative feedback and an associative aspect, where choice of action depends on the state/situation.
- MDPs involve sequential decision making, where actions influence immediate and future rewards. MDPs estimate the value $q_*(s,a)$ of action a and state s, or estimate the value $v_*(s)$ given optimal action selections.
- Agent-Environment Interface:
  - The *agent* is the learner/decision maker, the *environment* is what the agent interacts with and is what provides the *rewards* to the agent.
  - In a discrete time step environment, the agent reacts to the environment state $S_t \in \mathcal{S}$ by selecting action $A_t \in \mathcal{A}(s)$. A timestep later, the agent receives a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and finds itself in a new state S_{t+1}.
  - A finite MDP refers to problem where the set of states, actions and rewards are finite. Given that this is a *Markov* decision process, the reward R_t and state S_t both have a well defined probability decision which depends on its preceding action and state. For random variables $s' \in \mathcal{S}$ and $r \in \mathcal{R}$, the probability of them occurring at time t is

    $$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\},$$

    where the dynamic function $p\colon \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,1]$ defines a probability distribution such that

    $$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1 \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s).$$

  - The *Markov property* places a restriction on the state rather than the decision process, wherein the state must include information about all aspects of the past agent-environment interaction.
  - From the four-argument dynamics function p, one can find other values such as the state-transition probability $p(s'|s,a)$, expected reward for state action pair $r(s,a)$ and expected reward for state-action-next-state triple $r(s,a,s')$.
  - It is important to define the agent-environment boundary. A general rule is that anything that cannot be changed arbitrarily by the agent is considered to be outside of the agent and part of the environment. In the case of a robot, the agent is the 'processor' controlling the robot, and the environment includes the robot's motor and physical sensors.

        - This however does not assume that everything about the environment is unknown to an agent. An agent often understands how rewards are computed from the actions and states of the environment, despite the rewards being computed externally by the environment.
    - The MDP framework proposes that any problem of learning goal-directed behaviour can be reduced to three signals passing between the agent and the environment - actions, states and rewards.
- Rewards:
    - Reward hypothesis: all of what we mean by goals and purposes can be thought of as the maximisation of the expected value of the cumulative sum of a received scalar signal (reward).
    - Examples of rewards: -1 for each time step in attempting to solve a maze, +1, 0 and -1 for winning, drawing and losing in a game of checkers
    - Critical to reward what we want to truly accomplish. The reward signal is not the place to impart to the agent prior knowledge about how to achieve what we want to do. For example, in a game of chess, we want to reward the agent when it wins rather than when it achieves sub-goals such as taking out opponents pieces.
- Returns and Episodes:
    - To be more specific, we want to maximise the *expected return* of the problem. In the simplest case, the return G_t can be written as $G_t \doteq R_{t+1} + R_{t+2} + \cdots + R_T$, where T is the final time step.
    - Tasks can be either an *episodic* or *continuing* task, where the former refers to tasks with independent episodes and the latter refers to tasks that goes on infinitely.
    - In an episodic task, the set of all non-terminal states are denoted by $\mathcal{S}$, and the set of all sates including the terminal states are denoted by $\mathcal{S}^+$
    - Discounted return is given as

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \quad 0 \leq \gamma \leq 1$$

      If $\gamma > 0$, this means that rewards in the future have a lesser weight than the more immediate rewards. It can also be written as $G_t \doteq R_{t+1} + \gamma G_{t+1}$ since it is a sum to ∞.
    - When referring to episodic tasks, one would think that it is important to include the ith episode. However, it turns out that there is almost never a need to distinguish between different episodes.
    - An *absorbing state* can be introduced into the sum to infinity of the expected reward, where by all rewards after the absorbing state is 0. This helps to unify the notation for

the expected return for both episodic and continuing tasks. Alternatively, we can write

$$G_t \doteq \sum_{k=0}^{T} \gamma^{k-t-1} R_k, \quad \text{where } T = \infty \text{ OR } \gamma = 1.$$

- Policies and Value Functions:
  - *Policies* are the ways in which an agent acts (select actions), often with the goal to maximise expected returns from the environment. Formally, it is a probabilistic mapping from states to actions, and denoted as $\pi(a|s)$.
  - The state-value function and action value function for policy π are given as

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

  - Both these value functions can be estimated from experience, if an agent follows policy $\pi(a|s)$. As the number of states/actions tend to infinity, the average of actual returns that follows a state s converges to $v_\pi(s)$, and the average of actual returns that follow a state s and action a converges to $q_\pi(s,a)$. These estimation methods are called *Monte Carlo methods*, since they involve averaging over many random samples. However, this can get very cumbersome if the state space is large. Thus, the value functions v_π and q_π are often maintained as parametrised functions.
  - These value functions satisfy recursive relationships, and can be expressed as

$$\begin{aligned} v_\pi &\doteq \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S},\, r \in \mathcal{R}} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad \text{for all } s \in \mathcal{S} \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) \quad \text{for all } s \in \mathcal{S} \end{aligned}$$

    This is known as the *Bellman equation* for v_π (referring to the first equation), which expresses the relationship between the value of a state and its possible successor states.

$$q_\pi \doteq \sum_{s' \in \mathcal{S},\, r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a') \right] \quad \text{for } s \in \mathcal{S}$$

    This is known as the Bellman equation for q_π.
  - One can draw back diagrams to illustrate the relationships that form the basis of the update or backup operations at the heart of RL methods. These operations transfer value information back to a state from its successor states.
- Optimal Policies:

- a 'good' policy is one that maximises the the rewards an agent receives over the long run. In particular, we can state that $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$.
- For any finite MDP problem, there is always at least, if not more, than one policy that is better than or equal to all other policies. This is the *optimal policy*. Optimal policies share the same optimal state-value function and the same optimal action-value function, which are given by $v_*(s) \doteq \max_\pi v_\pi(s)$ and $q_*(s,a) \doteq \max_\pi q_\pi(s,a)$.
- *Bellman optimality equation* expresses states that the value of a state under the optimal policy must equal the expected return for the best action from that state:

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]. \end{aligned}$$

  The Bellman optimality equation for q_* is given by

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_a q_*(S_{t+1}, a' | S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s', r | s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right]. \end{aligned}$$

  In a backup diagram, picking the maximum is indicated by an arc around the node between the branches of the node.
- For a finite MDP, the Bellman optimality equation for v_* has a unique solution. Note that it is also a system of equations, since there is one equation for each state s. If p is known, then in principle one can solve this system of equations.
- An optimal policy can be selected by only assigning a non-zero probability to those actions that fulfils the Bellman optimality equation (essentially being a maximum). This is a one-stop search, and can be considered to be greedy. Given that v_* contains information about the reward consequences of possible future behaviour, being greedy here yields long term optimal actions.
- Having q_* makes choosing the optimal action even easier, since it acts as a cache of the results of the one-step search, providing a locally available long term expected return for all state-action pair. This essentially means that the optimal actions can be selected without having to know about possible successor states and values, thus without the need to know the dynamics of the environment.
- Explicitly solving the Bellman optimality equation solves the reinforcement learning problem, but it is rarely useful, since it is similar to an exhaustive search, looking ahead

at all possibilities, and calculating the expected rewards. This solution assumes three things:

    - the dynamics of the environment are accurately known
    - sufficient computational resource
    - states have the Markov property
  - Some decision making method approximate the Bellman optimality equation, for example by only looking ahead up to a certain depth, or using actual experienced transitions in place of the expected transitions.
- It rarely happens that an agent picks an optimal policy, since finding the optimal policy is often computationally expensive. Even if an agent has complete understanding of the dynamics of the environment, it is usually not possible to solve the Bellman optimality equation.
- Thus, making approximations of the policies, value functions and models is very important in solving RL problems. In practice, there are also a lot of states such that it is not possible to play them in a table.

Exercises

3.1 *Task 1*: Learning to play a gambling game, such as Poker. *States*: the current hand of the agent, the cards dealt to the middle of the table, and the actions of the other players in the game. *Actions*: making bets of a specific value, or folding the hand. *Rewards:* the payout from the bet. *Task 2*: Playing a computer game, such as Mario. States: the individual frames of the game, which includes what is surrounding Mario. *Actions*: Moving in all four directions. *Rewards*: The speed at which a level is completed, and the number of coins obtained from the level. *Task 3*: Keeping a power plant stable. *States*: the sensor readings of the power plant, and the external power demand from the town/city/country. *Actions*: The rate at which raw material is fed into the plant, the air flow, etc. *Rewards*: How closely matched the output is to the power demand of the city.

3.2 For an MDP, the fact that it is a *Markov process* might place limitations on what it can do. There might be instances where the current state does not fully reflect the past actions, or where actions from the distant past might have some effect on the current state of the system. In addition, when the state and action space gets extremely large, it might be difficult for an MDP to properly represented the task at hand .

3.3 For the task of driving, this appears to be a problem that needs to be broken down into a few sub-agents. It is difficult to draw a line between agent and behaviour, since the task of driving can be split into different sub-tasks, such as ensuring passenger safety or reaching the right destination, which all have different rewards associated to them.

3.4 Know that

$$r(s,a,s') = \sum_{r \in \mathcal{R}} r \frac{p(s',r|s,a)}{p(s'|s,a)}$$

s	a	s'	r	p(s', r | s, a)
high	search	high	r_search	\alpha
high	search	low	r_search	1 - \alpha
low	search	high	-3	1 - \beta
low	search	low	r_search	\beta
high	wait	high	r_wait	1
high	wait	low		0
low	wait	high		0
low	wait	low	r_wait	1
low	recharge	high	0	1
low	recharge	low		0

3.5 The state space has to include the terminal states as well, which is given as

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

where $\mathcal{S}$ denotes all non-terminal states and $\mathcal{S}^+$ denotes all states including the terminal state.

3.6 For the episodic task:

$$G_t \doteq \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} = -\gamma^{T-t-1}$$

For the continuing task:

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = -\gamma^{K-1}$$

where K is the number of timesteps to failure, and other subsequent failures. The expected return for the episodic and continuing tasks are identical.

3.7 For the task of a robot running through a maze, there needs to be negative reward at each time step in spends in the maze. The agent is not being penalised for remaining in the maze, thus it is not under any 'time pressure' to find a way out. Thus, a reward of -1 is given for every passing time step the agent spends in the maze. Note that without negative rewards, the expected return will always be either 1 (non-discounted) or γ^{T-t-1} (discounted).

3.8 $G_5 = 0,\ G_4 = 2,\ G_3 = 4,\ G_2 = 8,\ G_1 = 6,\ G_0 = 2.$

3.9 $G_1 = 7 \times 1/(1-\gamma) = 70,\ G_o = 65.$

3.10 This is a geometric series.

3.11 Given $\pi(a|s)$ and $p(s', r|s, a)$, we want to find $r(s)$:

$$r(s) = \mathbb{E}[R_t|S_{t-1} = s] = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{r \in \mathcal{R},\, s' \in \mathcal{S}} r\, p(s', r|s, a)$$

3.12 Equation of $v_\pi(s)$ in terms of $q_\pi(s, a)$ and $\pi(a|s)$ (know that $E(X) = E(X|Y)P(Y) + E(X|Y')P(Y')$):

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

3.13 Equation of $q_\pi(s, a)$ in terms of $v_\pi(s)$ and $p(s', r|s, a)$. This is tricky because p contains information about the next state s' and the associated reward r. What we want here is a sum across both all s' and r.

$$\begin{aligned} q_\pi(s, a) &= \sum_{s' \in \mathcal{S},\, r \in \mathcal{R}} p(s', r|s, a)(\text{expected return from next state being } s' \text{ and next reward } r) \\ &= \sum_{s' \in \mathcal{S},\, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s')) \end{aligned}$$

3.14 We are given $\gamma = 0.9$. The agent is equally likely to pick any of the four directions from the centre, which means $\pi(a|s) = 0.25\ \ \forall a$. Moving into any of the adjacent squares give reward $r = 0$, and we also know that $p(s', r|s, a) = 1\ \forall s', r$. Therefore, $v_\pi = 0.25 \times 0.9(2.3 + 0.4 - 0.4 + 0.7) = 0.675 \approx 0.7$.

3.15 Knowing the equation equation of the (discounted) expected return, the signs of the rewards aren't important, and only the intervals are relevant. Can show this via adding a constant c to the discounted expected return and state-value function:

$$\begin{aligned} G'_t &\doteq \sum_{k=0}^{\infty} \gamma_k (R_{t+k+1} + c) = \sum_{k=0}^{\infty} \gamma_k R_{t+k+1} + \sum_{k=0}^{\infty} \gamma_k c = \sum_{k=0}^{\infty} \gamma_k c R_{t+k+1} + \frac{c}{1-\gamma} \\ v'_\pi &= \mathbb{E}[G'_t|S_t = s] = \mathbb{E}\left[G_t + \frac{c}{1=\gamma}|S_t = s\right] = \mathbb{E}[G_t|S_t = s] + \frac{c}{1+\gamma} \end{aligned}$$

3.16 In a episodic task, a constant will have an effect on the expected return and value function, unlike in the continuing task. Given that the expected return is finite, this means that the additional term will depend on the length of an episode T, such that

$$G_t^* = \sum_{k=0}^{T} \gamma_k R_{t+k+1} + \sum_{k=0}^{T} \gamma_k c = \sum_{k=0}^{T} \gamma_k R_{t+k+1} + \frac{c(1-\gamma^T)}{1-\gamma}$$

This means that the longer the episode, the larger the expected return. This could potentially mean that the agent is more likely to spend a longer time in the problem since this gives a larger return, assuming that the constant $c > 0$. The agent will be essentially penalised less, if not at all, by spending a longer time on the problem.

3.17 Bellman equation for q_π, in terms of $q_\pi(s, a)$ and $q_\pi(s', a')$:

$$\begin{aligned} q_\pi &= \sum_{s' \in \mathcal{S},\, r \in \mathcal{R}} p(s', r | s, a)(r + \gamma v_\pi(s')) \quad \text{for } s \in \mathcal{S} \\ &= \sum_{s' \in \mathcal{S},\, r \in \mathcal{R}} p(s', r | s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}} \pi(a' | s') q_\pi(s', a') \right] \quad \text{for } s \in \mathcal{S} \end{aligned}$$

3.18 Give $v_\pi(s)$ in terms of $q_\pi(s, a)$:

$$v_\pi(s) = \mathbb{E}[q_\pi(s, a) | S_t = s] = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

3.19 Give $q_\pi(s, a)$ in terms of expected next reward R_{t+1} and expected next state value $v_\pi(S_{t+1})$:

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

3.20 Given that there is a penalty of -1 for each stroke, and that the value of a state of the negative of the number of strokes to the hole from that location.The optimal state-value function will involve a mix of using the driver and the putter, where the driver is used first, followed by the putter when the ball is on the green. Thus, the optimal state-value function will look a lot like the diagram for $q_*(s, \texttt{driver})$ outside of the green, whilst having a value of -1 across the entire green (since we can use a putter in the green).

3.21 The optimal action value function for putting $q_*(s, \texttt{putter})$ involves putting as the first action, and followed by the optimal actions after the first. Thus, within the green, the function will be -1. Outside the green, the function will look similar to the diagram for $v_{\texttt{putt}}$ for the regions of -2 and -3, but including the sand pit in the region of -3. The region of -4 in the same diagram will be replaced by -3, and the regions of -5 and -6 will be replaced by -4, since we can use a driver after making our first putt.

3.22 Here, there are three states, but both deterministic policies can only access two out of the three states. Let's label the states as **`top`**, **`left`** and **`right`**.

For the policy of $\pi_{\texttt{left}}$, it can access the **top** and **left** states. The state-value function $v_{\pi_{\texttt{left}}}$ is given by

$$v_{\pi_{\texttt{left}}} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = \sum_{j=0}^{\infty} [\gamma^{2j}(+1) + \gamma^{2j+1}(0)] = \sum_{j=0}^{\infty} \gamma^{2j} = \frac{1}{1-\gamma^2}$$

For the policy of $\pi_{\texttt{right}}$, it can access the **top** and **right** states. The state-value function $v_{\pi_{\texttt{right}}}$ is given by

$$v_{\pi_{\texttt{right}}} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = \sum_{j=0}^{\infty} [\gamma^{2j}(0) + \gamma^{2j+1}(+2)] = \sum_{j=0}^{\infty} 2\gamma^{2j+1} = \frac{2\gamma}{1-\gamma^2}$$

For $\gamma = 0$, $\pi_{\texttt{left}} = 1$ and $\pi_{\texttt{right}} = 2$, thus $\pi_{\texttt{right}}$ is optimal. For $\gamma = 0.9$, $\pi_{\texttt{left}} = 5.26$ and $\pi_{\texttt{right}} = 9.47$, thus $\pi_{\texttt{right}}$ is optimal. For $\gamma = 0.5$, $\pi_{\texttt{left}} = 1.33$ and $\pi_{\texttt{right}} = 1.33$, thus both policies are equally optimal.

3.23 The Bellman equation for q_* is given by

$$q_* = \sum_{s',r} p(s', r|s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]$$

Recall that there are two states **s** and **h**, three actions **s**, **w** and **r**. Let's start with the state **h**, where the available actions are **s** and **w**:

$$\begin{aligned} q_*(\mathtt{h}, \mathtt{s}) &= p(\mathtt{h}, r_{\mathtt{s}}|\mathtt{h}, \mathtt{s})[r_{\mathtt{s}} + \gamma \max_{a'} q_*(\mathtt{h}, a')] + p(\mathtt{l}, r_{\mathtt{s}}|\mathtt{h}, \mathtt{s})[r_{\mathtt{s}} + \gamma \max_{a'} q_*(\mathtt{l}, a')] \\ &= \alpha[r_{\mathtt{s}} + \gamma \max\{q_*(\mathtt{h}, \mathtt{s}), q_*(\mathtt{h}, \mathtt{w})\}] \\ &\quad + (1-\alpha)[r_{\mathtt{s}} + \gamma \max\{q_*(\mathtt{l}, \mathtt{s}), q_*(\mathtt{l}, \mathtt{w}), q_*(\mathtt{l}, \mathtt{r})\}] \end{aligned}$$

$$\begin{aligned} q_*(\mathtt{h}, \mathtt{w}) &= p(\mathtt{h}, r_{\mathtt{w}}|\mathtt{h}, \mathtt{w})[r_{\mathtt{w}} + \gamma \max_{a'} q_*(\mathtt{h}, a')] + p(\mathtt{l}, 0|\mathtt{h}, \mathtt{w})[0 + \gamma \max_{a'} q_*(\mathtt{l}, a')] \\ &= r_{\mathtt{w}} + \gamma \max\{q_*(\mathtt{h}, \mathtt{s}), q_*(\mathtt{h}, \mathtt{w})\} \end{aligned}$$

State **l** has actions **s**, **w** and **r**:

$$
\begin{aligned}
q_*(\mathtt{l},\mathtt{s}) &= p(\mathtt{h},-3|\mathtt{l},\mathtt{s})[-3+\gamma \max_{a'} q_*(\mathtt{h},a')] + p(\mathtt{l},r_\mathtt{s}|\mathtt{l},\mathtt{s})[r_\mathtt{s}+\gamma \max_{a'} q_*(\mathtt{l},a')] \\
&= (1-\beta)[-3+\gamma \max\{q_*(\mathtt{h},\mathtt{s}), q_*(\mathtt{h},\mathtt{w})\}] \\
&\quad + \beta[r_\mathtt{s}+\gamma \max\{q_*(\mathtt{l},\mathtt{s}), q_*(\mathtt{l},\mathtt{w}), q_*(\mathtt{l},\mathtt{r})\}]
\end{aligned}
$$

$$
\begin{aligned}
q_*(\mathtt{l},\mathtt{w}) &= p(\mathtt{h},0|\mathtt{l},\mathtt{w})[0+\gamma \max_{a'} q_*(\mathtt{h},a')] + p(\mathtt{l},r_\mathtt{w}|\mathtt{l},\mathtt{w})[r_\mathtt{w}+\gamma \max_{a'} q_*(\mathtt{l},a')] \\
&= r_\mathtt{w} + \gamma \max\{q_*(\mathtt{l},\mathtt{s}), q_*(\mathtt{l},\mathtt{w}), q_*(\mathtt{l},\mathtt{r})\}
\end{aligned}
$$

$$
\begin{aligned}
q_*(\mathtt{l},\mathtt{r}) &= p(\mathtt{h},0|\mathtt{l},\mathtt{r})[0+\gamma \max_{a'} q_*(\mathtt{h},a')] + p(\mathtt{l},0|\mathtt{l},\mathtt{r})[0+\gamma \max_{a'} q_*(\mathtt{l},a')] \\
&= \gamma \max\{q_*(\mathtt{h},\mathtt{s}), q_*(\mathtt{h},\mathtt{w})\}
\end{aligned}
$$

3.24 Rules of Gridworld: all steps within the grid have zero reward, stepping out of the grid has a reward of -1, and A → A' and B → B' have rewards of +10 and +5 respectively. There are two candidates for the optimal policy here - moving from A' → A upwards followed by the jump from A → A', and moving upwards from B' → B followed by the B → B' jump. Let's label them π_A and π_B respectively, and calculate the expected return for them:

$$
\begin{aligned}
G_{\pi_A} &= 10 \times \gamma^0 + 0 \times \gamma + 0 \times \gamma^2 + 0 \times \gamma^3 + 0 \times \gamma^4 + 10 \times \gamma^5 + \dots \\
&= 10 \sum_{k=0}^{\infty} \gamma^{5k} = \frac{10}{1-\gamma^5} \\
G_{\pi_B} &= 5 \times \gamma^0 + 0 \times \gamma + 0 \times \gamma^2 + 10 \times \gamma^3 + \dots \\
&= 5 \sum_{k=0}^{\infty} \gamma^{3k} = \frac{5}{1-\gamma^3}
\end{aligned}
$$

Can see that G_{π_A} is always larger than G_{π_B} for $0 \leq \gamma < 1$.

3.25

$$
v_*(s) = \max_{a \in \mathcal{A}} q_*(s,a)
$$

3.26

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

3.27

$$
\pi_*(a|s) = \max_{a \in \mathcal{A}} q_*(s,a)
$$

3.28

$$\pi_*(a|s) = \max_{a \in \mathcal{A}} \left[\sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \right]$$

3.29 The three-argument p and two-argument r are given given as

$$p(s'|s,a) = \sum_{r \in \mathcal{R}} p(s', r|s, a) \quad \text{and} \quad r(s,a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a).$$

The four Bellman equations for v_π, v_*, q_π and q_* in terms of $p(s'|s,a)$ and $r(s,a)$ are:

$$\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \\
&= \sum_a \pi(a|s) \left[r(s,a) + \gamma \sum_s v_\pi(s) p(s'|s,a) \right] \\
v_*(s) &= \max_a \left[r(s,a) + \gamma \sum_s v_\pi(s) p(s'|s,a) \right] \\
q_\pi(s,a) &= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a') \right] \\
&= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') q_\pi(s', a') \\
q_*(s,a) &= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a') \right] \\
&= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} [\pi(a'|s') q_\pi(s', a')]
\end{aligned}$$