Definition (Contraction Mapping). Let $T : M \to N$ be a mapping. T is a contraction if there exists a constant $\gamma \in [0, 1)$ such that for all $x, y \in M$, the following holds:

$$\|T(x) - T(y)\|_M \leq \gamma \|x - y\|_N$$

where $\|\cdot\|_M$ and $\|\cdot\|_N$ denote proper norms on M and N, respectively.

Theorem 1.4. The Bellman operator $\mathcal{B}$ is a contraction mapping for value function V w.r.t. the ∞-norm.

Proof. Let $|\cdot|$ denote the absolute value and $\|\cdot\|$ the infinity norm. $\forall s \in \mathcal{S}$ and value functions V_1, V_2:

$$\begin{aligned}
|\mathcal{B}V_1(s) - \mathcal{B}V_2(s)| &= \left| \left(\max_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_1(s') \right] \right) - \left(\max_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_2(s') \right] \right) \right| \\
&\leq \max_a \left| \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_1(s') \right] - \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V_2(s') \right] \right| \\
&= \gamma \left| \max_a \sum_{s' \in \mathcal{S}} p(s'|s,a) |V_1(s') - V_2(s')| \right| \\
&\leq \gamma \max_{s'} |V_1(s') - V_2(s')| \cdot \max_a \sum_{s' \in \mathcal{S}} p(s'|s,a) \\
&= \gamma \max_{s'} |V_1(s') - V_2(s')| \cdot 1 \\
&= \gamma \max_{s'} |V_1(s') - V_2(s')| \\
&= \gamma \|V_1 - V_2\|
\end{aligned}$$

Therefore, with $0 < \gamma < 1$, we have:

$$\|\mathcal{B}V_1 - \mathcal{B}V_2\| \leq \gamma \|V_1 - V_2\|$$

which proves that $\mathcal{B}$ is a contraction mapping. □

1.3 Model-Free Value Methods

In previous sections we introduced MRP and MDP. The key idea of MRP is to understand the value of states in terms of the expected cumulative rewards the agent can achieve starting from those states. MDP builds on this idea by introducing actions and characterizing the state-action transitions and rewards to determine the optimal policy.

However in previous discussions, we assume that we have access to the dynamics P and reward function R. (Recall that we need to know dynamics in order to do eigher value or policy iteration.)

In this section, let's consider a more general yet more difficult case: we don't have access to the dynamics P and reward function R. We try to figure these out by estimation methods.

1.3.1 Monte Carlo (MC) Policy Evaluation

Monte Carlo policy evaluation is a rather straightforward estimation method that rely merely on experience gained through interaction with the environment.

In Monte-Carlo Policy Evaluation, an agent interacts with the environment, following a specific policy, and collects a sequence of state-action-reward trajectories. The value of a state is then estimated by averaging the returns obtained from all visits to that state. This process is repeated over multiple episodes, and the state values are updated after each episode.

The algorithm is also quite simple:

Algorithm 5 Monte-Carlo evaluation

Require: Initialize $V_0(s) = 0, \forall s \in \mathcal{S}$
1: **repeat**
2: **for** $s \in \mathcal{S}$ **do**
3: $N(S_t) \leftarrow N(S_t) + 1$
4: $V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$
5: **until** Convergence

MC policy evaluation is good because it's super simple, easy to implement, and it learns only from experience. However, it also has some limitations: every single episodes must terminate in the sampling process; it requires a large number of samples to converge, and it does not exploit the Markov property through the Bellman equation.

1.3.2 Temporal Difference (TD) Evaluation

In temporal difference learning, an agent interacts with the environment and receives immediate rewards after each step. It updates the value estimates of states based on the difference between the current estimate and the estimated value of the subsequent state. The iteration equation should be:

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \tag{1.3.1}$$

This is intuitively logical because for a true $V(S_t)$, it should satisfy the Bellman equation $V(S_t) = R_{t+1} + \gamma V(S_{t+1})$. Therefore, in TD evaluation, after a single step, we update $V(S_t)$ with a learning rate α, gradually pulling it towards satisfying the Bellman equation.

Specifically, we have the following definitions:

- $R_{t+1} + \gamma V(S_{t+1})$ is called **TD target**
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called **TD error**

Bias and Variance

Now we would revisit the idea of MC estimation and TD estimation from the view of bias. Recall the definition of *return* and state-value function.

$$G_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{H-1} r_{t+H-1} \tag{1.3.2}$$

$$V(s) = \mathbb{E}[G_t | s_t = s] = \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k r_{t+k} | s_t = s\right] \tag{1.3.3}$$

Then let's draw a comparison between MC estimation, TD estimation, and policy evaluation:

$$\begin{aligned}
\text{policy evaluation:} \quad & V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})] \\
\text{MC estimation:} \quad & V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t)) \\
\text{TD estimation:} \quad & V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))
\end{aligned}$$

In MC estimation, the information source used to update $V(S_t)$ is G_t, which refers to the sampled return of the entire episode.

While in TD estimation, the information source used to update $V(S_t)$ is TD error, which is the information obtained after one-step forward sampling. This information contains the information of the entire episode since $V(S_{t+1})$ itself encompasses the expected reward of the entire trajectory. However, during the sampling process, we only rely on the next timestep's information (R_{t+1} and $V(S_{t+1})$).

In other word, return $G_t = r_t + \gamma r_{t+1} + \cdots + \gamma^{H-1} r_{t+H-1}$ is unbiased estimate of $v_\pi(s_t)$; true TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is also unbiased estimate of $v_\pi(s_t)$; but the TD target $R_{t+1} + \gamma v(S_{t+1})$ is biased estimate of $v_\pi(s_t)$.

It also holds true that TD has lower variance than MC. This is because TD estimation updates its value estimates by using a bootstrapping approach that incorporates information from subsequent time steps, whereas MC estimation relies solely on the sampled return of the entire episode. This dependence on the full episode for each update leads to higher variance.

To sum up, MC has high variance, zero bias, while TD has low variance and some bias.

N-Step Return and Lamda-Return

We could also find something in the middle between TD (which only samples one step further) and MC (which samples the episode till the end), which is called n-step return.

$$\begin{aligned} n=1 \quad & G_t^{(1)} = R_{t+1} + \gamma V\left(S_{t+1}\right) \\ n=2 \quad & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V\left(S_{t+2}\right) \\ & \vdots \\ n=\infty \quad & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T \end{aligned} \tag{1.3.4}$$

Similarly we have the definition of n-step return and n-step TD:

$$\begin{aligned} \text{n-step return} \quad & G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V\left(S_{t+n}\right) \\ \text{n-step TD} \quad & V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha\left(G_t^{(n)} - V\left(S_t\right)\right) \end{aligned} \tag{1.3.5}$$

N-step return partially find a balance between TD and MC, but it still cannot provide a reasonable trade-off of variance and bias: if the initialization is too bad, then there is too much bias so I might want to do MC; while if value is already somewhat good, I'd prefer TD because it has less variance.

One possible solution is λ-return. The λ-return G_t^λ combines all n -step returns $G_t^{(n)}$ using weight $(1-\lambda)\lambda^{n-1}$

$$G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)} \tag{1.3.6}$$

And the corresponding time difference evaluation method is called forward-view TD(λ).

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha\left(G_t^\lambda - V\left(S_t\right)\right) \tag{1.3.7}$$

This is reasonable because the sampling data from farther future time steps is given less weight, while we are more concerned with the sampling data from the nearer time steps.

Eligibility Trace

Actually there is something wrong with $TD(\lambda)$. We introduce the idea of time difference evaluation because we want to update with incomplete sequences without sampling the entire episode. But as λ-return G_t^λ combines all n -step returns $G_t^{(n)}$ using weight $(1-\lambda)\lambda^{n-1}$, we are dragged back to offline policy with episodic updates. To solve this problem we present a contrary view called *backward-view* $TD(\lambda)$

Note that there are two things that we want to take into account while making a policy:

- **frequency heuristics:** it prioritizes actions that have been chosen frequently in the past. This strategy assumes that actions that have been successful in the past are more likely to lead to positive outcomes in the future.
- **recency heuristics:** it prioritizes actions that have been successful recently. This strategy assumes that recent experiences are more indicative of the current state of the environment and that the agent should focus on exploiting recent successful actions to maximize its reward.

Based on this, we have eligibility traces which combine both heuristics:

$$\begin{aligned} & E_0(s) = 0 \\ & E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}\left(S_t = s\right) \end{aligned} \tag{1.3.8}$$

$\mathbf{1}\left(S_t = s\right)$ here denotes an indicator which is set to 1 while $(S_t = s)$ and 0 otherwise.

It tells us that every past state has an influence on the current TD error by:

$$\delta_t = R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right) \tag{1.3.9}$$

and therefore we can propagate the TD error now to all the past states (namely the backward-view) by:

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s) \tag{1.3.10}$$

Next, we will show that in offline update, forward view and backward view are actually two different interpretation of the same mathematical result: Consider this case: state s is only visited at time k, then the eligibility trace would be:

$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}\left(S_t = s\right) = \begin{cases} 0 & \text{if } t < k \\ (\gamma\lambda)^{t-k} & \text{if } t \geq k \end{cases} \tag{1.3.11}$$

Consequently for the entire episode, we have:

$$\sum_{t=1}^{T} \alpha \delta_t E_t(s) = \alpha \sum_{t=k}^{T} (\gamma\lambda)^{t-k} \delta_t = \alpha \left(G_k^\lambda - V\left(S_k\right)\right) \tag{1.3.12}$$

For general λ, TD errors also telescope to λ-error:

$$\begin{aligned} G_t^\lambda - V\left(S_t\right) &= -V\left(S_t\right) + (1-\lambda)\lambda^0 \left(R_{t+1} + \gamma V\left(S_{t+1}\right)\right) \\ &\quad + (1-\lambda)\lambda^1 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 V\left(S_{t+2}\right)\right) \\ &\quad + (1-\lambda)\lambda^2 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V\left(S_{t+3}\right)\right) \\ &\quad + \dots \\ &= -V\left(S_t\right)(\gamma\lambda)^0 \left(R_{t+1} + \gamma V\left(S_{t+1}\right) - \gamma\lambda V\left(S_{t+1}\right)\right) \\ &\quad + (\gamma\lambda)^1 \left(R_{t+2} + \gamma V\left(S_{t+2}\right) - \gamma\lambda V\left(S_{t+2}\right)\right) \\ &\quad + (\gamma\lambda)^2 \left(R_{t+3} + \gamma V\left(S_{t+3}\right) - \gamma\lambda V\left(S_{t+3}\right)\right) \\ &\quad + \dots \\ &\quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right)\right) \\ &\quad + (\gamma\lambda)^1 \left(R_{t+2} + \gamma V\left(S_{t+2}\right) - V\left(S_{t+1}\right)\right) \\ &\quad + (\gamma\lambda)^2 \left(R_{t+3} + \gamma V\left(S_{t+3}\right) - V\left(S_{t+2}\right)\right) \\ &\quad + \dots \\ &= \dots \\ &= \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2 \delta_{t+2} + \dots \end{aligned} \tag{1.3.13}$$

This shows that after one whole episode, forward and backward would give the same results.

1.3.3 Q-Learning: Off-Policy TD Learning

In MC and TD methods, we solve the problem of not knowing value function by estimation based on experiences. However, in policy improvement:$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T\left(s,a,s'\right)\left[R\left(s,a,s'\right) + \gamma V^{\pi_i}\left(s'\right)\right]$ we still need the transition model to decide on the actions. Q learning solve this problem by estimating the Q-function instead of value function, which directly take action into consideration.

Q-learning is a foundational method for reinforcement learning. It is a TD method that estimates the future reward $V(s')$ using the *Q-function* itself, assuming that from state s', the *best* action (according to the Q function) will be executed at each state.

Need to explain more about Q learning and sarsa (why we should set value function to be the argmax of Q function, why iterating over the error term would help improve the policy, the highlevel idea of SARSA and Q-learning).

Therefore, we have the Bellman equation expressed with Q function:

$$\begin{aligned} Q(s,a) &= R(s) + \gamma \sum_{s'} T\left(s,a,s'\right) V\left(s'\right) \\ &= R(s) + \gamma \sum_{s'} T\left(s,a,s'\right) \max_{a'} Q\left(s',a'\right) \end{aligned} \tag{1.3.14}$$

Algorithm 6 Q-learning

Require: Initialize Q function (e.g. $Q(s,a) = 0, \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$).

1: **repeat**
2: Take action $a$according to some exploitation/exploration policy given state s
3: Observe reward r and new state s'
4: Update $Q(s,a)$ using the Bellman equation:

$$Q'(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

5: **until** Convergence

Note that we estimate the future value using $\max_a Q(s',a)$, which means it ignores the actual next action that will be executed, and updates based on the estimated best action. This is known as *off-policy* learning.

Definition (On-Policy and Off-Policy). If an RL algorithm does not depend on the actual next action(s), it is *off-policy*. Otherwise, it is *on-policy*.

Theorem 1.5 (Convergence of Q-Learning). Q-learning converges to the optimal Q-value function in the limit with probability 1, if every state-action pair is visited infinitely often (Jaakkola et al., 1993).

Proof.

Proof of convergence of Q-learning

□

1.3.4 SARSA: On-Policy TD Learning

SARSA (State-action-reward-state-action) is an on-policy reinforcement learning algorithm. It is very simil

ar to Q-learning, except that in its update rule, it actually selects the next action that it will execute, and updates accordingly. Taking this approach is known as *on-policy* reinforcement learning.

Algorithm 7 SARSA

Require: Initialize Q function (e.g. $Q(s,a) = 0, \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$).

1: **repeat**
2: Take action a_n according to some exploitation/exploration policy given state s_n
3: Observe reward $r(s_n, a_n)$ and next state s_{n+1}
4: Select action a_{n+1}according to some policy and update Q_{n+1} using:

$$Q'(s_n,a_n) = Q(s_n,a_n) + \alpha\left[r(s_n,a_n) + \gamma Q(s_{n+1},a_{n+1}) - Q(s_n,a_n)\right]$$

5: **until** Convergence

1.4 Exploration strategies

In previous discussions we always use the greedy policy, where we take

$$\pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi\left(\mathbf{s}_t, \mathbf{a}_t\right) \\ 0 \text{ otherwise} \end{cases} \tag{1.4.1}$$

However, while learning is progressing, using the greedy policy might not be the best solution. Part of why we might not want to use greedy policy in step one of online Q iteration is that this argmax policy is deterministic. And if our intial Q function is quite bad, we might be stuck in taking a bad action, failing to discover any other probably better actions. In practice, we introduce randomness to avoid this situation. One common choice is called ϵ **- greedy**:

$$\pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right) = \begin{cases} 1-\epsilon \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi\left(\mathbf{s}_t, \mathbf{a}_t\right) \\ \epsilon/(|\mathcal{A}|-1) \text{ otherwise} \end{cases} \tag{1.4.2}$$

It's a very simple exploration rule and very useful in practice. Another way of doing this is matching the probability of taking an action with the goodness of the value obtained by calculating that action. This is what **Boltzmann exploration** (also called the softmax rule) do:

$$\pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \propto \exp\left(Q_\phi\left(\mathbf{s}_t, \mathbf{a}_t\right)\right) \tag{1.4.3}$$

Boltzmann exploration is to some extent smarter than ϵ - greedy, since if two action are almost as good, in ϵ - greedy you hardly have chance to explore the second best one, while in Boltzmann exploration they share similar probability of being explored. Also in Boltzmann exploration you don't waste time on actions that are pretty bad.

Note that exploration in reinforcement learning is still an active research area. And we will touch a little bit deeper later.

1.5 Exercises

1.5.1 Reward Choices

This problem is adapted from Assignment 1 of CS234 (2023 Spring).

Consider a tabular MDP with $0 < \gamma < 1$ and no terminal states. The agent will act forever. The original optimal value function for this problem is V_1^* and the optimal policy is π_1^*.

(a) Now someone decides to add a small reward bonus c to all transitions in the MDP. This results in a new reward function $\hat{r}(s,a) = r(s,a) + c; \forall s, a$ where $r(s,a)$ is the original reward function. What is an expression for the new optimal value function? Can the optimal policy in this new setting change? Why or why not?

(b) Instead of adding a small reward bonus, someone decides to multiply all rewards by an arbitrary constant $c \in \mathbb{R}$. This results in a new reward function $\hat{r}(s,a) = c \times r(s,a); \forall s, a$ where $r(s,a)$ is the original reward function. Are there cases in which the new optimal policy is still π_1^* and the resulting value function can be expressed as a function of c and V_1^* (if yes, give the resulting expression and explain for what value(s) of c and why? If not, explain why not)? Are there cases where the optimal policy would change (if yes, provide a value of c and a description of why the optimal policy would change. If not, explain why not)? Is there a choice of c such that all policies are optimal?

(c) If the MDP instead has terminal states that can end an episode, does that change your answer to part (a)? If yes, provide an example MDP where your answers to part (a) and this part would differ.

1.5.2 Markov Decision Process

This problem is adapted from Problem Session 1 of CS234 (2023 Spring).

For all the three questions below, define $V^\pi(s)$ as the expected discounted reward starting from state s and following policy π.

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\left(s_t, a_t\right) \mid s_0 = s, \pi\right]$$

Q1

For an arbitrary $Z \in \mathbb{N}$, consider learning with $Z+1$ distinct discount factors $\gamma_0, \gamma_1, \ldots, \gamma_Z$ where the final discount factor matches that of the MDP $\mathcal{M}, \gamma_Z = \gamma$. Letting $[Z] \triangleq \{1, 2, \ldots, Z\}$ denote the index set, we define the following functions for any policy π :

$$V_{\gamma_z}^\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_z^t \mathcal{R}\left(s_t, a_t\right) \mid s_0 = s, \pi\right] \quad W_z^\pi = V_{\gamma_z}^\pi - V_{\gamma_{z-1}}^\pi, \quad \forall z \in [Z]$$

where $W_0 = V_{\gamma_0}^\pi$.

(a) For any $z \in [Z]$; any policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$; and any $s \in \mathcal{S}$, write an expression for $V_{\gamma_z}^\pi(s)$ exclusively in terms of $\{W_0^\pi, W_1^\pi, \ldots, W_Z^\pi\}$.

(b) Show that W_z^π obeys the following Bellman equation for any $z \in [Z]$ and $s \in \mathcal{S}$:

$$W_z^\pi(s) = \mathbb{E}_{\substack{a\sim\pi(\cdot|s)\\ s'\sim\mathcal{T}(\cdot|s,a)}}\left[(\gamma_z - \gamma_{z-1})\, V_{\gamma_{z-1}}^\pi(s') + \gamma_z W_z(s')\right]$$

Q2

Let $\gamma, \beta \in [0,1)$ be two discount factors such that $\beta \leq \gamma$. Let $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ be an arbitrary policy that induces value functions V_γ^π and V_β^π under the two discount factors, respectively. Similarly, define the Bellman operators

$$\mathcal{B}_\gamma^\pi V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\mathcal{R}(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}\left[V(s')\right]\right]$$
$$\mathcal{B}_\beta^\pi V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\mathcal{R}(s,a) + \beta \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}\left[V(s')\right]\right].$$

With the reward upper bound $R_{\text{MAX}} = \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathcal{R}(s,a)$, prove that

$$\left\|V_\gamma^\pi - V_\beta^\pi\right\|_\infty \leq \frac{(\gamma-\beta)R_{\text{MAX}}}{(1-\gamma)(1-\beta)}.$$

Q3

Let $\alpha, \gamma \in [0,1)$ be two discount factors such that $\gamma \leq \alpha$. Consider a new MDP $\mathcal{M}' = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}', \mathcal{R}, \alpha \rangle$ with a different transition function $\mathcal{T}' : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ defined for $\lambda \in [0,1]$ as

$$\mathcal{T}'(s' \mid s,a) = (1-\lambda)\mathcal{T}(s' \mid s,a) + \lambda \mathbf{1}(s = s'), \quad \forall (s,a,s') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S}.$$

In words, the new transition function $\mathcal{T}'$ follows the transitions of the original MDP $\mathcal{T}$ with probability $(1-\lambda)$ and takes a self-looping transition with probability λ. We will use subscripts to distinguish between value functions of $\mathcal{M}$ versus those of $\mathcal{M}'$.Assuming that both $\mathcal{M}$ and $\mathcal{M}'$ are tabular, recall the matrix form of the Bellman equations for any policy π:

$$V_{\mathcal{M}}^\pi = (I - \gamma\mathcal{T}^\pi)^{-1}\mathcal{R}^\pi \quad V_{\mathcal{M}'}^\pi = (I - \alpha\mathcal{T}'^\pi)^{-1}\mathcal{R}^\pi,$$

where

$$\mathcal{R}^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[\mathcal{R}(s,a)] \quad \mathcal{T}^\pi(s' \mid s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\mathcal{T}(s' \mid s,a)\right] \quad \mathcal{T}'^\pi(s' \mid s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\mathcal{T}'(s' \mid s,a)\right]$$

(a) Give a value of λ such that, for any policy π,

$$V_{\mathcal{M}'}^\pi = \frac{1-\gamma}{1-\alpha} \cdot V_{\mathcal{M}}^\pi$$

(b) If $\pi^\star$ is the optimal policy of MDP $\mathcal{M}$, prove that $\pi^\star$ is also optimal in $\mathcal{M}'$.

1.5.3 Bellman Residuals and Performance Bounds

This problem is adapted from Assignment 1 of CS234 (2023 Spring).

In this problem, we will study **Bellman residuals**, which we will define below, and how it helps us bound the performance of a policy. But first, some recap and definitions.

Recall that the Bellman backup operator B defined below is a contraction with the fixed point as V^*, the optimal value function of the MDP. The symbols have their usual meanings. γ is the discount factor and $0 \leq \gamma < 1$. In all parts, $\|v\|$ is the infinity norm of the vector.

$$(BV)(s) = \max_a \left[r(s,a) + \gamma \sum_{s'\in S} p(s' \mid s,a)\, V(s')\right]$$

We also saw the contraction operator B^π with the fixed point V^π, which is the Bellman backup operator for a particular policy given below:

$$\left(B^{\pi}V\right)(s) = \mathbb{E}_{a\sim\pi}\left[r(s,a) + \gamma \sum_{s'\in S} p\left(s' \mid s,a\right) V\left(s'\right)\right]$$

Previously, we showed that $\|BV - BV'\| \leq \gamma \|V - V'\|$ for two arbitrary value functions V and V'. For the rest of this question, you can also assume that $\|B^{\pi}V - B^{\pi}V'\| \leq \gamma \|V - V'\|$. The proof of this is very similar to the proof we saw in class for B.

(a) Prove that the fixed point for B^{π} is unique.

We can recover a greedy policy π from an arbitrary value function V using the equation below.

$$\pi(s) = \arg\max_{a}\left[r(s,a) + \gamma \sum_{s'\in S} p\left(s' \mid s,a\right) V\left(s'\right)\right]$$

(b) When π is the greedy policy, what is the relationship between B and B^{π} ?

Insight. The *motivation* is that it is often helpful to know what the performance will be if we extract a greedy policy from a value function. In the rest of this problem, we will prove a bound on this performance.

Recall that a value function is a $|S|$-dimensional vector where $|S|$ is the number of states of the MDP. When we use the term V in these expressions as an "arbitrary value function", we mean that V is an arbitrary $|S|$-dimensional vector which need not be aligned with the definition of the MDP at all. On the other hand, V^{π} is a value function that is achieved by some policy π in the MDP. For example, say the MDP has 2 states and only negative immediate rewards. $V = [1, 1]$ would be a valid choice for V even though this value function can never be achieved by any policy π, but we can never have a $V^{\pi} = [1, 1]$. This distinction between V and V^{π} is important for this question and more broadly in reinforcement learning.

Why do we care about setting V to vectors than can never be achieved in the MDP? Sometimes algorithms, such as Deep Q-networks, return such vectors. In such situations we may still want to extract a greedy policy π from a provided V and bound the performance of the policy we extracted, aka V^{π}.

Definition (Bellman Residual). The Bellman residual is defined as $BV - V$ and the Bellman error magnitude is defined as $\|BV - V\|$.

Insight. Bellman residual is an important component of several popular RL algorithms such as the Deep Q-networks, referenced above. Intuitively, you can think of it as the difference between what the value function V specifies at a state and the one-step look-ahead along the seemingly best action at the state using the given value function V to evaluate all future states (the BV term).

(c) For what V does the Bellman error magnitude equal 0?

(d) Prove the following statements for an arbitrary value function V and any policy π. [Hint: Try leveraging the triangle inequality by inserting a zero term.]

$$\|V - V^{\pi}\| \leq \frac{\|V - B^{\pi}V\|}{1-\gamma}$$
$$\|V - V^{*}\| \leq \frac{\|V - BV\|}{1-\gamma}$$

The result you proved in part (d) will be useful in proving a bound on the policy performance in the next few parts. Given the Bellman residual, we will now try to derive a bound on the policy performance, V^{π}.

(e) Let V be an arbitrary value function and π be the greedy policy extracted from V. Let $\varepsilon = \|BV - V\|$ be the Bellman error magnitude for V. Prove the following for any state s. [Hint: Try to use the results from part (d).]

$$V^{\pi}(s) \geq V^{*}(s) - \frac{2\varepsilon}{1-\gamma}$$

A little bit more notation: define $V \leq V'$ if $\forall s, V(s) \leq V'(s)$. What if our algorithm returns a V that satisfies $V^* \leq V$, i.e., it returns a value function that is better than the optimal value function of the MDP. Once again, remember that V can be any vector, not necessarily achievable in the MDP but we would still like to bound the performance of V^π where π is extracted from said V. We will show that if this condition is met, then we can achieve an even tighter bound on policy performance.

(f) Using the same notation and setup as part (e), if $V^* \leq V$, show the following holds for any state s. [Hint: Recall that $\forall \pi, V^\pi \leq V^*$.]

$$V^\pi(s) \geq V^*(s) - \frac{\varepsilon}{1-\gamma}$$

(g) It's not easy to show that the condition $V^* \leq V$ holds because we often don't know V^* of the MDP. Show that if $BV \leq V$ then $V^* \leq V$. Note that this sufficient condition is much easier to check and does not require knowledge of V^*. [Hint: Try to apply induction. What is $\lim_{n\to\infty} B^n V$?]

Insight. A useful way to interpret the results from parts (e) (and (f)) is based on the observation that a constant immediate reward of r at every time-step leads to an overall discounted reward of $r + \gamma r + \gamma^2 r + \ldots = \frac{r}{1-\gamma}$. Thus, the above results say that a state value function V with Bellman error magnitude ε yields a greedy policy whose reward per step (on average), differs from optimal by at most 2ε. So, if we develop an algorithm that reduces the Bellman residual, we're also able to bound the performance of the policy extracted from the value function outputted by that algorithm, which is very useful!

(h) (**Challenge**) It is possible to make the bounds from parts (e) and (f) tighter. Try to prove the following if you're interested.

Let V be an arbitrary value function and π be the greedy policy extracted from V. Let $\varepsilon = \|BV - V\|$ be the Bellman error magnitude for V. Prove the following for any state s :

$$V^\pi(s) \geq V^*(s) - \frac{2\gamma\varepsilon}{1-\gamma}$$

Further, if $V^* \leq V$, prove for any state s

$$V^\pi(s) \geq V^*(s) - \frac{\gamma\varepsilon}{1-\gamma}$$

1.5.4 The Error Bound

This problem is adapted from hw3 of CS285.

Consider the problem of imitation learning within a discrete MDP with horizon T and an expert policy π^*. We gather expert demonstrations from π^* and fit an imitation policy π_θ to these trajectories so that

$$\mathbb{E}_{p_{\pi^*}(s)} \pi_\theta \left(a \neq \pi^*(s) \mid s\right) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{p_{\pi^*}(s_t)} \pi_\theta \left(a_t \neq \pi^*\left(s_t\right) \mid s_t\right) \leq \varepsilon,$$

i.e., the expected likelihood that the learned policy π_θ disagrees with the expert π^* within the training distribution p_{π^*} of states drawn from random expert trajectories is at most ε.

For convenience, the notation $p_\pi\left(s_t\right)$ indicates the state distribution under π at time step t while $p(s)$ indicates the state marginal of π across time steps, unless indicated otherwise.

Q1

Show that $\sum_{s_t} \left|p_{\pi_\theta}\left(s_t\right) - p_{\pi^*}\left(s_t\right)\right| \leq 2T\varepsilon$. Hints:

- In lecture, we showed a similar inequality under the stronger assumption $\pi_\theta\left(s_t \neq \pi^*\left(s_t\right) \mid s_t\right) \leq \varepsilon$ for every $s_t \in \operatorname{supp}\left(p_{\pi^*}\right)$. Try converting the inequality above into an expectation over p_{π^*}.
- Use the union bound inequality: for a set of events $E_i, \Pr\left[\bigcup_i E_i\right] \leq \sum_i \Pr\left[E_i\right]$.

Q2

Consider the expected return of the learned policy π_θ for a state-dependent reward $r(s_t)$, where we assume the reward is bounded with $|r(s_t)| \leq R_{\max}$:

$$J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{p_\pi(s_t)} r(s_t)$$

(a) Show that $J(\pi^*) - J(\pi_\theta) = \mathcal{O}(T\varepsilon)$ when the reward only depends on the last state, i.e., $r(s_t) = 0$ for all $t < T$.

(b) Show that $J(\pi^*) - J(\pi_\theta) = \mathcal{O}\left(T^2\varepsilon\right)$ for an arbitrary reward.