Notes on RL

Haosheng Zhou

August, 2022

Contents

This note refers to the book *Reinforcement Learning: an introduction, by R.S. Sutton and A.G. Barto*. **All plots are generated by Python and all codes can be found at https://github.com/Haosheng-Zhou/RL_code.**

Introduction

The essential point of reinforcement learning (RL) is the engagement of **reward**. The final objective of RL is to achieve the highest possible aggregated reward over a period of time. However, greedy strategy always taking the action with the highest reward based on the current knowledge is not the best thing to do since short-sighted actions may cause the missing of a potentially better action in long term.

RL problem is modelled by introducing an **environment** and a set of **actions** that can change the **states** of the environment. Different rewards are then given based on different states and actions. The states, actions and rewards at time t are denoted S_t, A_t, R_t as random variables, while s_t, a_t, r_t are real numbers/vectors that stand for the values of those random variables. In each turn, the player chooses an action to take that results in the change of the state. The rewards in each turn is given based on the current state-action pair. The main difficulty in RL is to enable the computer to think in a long-term sense. As a classical approach in the control setting, **value function** is backwardly constructed to indicate the long-term benefits of a certain action at a certain state. From statistical decision theory, randomized decision rule called **policy** is also introduced to maintain the exploration among different actions.

However, there are still other specific difficulties we have to face in RL, including large state space (e.g. chess), estimation of value functions, dealing with multi-player games etc. In particular, a simplified RL problem with no state transition can be described as a bandit problem, which still has its importance despite its simplicity. We start from introducing bandit problems, then to the general RL problem setting and RL algorithms.

Bandit Problem

Basics of Bandit Problems

The bandit problem refers to a situation where there's always K actions to choose from at any time and there's only a single state in the state space. As a result, the reward only depends on which action to take and is assumed to be random (deterministic reward is trivial for bandit problem). Different actions have their respective mean of reward, called their **value**. The mean reward for action a is denoted $q_*(a)$,

$$q_*(a) = \mathbb{E}[R_t | A_t = a] \tag{1}$$

the reward of taking action a, WLOG, is a random number generated from $N(q_*(a), 1)$, and rewards are assumed to be independently generated. With a finite time horizon T provided, the goal is to find a strategy to produce a sequence of actions $A_1, ..., A_T$ that maximizes the aggregated reward $\sum_{i=1}^{T} R_i$.

Remark. *The bandit problem we consider is the stationary bandit problem, i.e. the distribution of R_t does not change w.r.t. time. On the other hand, if the distribution of R_t changes w.r.t. time, it's called an non-stationary bandit problem and is much harder to analyze.*

Unfortunately, we don't know about $q_*(a)$ (otherwise we would always take the action with the highest value). However, it's still not too bad because we might learn from the experience of playing this game and construct estimates for those values. The estimates for $q_*(a)$ at time t is denoted $Q_t(a)$. It seems natural that at time t, we shall look at $Q_t(a)$ for all possible action a to find the action with the highest estimated value, which is called a **greedy action**.

Based on current knowledge, the greedy action is definitely the best, but shall we necessarily take it? The answer is NO because our knowledge evolve with time. If we **exploit** (choose the greedy action) all the time and do not **explore** (try other actions), we would not be able to update our knowledge on other actions. This would result in the consequence that a potentially better action is never discovered. On the other hand, exploring too much is also not what we want since the loss in the aggregated reward would be too much. That means, **the trade-off between exploitation and exploration** is a key point in RL.

Let's think about how we shall construct the estimated value function Q_t to our current knowledge. $Q_t(a)$ should provide an estimate for the mean reward received by taking action a based on the experience until time t. It's natural that sample mean would be a good approximation to population mean,

$$Q_t(a) \overset{def}{=} \frac{\sum_{i=1}^{t-1} R_i \mathbb{I}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{I}_{A_i=a}}, N_t(a) = \sum_{i=1}^{t-1} \mathbb{I}_{A_i=a} \tag{2}$$

where $\mathbb{I}$ stands for the indicator. Such estimate $Q_t(a)$ is formed as the sample average of rewards before time t when action a is taken, called the **sample average** method for estimating action values, $N_t(a)$ is the number of times

action a is taken before time t. The greedy action is formulated as:

$$A_t = \arg\max_a Q_t(a) \tag{3}$$

ε-greedy Strategy

To set space for exploration to happen, ε-greedy strategy is introduced. In each turn, we behave greedily with probability $1-\varepsilon$ and explore with probability ε. When exploring, uniformly randomly take one action out of all possible actions to gain knowledge. Refer to Fig. 1 for advantages of exploration in long term.

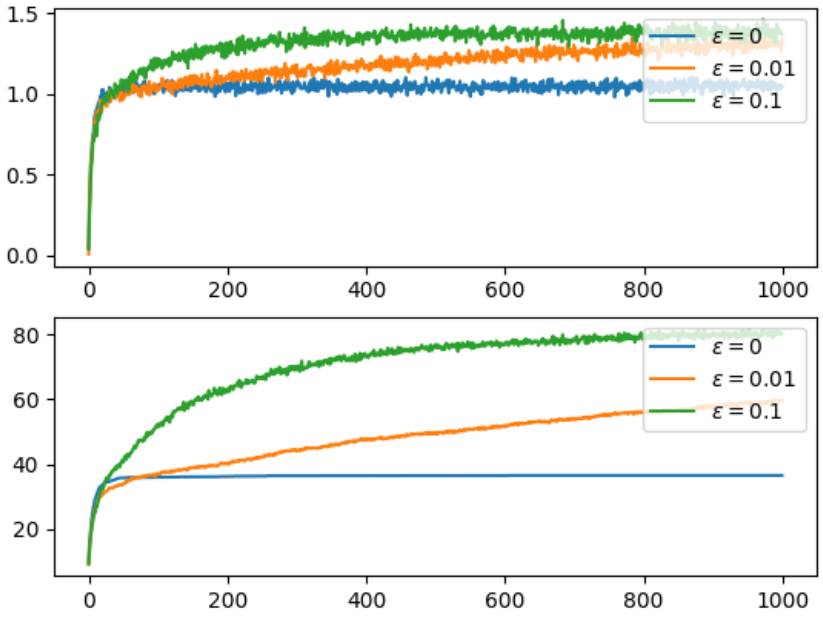

Figure 1: Advantage of ε-greedy strategy over greedy strategy

A K bandit-model with $K = 10, T = 1000$, action values are generated following $N(0,1)$ and rewards of action a are generated following $N(q_*(a), 1)$.
The upper plot shows the average reward derived at each step and the lower plot shows the percentage for the current action to be the best action at each step. All data points are averages among 2000 different bandit problems.
As can be seen, greedy strategy does not explore at all. Although it's converging faster, the reward derived is far from optimal. As ε is slowly increased, more explorations are conducted, resulting in both the average reward and the best action percentage rate being much higher than that of the greedy strategy.

Remark. *$Q_t(a)$ can actually be written in an **incremental form**. When action a is taken at time t resulting in*

reward R_t,

$$\forall a' \neq a, Q_{t+1}(a') = Q_t(a') \tag{4}$$

while

$$Q_{t+1}(a) = \frac{\sum_{i=1}^{t-1} R_i \mathbb{I}_{A_i=a} + R_t}{\sum_{i=1}^{t-1} \mathbb{I}_{A_i=a} + 1} = \frac{N_t(a)Q_t(a) + R_t}{N_t(a) + 1} = Q_t(a) + \frac{1}{N_t(a) + 1}(R_t - Q_t(a)) \tag{5}$$

which means that we don't have to record and look through all history of actions and rewards each time we have seen a new action a.

Note that this incremental form also provides the intuition that we are adjusting the old estimate $Q_t(a)$ in the direction of the difference $R_t - Q_t(a)$ by a **step size factor** $\alpha_{t+1} = \frac{1}{N_t(a)+1}$. As a result, when it comes to non-stationary bandit problems where the reward distribution changes over time, we may replace the step size factor $\frac{1}{N_t(a)+1}$ with a fixed constant α formed as a hyperparameter.

Constant Step Size Methods for Non-stationary Problems

For the stationary bandit problem with sample mean method to estimate the value, use $\alpha_n(a)$ to denote the step size factor when seeing action a for the n-th time, then $\alpha_n = \frac{1}{n}$, satisfying the well-known convergence condition for stochastic optimization that $\sum_n \alpha_n(a) = \infty, \sum_n \alpha_n^2(a) < \infty$ (it's the iff condition for the a.s. convergence of stochastic optimization with varying step size). Although a constant step size factor violates these conditions, it's still adopted for its simplicity and effectiveness. Refer to Fig. 2 for the numerical experiment of **non-stationary bandit problem with constant step size** compared to that with sample mean method.

Optimistic Start

Some tricks for picking up special initial values of the estimated value Q may apply for stationary bandit problems, e.g. the trick of optimistic start. Since in our model the distribution of action values has mean 0. If we could start with an optimistic estimated value (set the initial values of Q as big positive numbers), after picking up an action to take and figuring out the reward of such action, the bandit algorithm would realize that the actual reward is far lower than the optimistic start provided. This results in the disappointment towards such an action so that other actions that still have the optimistic start with them will be explored.

Refer to Fig. 3 for numerical experiments. Although this trick works well for some bandit problems, the performance is not guaranteed so it does not deserve much attention.

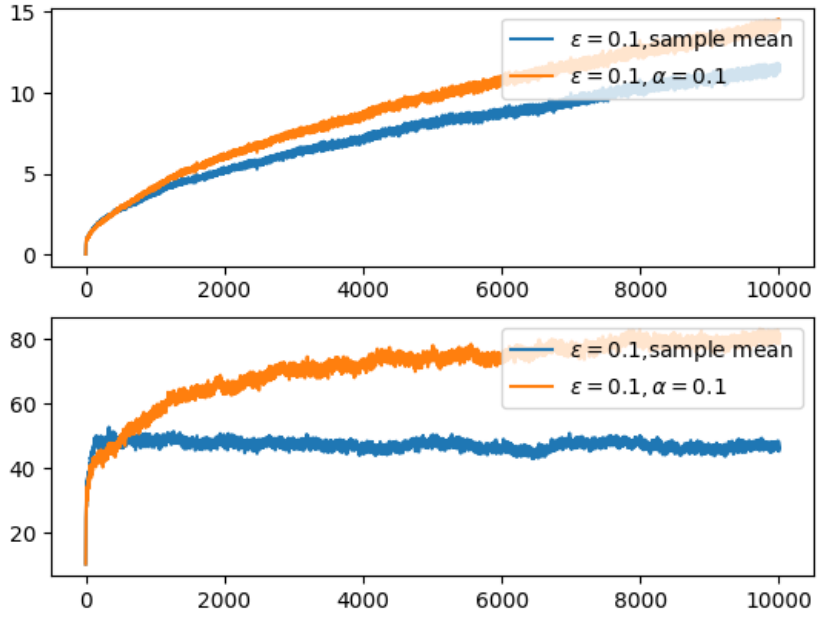

Figure 2: An non-stationary bandit problem

A K bandit-model with $K = 10, T = 10000$, values of actions are generated following $N(0, 1)$ but change over time. Each value has an independent $N(0, 0.01)$ random number added to it on each time step. The rewards of action a are generated following $N(q_*(a), 1)$.
The upper plot shows the average reward derived at each step and the lower plot shows the percentage for the current action to be the best action at each step. All data points are averages among 1000 different bandit problems.
As can be seen, sample mean method as a method with varying step size parameter $\alpha_n(a) = \frac{1}{n}$, behaves not as well as the value estimate with constant step size parameter $\alpha = 0.1$ practically.

Upper Confidence Bound (UCB) Method

In the context above, discussion are organized within the scope of ε-greedy strategy. When the algorithm decides to explore, all available actions are given the same chance of being selected. However, it would have been much more efficient if the actions that are more likely to be the optimal ones could be explored. This leads to the **upper confidence bound (UCB)** method that first visits all possible actions once and then selects action based on

$$A_t = \arg\max_a \left(Q_t(a) + c\sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(a)}} \right) \tag{6}$$

where $\delta > 0$ is a small probability to be specified and $c > 0$ is a hyperparameter for the degree of exploration. If action a is selected, the increase of $N_t(a)$ narrows the confidence bound and it's believed that the updated $Q_t(a)$

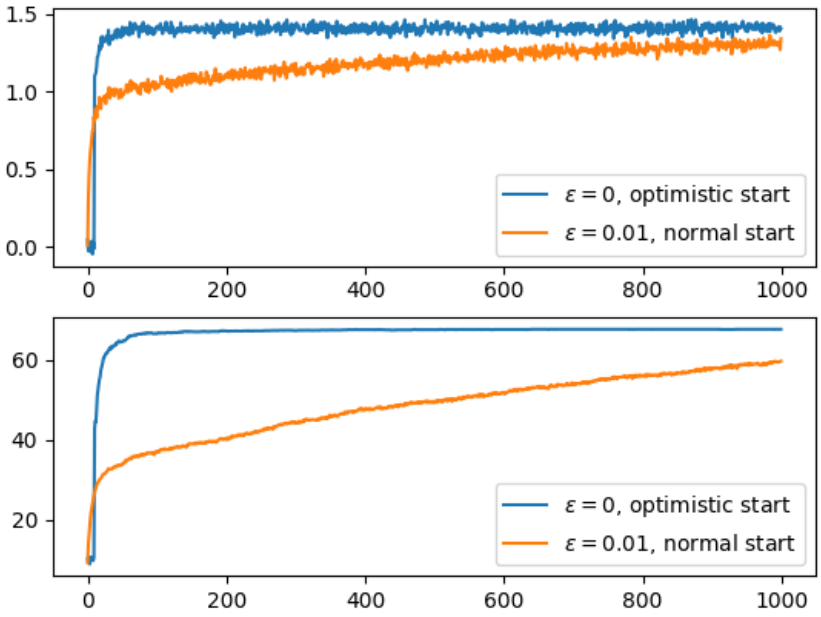

Figure 3: Trick of optimistic start

A stationary K bandit-model with $K = 10, T = 1000$, values of actions are generated following $N(0, 1)$. The rewards of action a are generated following $N(q_*(a), 1)$.
The upper plot shows the average reward derived at each step and the lower plot shows the percentage for the current action to be the best action at each step. All data points are averages among 2000 different bandit problems.
As can be seen, for stationary bandit problem, the bandit algorithm with greedy strategy and optimistic start (for any action a, initial values $Q_1(a) = 5$) converges much faster than that with ε-greedy strategy ($\varepsilon = 0.01$) and normal start (for any action a, initial values $Q_1(a) = 0$).

reflects more accurate information on action a. In contrast, if action a has never been visited again since the start of the game, it has quite a high chance to be explored as time goes by, except that some really good action with a high action value exists.

The algorithm gets its name because $Q_t(a) + c\sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(a)}}$ is the upper confidence bound of $Q_t(a)$, so the strategy is called **optimism towards uncertainty**, i.e. taking actions based on the upper confidence bound to ensure exploration. The reason why such expression provides a confidence bound remains a mystery at this point. Due to the importance of UCB, we will provide a proof for this algorithm in a later context.

Remark. *The UCB algorithm for bandit problem has a lot of different variants built to satisfy different bounds. Here we present one of them which has a nice probabilistic concentration bound shown in a later context. Just don't be surprised when you see slightly different algorithms all with the name UCB.*

Gradient Bandit Methods

So far, the action-value methods we have discussed all focus on picking up the action that maximizes some kind of estimates for action value to ensure exploration. However, in order to maintain a certain amount of exploration, why don't let randomness help us, i.e. why don't we **randomize the action**? That's exactly the motivation of gradient bandit method where a numerical **preference** for each action a at time t, denoted $H_t(a)$ is introduced to generate a probability distribution on the action space.

In order to turn the real numbers into a probability distribution, the softmax function is naturally considered

$$\pi_t(a) \overset{def}{=} \mathbb{P}\left(A_t = a\right) \tag{7}$$

$$\overset{def}{=} \frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}} \tag{8}$$

where $\pi_t(a)$ denotes the probability of taking action a at time t.

The only question we have to think about now is how to update the preference H_t based on the experience. Gradient bandit algorithm tells us that it's given by

$$H_{t+1}(a) = H_t(a) + \alpha\left(R_t - \overline{R}_t\right)\left(\mathbb{I}_{A_t=a} - \pi_t(a)\right) \tag{9}$$

where $\alpha > 0$ is the learning rate as a hyperparameter and $\overline{R}_t$ is the mean reward received up to time t with time t included, which works as a **baseline**.

Remark. *A direct explanation is that for action a, if it has been selected as the optimal action at time t, the preference shall rise if the reward is higher than past average (a positive effect). For other actions that have not been selected at time t, the similar logic holds, with the direction of change different from that of the selected action. This algorithm exhibits the process of learning, i.e. prioritizing the actions that bring with benefits and eliminating the actions that bring with loss.*

The update of preference seems intuitively correct but where does it come from? The answer is gradient. Keep in mind that in the context of bandit problem our goal is to maximize the aggregated expected reward received. Viewed in single time step, H_t shall be updated such that it maximized the expected reward received at time t, which is $\mathbb{E}R_t$. From a gradient ascent perspective,

$$H_{t+1}(a) = H_t(a) + \alpha\frac{\partial \mathbb{E}R_t}{\partial H_t(a)} \tag{10}$$

with

$$\mathbb{E}R_t = \sum_a \mathbb{P}\left(A_t = a\right)\mathbb{E}(R_t|A_t = a) \tag{11}$$

$$= \sum_a \pi_t(a) q_*(a) \tag{12}$$

However, $q_*(a)$ is unknown so such gradient ascent method can't be implemented practically. Despite this fact, let's still calculate the partial derivative to see what we can get.

$$\frac{\partial \mathbb{E}R_t}{\partial H_t(a)} = \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \tag{13}$$

$$= \sum_x q_*(x) \frac{\partial \frac{e^{H_t(x)}}{\sum_b e^{H_t(b)}}}{\partial H_t(a)} \tag{14}$$

$$= \sum_x q_*(x) \frac{\mathbb{I}_{a=x} e^{H_t(x)} \sum_b e^{H_t(b)} - e^{H_t(x)} e^{H_t(a)}}{\left(\sum_b e^{H_t(b)}\right)^2} \tag{15}$$

$$= \sum_x q_*(x) \pi_t(x) \left(\mathbb{I}_{a=x} - \pi_t(a)\right) \tag{16}$$

notice that there is a special structure of π_t that

$$\sum_x \pi_t(x) = 1 \tag{17}$$

$$\sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = \sum_x \pi_t(x) \mathbb{I}_{x=a} - \sum_x \pi_t(x) \pi_t(a) \tag{18}$$

$$= \pi_t(a) - \pi_t(a) = 0 \tag{19}$$

use this property to add an arbitrary baseline B_t to be specified later (as a function of $R_1, ..., R_t$ already observed),

$$\frac{\partial \mathbb{E}R_t}{\partial H_t(a)} = \sum_x \left(q_*(x) - B_t\right) \frac{\partial \pi_t(x)}{\partial H_t(a)} \tag{20}$$

$$= \sum_x \pi_t(x) \left(q_*(x) - B_t\right) \left(\mathbb{I}_{a=x} - \pi_t(a)\right) \tag{21}$$

Since π_t is a probability distribution on the action space, this partial derivative has a natural structure as an expectation w.r.t. the action $A_t \sim \pi_t$:

$$\frac{\partial \mathbb{E}R_t}{\partial H_t(a)} = \mathbb{E}_{A_t \sim \pi_t} \left[\left(q_*(A_t) - B_t\right) \left(\mathbb{I}_{A_t=a} - \pi_t(a)\right)\right] \tag{22}$$

up to this point, the original gradient ascent scheme can be rewritten as

$$H_{t+1}(a) = H_t(a) + \alpha \frac{\partial \mathbb{E}R_t}{\partial H_t(a)} \tag{23}$$

$$= H_t(a) + \alpha \mathbb{E}_{A_t \sim \pi_t} \left[\left(q_*(A_t) - B_t\right) \left(\mathbb{I}_{A_t=a} - \pi_t(a)\right)\right] \tag{24}$$

with action value $q_*(A_t)$ unknown, which has exactly the form of **stochastic gradient ascent** method, where the ascent direction is taken randomly and it's only ensured that the expectation of the random ascent direction equals

the true gradient direction.

Although it's impossible to figure out this expectation (the true gradient), it's completely possible to sample random directions from the distribution with mean $\mathbb{E}_{A_t\sim\pi_t}\left[(q_*(A_t)-B_t)\left(\mathbb{I}_{A_t=a}-\pi_t(a)\right)\right]$. To show why it's possible, $q_*(A_t)=\mathbb{E}(R_t|A_t)$ results in

$$\mathbb{E}_{A_t\sim\pi_t}\left[(q_*(A_t)-B_t)\left(\mathbb{I}_{A_t=a}-\pi_t(a)\right)\right]=\mathbb{E}_{A_t\sim\pi_t,R_t}\left[(R_t-B_t)\left(\mathbb{I}_{A_t=a}-\pi_t(a)\right)\right] \tag{25}$$

and $(R_t-B_t)(\mathbb{I}_{A_t=a}-\pi_t(a))$ is the random direction we want, with R_t observable.

So far, we have proved that **the gradient bandit algorithm is an instance of stochastic gradient ascent**. Its general form for any baseline B_t as a function of $R_1,...,R_t$ is as follows:

$$H_{t+1}(a)=H_t(a)+\alpha\left(R_t-B_t\right)\left(\mathbb{I}_{A_t=a}-\pi_t(a)\right) \tag{26}$$

in practice, $B_t=\overline{R}_t$ as past averages works pretty well. Although the gradient bandit algorithm is ensured to be an instance of stochastic gradient ascent regardless of the value of B_t, a lack of baseline for which $B_t=0$ has poor performance empirically.

Comparison of Bandit Algorithms

The comparison can be conducted in different criterion. See Fig. 4 for one comparison criterion. Actually the most useful comparison criterion in the setting of bandit problem is the cumulative regret, see Fig. 5, 6 for the plots and later sections for the definition and theoretical analysis.

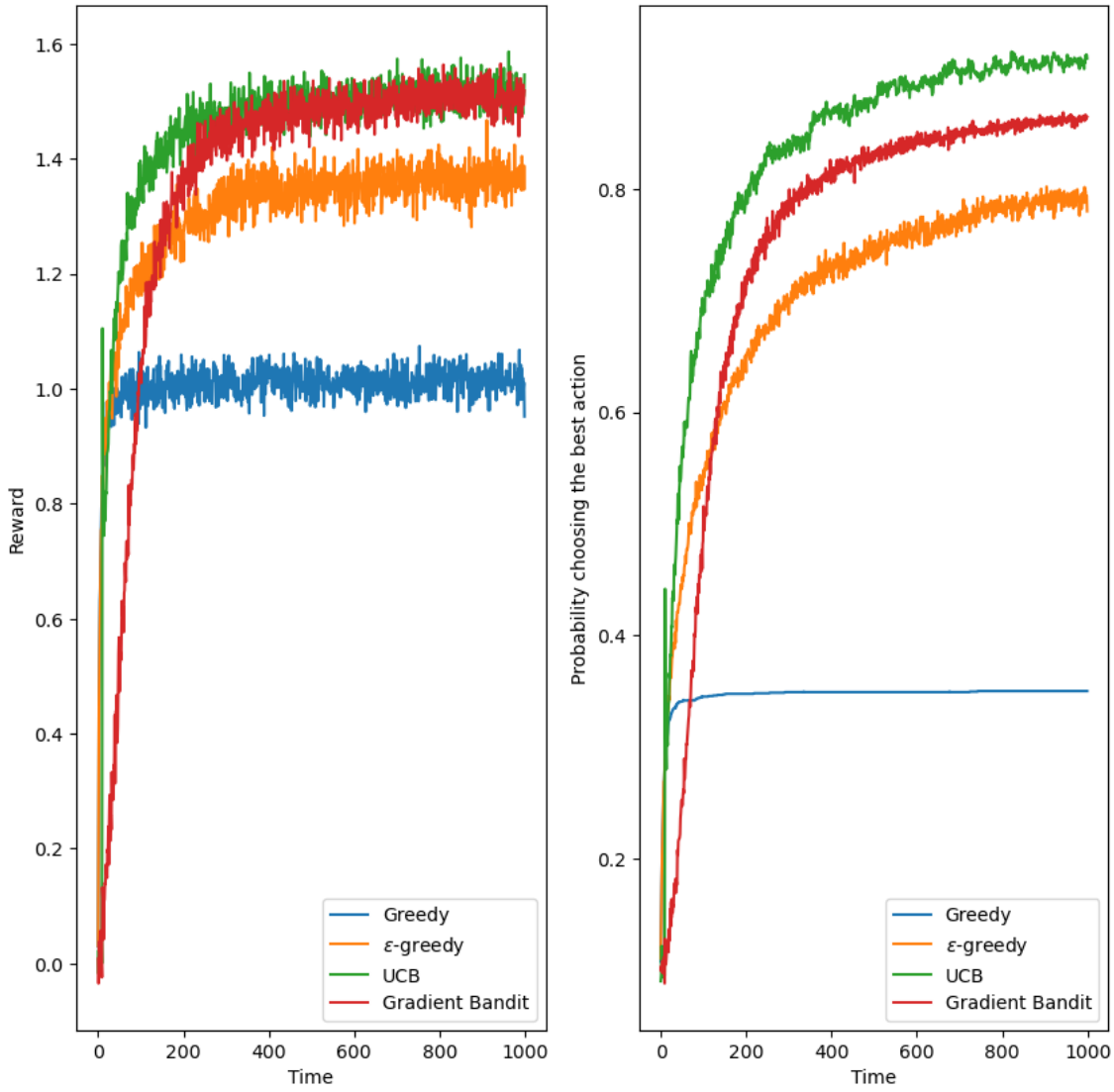

Figure 4: Comparison of Bandit Algorithms
The numerical experiment setting is the same as that in the plots above. Four algorithms are attempted. The left plot shows the reward we get as time goes by while the right plot shows the probability of choosing the best action as time goes by.

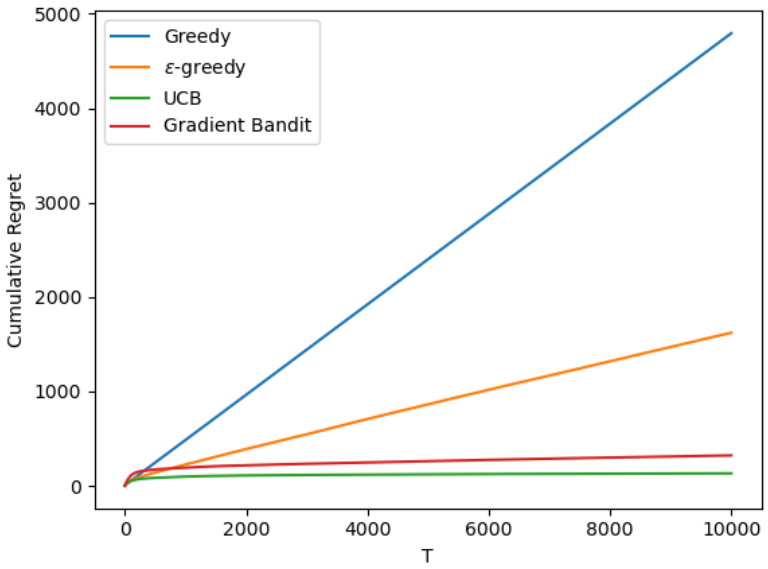

Figure 5: Cumulative Regret of Bandit Algorithms

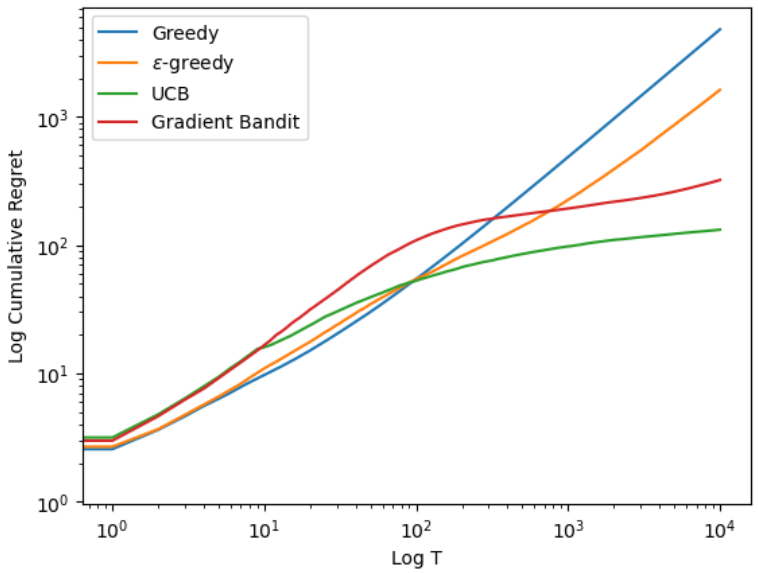

Figure 6: Cumulative Regret of Bandit Algorithms, Log-Log Plot

Markov Decision Process (MDP)

In the last section, we have looked into RL problems that have only one state and multiple actions. Action-value methods can be applied to estimate the value of each action and select the action with the highest estimated value. Gradient bandit method, as an instance of stochastic gradient ascent method, updates the preferences such that the preferences maximize the expectation of the reward.

However, the introduction of different states brings with much complexity. The set of available actions may depend on the state, thus affecting the selection for the optimal action. Under such circumstances, the introduction of MDP helps to set up the general RL framework.

Basic Settings

The MDP adopts the agent-environment interaction, where the **agent** means the learner or decision maker and the **environment** means everything other than the agent. The logic is that on seeing the state the agent is at and the reward the agent receives, the agent makes an action that changes the environment, causing updates in the state and reward. The state $s \in \mathscr{S}$, with $\mathscr{S}$ to be the set of all possible states. The action $a \in \mathscr{A}(s)$, with $\mathscr{A}(s)$ to be the set of all possible actions given state s (Note that the set of available actions may depend on the state). The reward $r \in \mathscr{R} \subset \mathbb{R}$. MDP is generally formed as a sequential decision-making process, generating a sequence $S_0, A_0, R_1, S_1, A_1, R_2, ...$ of states, actions and rewards, where R_{t+1} is actually the reward for selecting the action A_t.

Only **finite MDP** is in our scope, for which the sets of states, actions and rewards are all finite. It's then obvious that we can denote the set of all possible actions as $\mathscr{A}$ now, not depending on the state s and it remains finite. The **dynamics** of MDP decides how a sequence is generated and how states, actions and rewards come into play, and it's assumed to be time homogeneous.

$$p(s', r|s, a) \overset{def}{=} \mathbb{P}\left(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\right) \tag{27}$$

The "Markov" in the name of MDP actually refers to the fact that given the present, i.e. S_t, A_t, the future, i.e. $R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, ...$ is independent of the past, i.e. $S_0, A_0, ..., S_{t-1}, A_{t-1}$. Here some notations are introduced as functions of the dynamics.

$$\begin{aligned} p(s'|s, a) &\overset{def}{=} \mathbb{P}\left(S_t = s'|S_{t-1} = s, A_{t-1} = a\right) && (28)\\ &= \sum_r \mathbb{P}\left(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\right) && (29)\\ &= \sum_r p(s', r|s, a) && (30) \end{aligned}$$

gives the **state-transition probabilities**.

$$r(s,a) \overset{def}{=} \mathbb{E}\left(R_t | S_{t-1}=s, A_{t-1}=a\right) \tag{31}$$
$$= \sum_r r \cdot \mathbb{P}\left(R_t = r | S_{t-1}=s, A_{t-1}=a\right) \tag{32}$$
$$= \sum_r r \cdot \sum_{s'} p(s', r|s, a) \tag{33}$$

gives the **expected rewards for state-action pairs**.

$$r(s,a,s') \overset{def}{=} \mathbb{E}\left(R_t | S_{t-1}=s, A_{t-1}=a, S_t=s'\right) \tag{34}$$
$$= \sum_r r \cdot \mathbb{P}\left(R_t = r | S_{t-1}=s, A_{t-1}=a, S_t=s'\right) \tag{35}$$
$$= \sum_r r \cdot \frac{p(s', r|s, a)}{\mathbb{P}\left(S_t = s' | S_{t-1}=s, A_{t-1}=a\right)} \tag{36}$$
$$= \frac{\sum_r r \cdot p(s', r|s, a)}{\sum_r p(s', r|s, a)} \tag{37}$$

gives the **expected rewards for state-action-next-state triples**.

Construction of Return

MDP is set up with a **reward hypothesis**, i.e. we want to maximize the expectation of the cumulative sum of the rewards. Since the rewards are constructed and given for each RL problem, the design of rewards is always subtle. A good design can inform the computer of the real objective and clarify what one really hope to achieve.

RL tasks are basically separated into two kinds based on their time structures: **episodic tasks** and **continuing tasks**. Episodic tasks see the end of the environment for each episode, and different episodes are independent. For such tasks, a stopping time T is often used to denote the random time of the end of the current episode. Continuing tasks bring with more difficulties since the task shall never end. To ensure that "the expectation of the cumulative sum of the rewards" is well-defined, **discounting techniques** are introduced to compute the expected discounted return.

$$G_t \overset{def}{=} \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{38}$$

where γ is the discount rate, T can take value as ∞ and G_t is called the **return** at time t. **NOTE:** R_{t+1} is actually the reward given based on state S_t and action A_t, so the return at time t actually contains information of the reward based on state and action at time t.

Similarly to what we have done in the last section, the return can be written in its incremental form for the

simplicity of realization.

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{39}$$

Actually, such discounted return works for both episodic tasks and continuing tasks by adding an absorbing state to the episodic task that always generates reward 0. That's the reason why we won't deal with these two kinds of problems separately. **NOTE**: Return is totally different from reward since return takes long-term rewards into consideration. We will see that it's much more reasonable to use returns to define value functions than rewards because the motivation of value functions is just to measure the long-term value of a certain action.

Policy and Value Functions

After defining the return that focuses on the long-term discounted reward, the estimation of value functions is naturally built upon the return. However, before introducing the definitions of value functions, it has to be clarified that we are actually training something called **policy** in the RL problem. The policy π is actually a conditional distribution, telling us the probability taking any action given the current state.

$$\pi(a|s) = \mathbb{P}\left(A_t = a | S_t = s\right) \tag{40}$$

If the current state is S_t and actions are selected according to policy π, the the expectation of R_{t+1} in terms of π and the dynamics p would be:

$$\begin{aligned}
\mathbb{E}R_{t+1} &= \sum_r r \cdot \mathbb{P}\left(R_{t+1} = r\right) && (41)\\
&= \sum_r r \cdot \sum_{s,a} \mathbb{P}\left(S_t = s, A_t = a\right) \mathbb{P}\left(R_{t+1} = r | S_t = s, A_t = a\right) && (42)\\
&= \sum_r r \cdot \sum_{s,a} \pi(a|s) \mathbb{P}\left(S_t = s\right) \cdot \sum_{s'} p(s', r | s, a) && (43)\\
& && (44)
\end{aligned}$$

Since we already know that the current state is S_t, the conclusion is that (Exercise 3.11)

$$\mathbb{E}R_{t+1} = \sum_r r \cdot \sum_a \pi(a|S_t) \cdot \sum_{s'} p(s', r | S_t, a) \tag{45}$$

Now let's introduce the notions of general value functions. Value functions are tightly related to the policy π chosen, and should also depend on the state or action. The **state value function for policy** π is defined as the expectation of return conditional on the given current state:

$$v_\pi(s) \overset{def}{=} \mathbb{E}_\pi(G_t|S_t = s) \tag{46}$$

$$= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\Big|S_t = s\right) \tag{47}$$

Another variant of value function has action involved and is called **the action value function for policy** π:

$$q_\pi(s,a) \overset{def}{=} \mathbb{E}_\pi(G_t|S_t = s, A_t = a) \tag{48}$$

$$= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\Big|S_t = s, A_t = a\right) \tag{49}$$

Let's provide a formula for v_π in terms of q_π, π. (Exercise 3.12)

$$v_\pi(s) = \mathbb{E}_\pi(G_t|S_t = s) \tag{50}$$

$$= \sum_a \mathbb{P}\left(A_t = a|S_t = s\right)\mathbb{E}_\pi(G_t|S_t = s, A_t = a) \tag{51}$$

$$= \sum_a q_\pi(s,a)\pi(a|s) \tag{52}$$

Similarly, provide a formula for q_π in terms of v_π, p. (Exercise 3.13)

$$q_\pi(s,a) = \mathbb{E}_\pi(G_t|S_t = s, A_t = a) \tag{53}$$

$$= \mathbb{E}_\pi(R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a) \tag{54}$$

$$= \mathbb{E}_\pi(R_{t+1}|S_t = s, A_t = a) + \gamma\mathbb{E}_\pi(G_{t+1}|S_t = s, A_t = a) \tag{55}$$

$$= \sum_r r \cdot \sum_{s'} p(s', r|s, a) + \gamma\sum_{s'}\mathbb{E}_\pi(G_{t+1}|S_{t+1} = s', S_t = s, A_t = a)\mathbb{P}\left(S_{t+1} = s'|S_t = s, A_t = a\right) \tag{56}$$

$$= \sum_r r \cdot \sum_{s'} p(s', r|s, a) + \gamma\sum_{s'} v_\pi(s')\sum_r p(s', r|s, a) \tag{57}$$

$$= \sum_{r,s'}(r + \gamma v_\pi(s'))p(s', r|s, a) \tag{58}$$

Note that $\mathbb{E}_\pi(G_{t+1}|S_{t+1} = s', S_t = s, A_t = a) = \mathbb{E}_\pi(G_{t+1}|S_{t+1} = s')$ is due to the Markov property since G_{t+1} is a function of $R_{t+2}, R_{t+3}, ...$ Recognize the time $t+1$ as present, the states and actions before time $t+1$ are the past and the rewards after time $t+1$ are the future, so they are independent.

To understand these two value functions deeper, let's consider some other formulations.

$$q_\pi(s,a) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\Big|S_t=s,A_t=a\right) \tag{59}$$

$$= \sum_{k=0}^{\infty}\gamma^k \mathbb{E}_\pi\left(R_{t+k+1}\Big|S_t=s,A_t=a\right) \tag{60}$$

$$\tag{61}$$

Notice that $\mathbb{E}_\pi\left(R_{t+k+1}\Big|S_t=s,A_t=a\right)$ has nothing to do with t since the MDP is Markov and time-homogeneous and the policy does not change with time:

$$\mathbb{E}_{\pi,S_0,A_0}\left(R_{t+k+1}\Big|S_t,A_t\right) \tag{62}$$

$$= \mathbb{E}_{\pi,S_0,A_0}\left(R_{k+1}\circ\theta_t\Big|\mathscr{F}_t\right) \tag{63}$$

$$= \mathbb{E}_{\pi,S_t,A_t}\left(R_{k+1}\right) \tag{64}$$

$$\mathbb{E}_\pi\left(R_{t+k+1}\Big|S_t=s,A_t=a\right) = \mathbb{E}_{\pi,S_t=s,A_t=a}\left(R_{k+1}\right) \tag{65}$$

The same thing happens for q_π since $v_\pi(s) = \sum_a q_\pi(s,a)\pi(a|s)$. As a result, we conclude that both the state value function and the action value function are **time-invariant**.

After stating the properties and connections between value functions, let's briefly talk about estimating value functions in practice. A direct thought is to do **Monte Carlo** simulations since v_π, q_π both have the structure as conditional expectations. By adopting policy π to do simulations, multiple chains $S_0, A_0, R_1, ...$ would be derived. For each of these chains, pick out the realizations of states equal to s (here assume that we have $S_t = s$), then G_t, the return at time t can be computed since all rewards afterwards are known. By taking the sample average of all G_t such that $S_t = s$, an estimate for $v_\pi(s)$ is then constructed. Similar operations work for estimating $q_\pi(s,a)$, the only difference is that we have to make sure both the state and the action at a certain time meet the restrictions.

Despite the help of Monte Carlo, many problems exist for value function estimations. One might assert that such estimation may take too long time, which often happens when a certain state s has very little possibility appearing under policy π, so the appearance of s cannot be empirically observed, resulting in the failure of Monte Carlo. One might also assert that when there are too many states, Monte Carlo will do a poor job since the convergence of Monte Carlo greatly depends on the size of samples. If each state is only distributed with a small number of samples, the estimation will be largely biased. Besides, it's also impossible to keep record of $v_\pi(s)$ for each state s and policy π, so some parametric skills would be required. That's why there's still much work to do in the field of estimating value functions.

The last thing to say is the **Bellman Equation**. It's an equation exhibiting the structure of value functions and can be widely used in the estimation of value functions. The Bellman equation w.r.t. v_π is provided below with

the use of Markov property:

$$v_\pi(s) = \mathbb{E}(G_t|S_t = s) \tag{66}$$
$$= \mathbb{E}(R_{t+1}|S_t = s) + \gamma\mathbb{E}(G_{t+1}|S_t = s) \tag{67}$$
$$= \sum_r r \cdot \mathbb{P}\left(R_{t+1} = r|S_t = s\right) + \gamma \sum_{s',a} \mathbb{P}\left(S_{t+1} = s', A_t = a|S_t = s\right) \mathbb{E}(G_{t+1}|S_{t+1} = s', S_t = s, A_t = a) \tag{68}$$
$$= \sum_r r \cdot \sum_a \mathbb{P}\left(A_t = a|S_t = s\right) \mathbb{P}\left(R_{t+1} = r|S_t = s, A_t = a\right) + \gamma \sum_{s',a} \mathbb{P}\left(S_{t+1} = s', A_t = a|S_t = s\right) \mathbb{E}(G_{t+1}|S_{t+1} = s') \tag{69}$$
$$= \sum_r r \cdot \sum_a \pi(a|s) \cdot \sum_{s'} p(s', r|s, a) + \gamma \sum_{s',a} \mathbb{P}\left(S_{t+1} = s'|S_t = s, A_t = a\right) \mathbb{P}\left(A_t = a|S_t = s\right) v_\pi(s') \tag{70}$$
$$= \sum_r r \cdot \sum_a \pi(a|s) \cdot \sum_{s'} p(s', r|s, a) + \gamma \sum_{s',a} \sum_r p(s', r|s, a)\pi(a|s)v_\pi(s') \tag{71}$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{72}$$

The similar operations provide the Bellman equation w.r.t. q_π with the use of the relationship between v_π and q_π(Exercise 3.17):

$$q_\pi(s, a) = \mathbb{E}(G_t|S_t = s, A_t = a) \tag{73}$$
$$= \mathbb{E}(R_{t+1}|S_t = s, A_t = a) + \gamma\mathbb{E}(G_{t+1}|S_t = s, A_t = a) \tag{74}$$
$$= \sum_r r \cdot \mathbb{P}\left(R_{t+1} = r|S_t = s, A_t = a\right) + \gamma \sum_{s'} \mathbb{P}\left(S_{t+1} = s'|S_t = s, A_t = a\right) \mathbb{E}(G_{t+1}|S_{t+1} = s', S_t = s, A_t = a) \tag{75}$$
$$= \sum_r r \cdot \sum_{s'} p(s', r|s, a) + \gamma \sum_{s'} \sum_r p(s', r|s, a)v_\pi(s') \tag{76}$$
$$= \sum_r r \cdot \sum_{s'} p(s', r|s, a) + \gamma \sum_{s'} \sum_r p(s', r|s, a) \cdot \sum_{a'} \pi(a'|s')q_\pi(s', a') \tag{77}$$
$$= \sum_{r,s'} p(s', r|s, a)[r + \sum_{a'} \gamma\pi(a'|s')q_\pi(s', a')] \tag{78}$$

As a result, given policy π and MDP dynamics p, the two Bellman equations are linear systems in $v_\pi(s)$ and $q_\pi(s, a)$. For a small RL model, these value functions can be derived easily and accurately.

Gridworld Model as an Example

Let's now try to set up a simple RL task called **Gridworld** (Example 3.5). All possible states are the 25 grids in the 5×5 square and all possible actions at each state are going north, south, west, east. If we are already at the boundary of the square and choose the action that steps outside the square, the location is unchanged with a reward -1. All the other actions has reward 0. Nevertheless, there are two special grids (1,4) and (3,4) (coordinates

are counted starting from 0 to 4). At (1,4), all actions have reward 10 and the state is transitioned immediately to (1,0). At (3,4), all actions have reward 5 and the state is transitioned immediately to (3,2). Refer to Fig. 7 for a graph illustration of Gridworld.

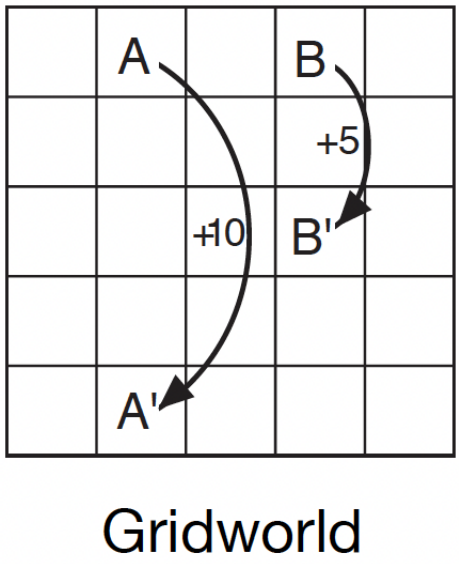

Figure 7: Graph illustration of the Gridworld model

Note that the x-coordinates increase from left to right and the y-coordinates increase from below to above. In other words, (0,0) stands for the most left-down block.

Let's investigate the state value function v_π for this model given the policy π to be the equiprobable policy, i.e. four actions always have $\frac{1}{4}$ of being taken at each state. By adopting the Monte Carlo method mentioned above, we get the estimate for state value function v_π in Fig. 8.

	0	1	2	3	4
0	-1.856261	-0.978066	0.042301	1.521796	3.311155
1	-1.344770	-0.435928	0.740045	2.994327	8.790195
2	-1.226271	-0.354464	0.675206	2.247773	4.421236
3	-1.421055	-0.587406	0.352294	1.902699	5.314808
4	-1.976971	-1.188604	-0.409326	0.544080	1.492610

Figure 8: Monte Carlo estimate for state value function of the Gridworld model

1000 times of simulations have been done to form the estimates. Within each simulation, state, action and reward are being generated until time 20000. The discount rate $\gamma = 0.9$.

By comparing this matrix with the Figure 3.2 in the book, we conclude that the Monte Carlo estimation works pretty well. Let's first try to explain the structure of v_π. As can be expected, the entry at (1,4) and (3,4) has the highest state values among all states, which is natural since the rewards for reaching these two states are the

only positive rewards in the model. As expected, the states next to (1,4) and (3,4) also benefit from the fact that they have larger probabilities of transiting into these two states. When it comes to states far away from these two high-value states and next to the boundary (like column 0), the value is the lowest since it's very likely for these states to go outside the boundary and receive reward -1. However, the interesting is that (1,4) is having a value slightly lower than its reward 10 and (3,4) is having a value slightly higher than its reward 5. This is due to the fact that although (1,4) has a very high immediate reward, the next state would be (1,0), which is a state on the boundary, a low-value state. For (3,4), the next state must be (3,2), a much better state, not on the boundary and closer to these high-value states.

Secondly, let's try to verify that the Bellman equation holds for such state value function

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{79}$$

Let's first pick out the state $s = (2, 2)$ and it's four neighboring states to see whether the Bellman equation holds.

$$LHS = 0.6752 \tag{80}$$

$$RHS = \frac{1}{4} \times 0.9 \times [v_\pi(s' = (2, 3)) + v_\pi(s' = (2, 1)) + v_\pi(s' = (3, 2)) + v_\pi(s' = (1, 2))] \tag{81}$$

$$= \frac{0.9}{4} \times (2.25 - 0.35 + 0.35 + 0.74) \tag{82}$$

$$= 0.67275 \tag{83}$$

the results on two sides are very close to each other, showing that Bellman equation works.

Let's verify the equation for another state (0,0) on the boundary:

$$LHS = -1.8563 \tag{84}$$

$$RHS = \frac{1}{4} \times [0.9 \times (v_\pi(s' = (0, 1)) + v_\pi(s' = (1, 0)) + v_\pi(s' = (0, 0)) + v_\pi(s' = (0, 0))) - 2] \tag{85}$$

$$= -1.859 \tag{86}$$

still quite close.

At last, verify the equation for (1,4), the high-value state.

$$LHS = 8.79 \tag{87}$$

$$RHS = 10 + 0.9 \times v_\pi(s' = (1, 0)) \tag{88}$$

$$= 8.794 \tag{89}$$

still holds.

Last but not least, we would like to mention that Gridworld is a continuing task since no specific ending criterion

of the task is assumed. As a result, adding a same constant to all rewards won't affect the relative values of any states under any policies. However, this does not hold for episodic tasks since episodic tasks may have different ending time under different policies, resulting in different discounted values for such constant.

A Proof for the Uniqueness of the Solution to Bellman Equations

Let's first consider the Bellman equation for state value function, write it in the matrix form and assume that there are altogether n different states $s_1, ..., s_n$, the coefficient matrix is

$$A = \begin{bmatrix} \gamma \sum_a \pi(a|s_1)p(s_1|s_1,a) - 1 & \gamma \sum_a \pi(a|s_1)p(s_2|s_1,a) & ... & \gamma \sum_a \pi(a|s_1)p(s_n|s_1,a) \\ \gamma \sum_a \pi(a|s_2)p(s_1|s_2,a) & \gamma \sum_a \pi(a|s_2)p(s_2|s_2,a) - 1 & ... & \gamma \sum_a \pi(a|s_2)p(s_n|s_2,a) \\ & ... & & \\ \gamma \sum_a \pi(a|s_n)p(s_1|s_n,a) & \gamma \sum_a \pi(a|s_n)p(s_2|s_n,a) & ... & \gamma \sum_a \pi(a|s_n)p(s_n|s_n,a) - 1 \end{bmatrix} \tag{90}$$

where $p(s'|s,a) = \mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a)$. Then notice the fact that

$$\gamma \sum_a \pi(a|s_1)p(s_2|s_1,a) + ... + \gamma \sum_a \pi(a|s_1)p(s_n|s_1,a) \tag{91}$$

$$= \gamma \sum_a \pi(a|s_1)[1 - p(s_1|s_1,a)] \tag{92}$$

$$= \gamma - \gamma \sum_a \pi(a|s_1)p(s_1|s_1,a) \tag{93}$$

$$< 1 - \gamma \sum_a \pi(a|s_1)p(s_1|s_1,a) \tag{94}$$

since $\gamma \in (0,1)$. This shows that the matrix A is strictly diagonally dominant. A simple application of the Gershgorin circle theorem then conclude that A must be invertible. As a result, we conclude that **the state value function v_π is the unique solution to its Bellman equation given policy π and MDP dynamics**.

Optimal Policy and Optimal Value Functions

Now let's take a look to the properties of optimal value functions. State value function v_π naturally induces a partial ordering on the set of all policies defined as

$$\pi \leq \pi' \iff \forall s, v_\pi(s) \leq v_{\pi'}(s) \tag{95}$$

The interesting fact is that the **optimal policy** π_*, the policy better than or equal to any other policy, must exist, though not necessarily unique (easy to raise counterexamples).

We first accept this fact and define the **optimal state value function** as the pointwise maximum w.r.t. all policies at each state.

$$\forall s, v_*(s) \overset{def}{=} \max_\pi v_\pi(s) \tag{96}$$

and similarly the **optimal action value function**

$$\forall s, a, q_*(s,a) \overset{def}{=} \max_{\pi} q_\pi(s,a) \tag{97}$$

The optimal policy can be constructed easily as a splicing of different "good" policies:

$$\forall s, \pi_* \overset{def}{=} \arg\max_{\pi} v_\pi(s) \tag{98}$$

For each state, the distribution $\pi_*(a|s)$ is chosen to be the one that achieves the maximum of the state value function at state s. To verify whether such construction satisfies the former definition:

$$\forall s, \pi, v_{\pi_*}(s) \geq v_\pi(s) \tag{99}$$

$$\forall \pi, \pi_* \geq \pi \tag{100}$$

NOTE: The construction of such optimal policy π_* actually only depends on v_π, which is accessible regardless of the existence of π_*, so we are not doing a circulation proof here.

There's no doubt that by definition, the optimal state value function is just the state value function of the optimal policy.

$$v_* = v_{\pi_*} \tag{101}$$

We would also expect that the optimal action value function is just the action value function of the optimal policy. The following provides a proof for this fact.

Recall the property that

$$q_\pi(s,a) = \sum_{r,s'} (r + \gamma v_\pi(s')) p(s', r|s, a) \tag{102}$$

gives the following equation by taking maximum w.r.t. π on both sides

$$q_*(s,a) = \sum_{r,s'} r \cdot p(s', r|s, a) + \gamma \max_{\pi} \sum_{r,s'} v_\pi(s') p(s', r|s, a) \tag{103}$$

$$\leq \sum_{r,s'} r \cdot p(s', r|s, a) + \gamma \sum_{r,s'} \max_{\pi} v_\pi(s') p(s', r|s, a) \tag{104}$$

$$= \sum_{r,s'} (r + \gamma v_{\pi_*}(s')) p(s', r|s, a) \tag{105}$$

$$= q_{\pi_*}(s,a) \tag{106}$$

Note that by definition, q_* is the pointwise maximum of q_π, so $q_*(s,a) \geq q_{\pi_*}(s,a)$. Conclude

$$q_* = q_{\pi_*} \tag{107}$$

These two properties seem to be trivial but are actually important since they tell us that v_*, q_* can be achieved by the optimal policy π_*, i.e. v_*, q_* are tight upper bounds of v_π, q_π.

Now that it's clear to us that v_*, q_* are value functions for the optimal policy, they have to follow the Bellman equations mentioned above. Take $\pi = \pi_*$ to get the following:

$$v_*(s) = \sum_a \pi_*(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{108}$$

$$q_*(s,a) = \sum_{r,s'} p(s', r|s, a)[r + \sum_{a'} \gamma \pi_*(a'|s') q_*(s', a')] \tag{109}$$

These are called Bellman optimality equations. However, they are not that useful since π_* still appears in these equations and there's no way for us to directly figure out what π_* is. Therefore, we hope to do some transformations to the Bellman optimality equations so that there's no reference to any specific policy.

First notice the relationship between v_* and q_*:

$$v_*(s) = \max_\pi v_\pi(s) \tag{110}$$

$$= \max_\pi \sum_a \pi(a|s) q_\pi(s,a) \tag{111}$$

$$\leq \max_\pi \sum_a \pi(a|s) q_*(s,a) \tag{112}$$

$$= \max_a q_*(s,a) \tag{113}$$

here the last equation is due to the fact that $\sum_a \pi(a|s) = 1, \pi(a|s) \geq 0$, so for any policy π, the sum $\sum_a \pi(a|s) q_*(s,a)$ can't be greater than $\max_a q_*(s,a)$ and that such upper bound can be attained. To see why $v_*(s) \geq \max_a q_*(s,a)$ holds, let's first state the **policy improvement theorem**. This theorem works for deterministic policies and gives a way to improve a policy w.r.t. its state value. For any deterministic policy π, i.e. $\pi(a|s)$ can only take values as either 0 or 1, $\pi(s)$ denotes the action to take with probability 1 at state s.

Theorem 1. *If π, π' are any two deterministic policies and $\forall s, q_\pi(s, \pi'(s)) \geq v_\pi(s)$, then $\forall s, v_{\pi'}(s) \geq v_\pi(s)$. Moreover, if the condition is a strict inequality, then the conclusion is also a strict inequality.*

Proof. First let's state a fact that $\mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = \pi'(s)] = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$. It's due to the fact that policy π' at state s would result in a deterministic action of $\pi'(s)$.

$$\mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] = \sum_a \pi'(a|s) \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \tag{114}$$

$$= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = \pi'(s)] \tag{115}$$

$$\forall s, v_\pi(s) \leq q_\pi(s, \pi'(s)) \tag{116}$$
$$= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \tag{117}$$
$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \tag{118}$$
$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \tag{119}$$
$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) \Big| S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \Big| S_t = s] \tag{120}$$
$$= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \Big| S_t = s] \tag{121}$$
$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \Big| S_t = s] \tag{122}$$
$$\leq ... \tag{123}$$
$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \Big| S_t = s] \tag{124}$$
$$= \mathbb{E}_{\pi'}[G_t | S_t = s] = v_{\pi'}(s) \tag{125}$$

It's an application of the law of iterated expectation and the most important step is to transform the terms into the expectation w.r.t. π'.

□

As a result, set $\pi = \pi_*$, a deterministic optimal policy (if exist). We can find out that $\forall s, a, q_{\pi_*}(s, a) \leq v_{\pi_*}(s)$. Otherwise, there always exists a strict improvement of the optimal policy π which is deterministic, a contradiction with the definition of the optimal policy. This proves the fact that $\forall s, \max_a q_*(s, a) \leq v_*(s)$.

So the last question left is: why will there always be a deterministic optimal policy? Since optimal policy always exists, take π_* as one of them so we can have access to v_*, q_*. Construct policy π' as

$$\pi'(s) = \arg\max_a q_*(s, a) \tag{126}$$

Under such a deterministic policy, the action that results in the greatest action value is always taken for the current state. From the policy improvement theorem, we know that π' is better than any other deterministic policies. However, optimal policy π_* can be seen as a convex combination of some deterministic policies, so $v_{\pi'} \geq v_*$. As a result, $v_{\pi'} = v_*$ and π' is also an optimal policy. That ends the whole proof. (actually one can see that policy improvement theorem holds for any policy π even if it's not deterministic)

Bellman Optimality Equations

The **compact form of Bellman optimality equation** we have concluded is just

$$v_*(s) = \max_a q_*(s, a) \tag{127}$$

This equation does not necessarily hold if the $*$ is replaced by a general policy π and it discloses the structure of the optimal value functions. From this compact form we can infer the frequently used **Bellman optimality equations**.

$$
\begin{aligned}
v_*(s) &= \max_a q_*(s,a) && (128)\\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] && (129)\\
&= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] && (130)\\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] && (131)
\end{aligned}
$$

which is a non-linear equation w.r.t. $v_*(s)$ with no relevance to any policy. Such Bellman optimality equation can be solved theoretically knowing the MDP dynamics and the discount rate but it's always hard to solve practically.

Similarly, the Bellman optimality equation for action value function is:

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] && (132)\\
&= \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] && (133)\\
&= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')] && (134)
\end{aligned}
$$

Bellman Optimality Equations as Fixed Point Equation for Contraction Mapping

Just to mention, we would expect that the Bellman optimal equation for state value function has **a unique solution**, since optimal policy π_* always exists and different optimal policies would share the same $v_*(s)$. To see this point from another point of view, we can make use of the contraction mapping theorem (Banach fixed point theorem) to conclude.

Recall the equation

$$
\begin{aligned}
v_*(s) &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] && (135)\\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] && (136)
\end{aligned}
$$

and organize it in the fixed point form

$$
\begin{aligned}
v_*(s) &= T v_*(s) && (137)\\
T &: \mathbb{R}^{|\mathscr{S}|} \to \mathbb{R}^{|\mathscr{S}|} && (138)\\
Tv(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s, A_t = a] && (139)
\end{aligned}
$$

In order to prove the existence and uniqueness of the fixed-point, the contraction mapping theorem immediately

comes to our mind. As a result, we only have to find a metric on $\mathbb{R}^{|\mathscr{S}|}$ (this is a finite-dimensional space since the state value function can be seen as a finite dimensional vector) such that T is a contraction mapping. Luckily, such metric can be induced by the infinity norm $||\cdot||_\infty$. The next lemma proves that such operator T is actually a contraction mapping under the metric induced by $||\cdot||_\infty$.

Lemma 1. *Consider metric space* $\left(\mathbb{R}^{|\mathscr{S}|}, ||\cdot||_\infty\right)$*, and the operator* T *defined above, then* $\exists 0 \leq k < 1$ *such that* $\forall v, v' \in \mathbb{R}^{|\mathscr{S}|}, ||Tv - Tv'||_\infty \leq k||v - v'||_\infty$.

Proof.

$$||Tv - Tv'||_\infty = \sup_s |Tv(s) - Tv'(s)| \tag{140}$$

$$= \sup_s |\max_a \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s, A_t = a] - \max_a \mathbb{E}[R_{t+1} + \gamma v'(S_{t+1})|S_t = s, A_t = a]| \tag{141}$$

$$\leq \sup_s \max_a |\mathbb{E}[\gamma v(S_{t+1})|S_t = s, A_t = a] - \mathbb{E}[\gamma v'(S_{t+1})|S_t = s, A_t = a]| \tag{142}$$

$$= \sup_s \max_a \gamma \mathbb{E}[|v(S_{t+1}) - v'(S_{t+1})|\Big|S_t = s, A_t = a] \tag{143}$$

$$\leq \gamma \sup_s \max_a ||v - v'||_\infty \tag{144}$$

$$= \gamma ||v - v'||_\infty \tag{145}$$

□

As a result, the contraction mapping theorems holds and the Bellman optimality equation for state value function has a unique solution $v_*(s)$, i.e. the Bellman optimality equation is the characterization of the optimal value functions.

Summary

It should be clear for now that when we are solving a RL problem, we do not care about v_π and q_π for a specific policy π since there are uncountably many policies π. In other words, although for any fixed policy π Bellman equations offer an easy way to solve out v_π, q_π since the equations are linear, it's unrealistic for us to find out the optimal value functions v_*, q_* by applying their definitions. Instead, the non-linear Bellman optimality equation is of our true concern. It has nothing to do with policy and only requires knowing the MDP dynamics p.

So what shall we do after knowing/estimating the optimal value functions v_*, q_*? If we know q_*, the best actions to pick at current state S_t are just the actions a such that $q_*(S_t, a)$ is maximized. If we know v_*, the best actions to pick at current state S_t are all the actions a such that the maximum in the Bellman optimality equation $v_*(S_t) = \max_a \sum_{s',r} p(s', r|S_t, a)[r + \gamma v_*(s')]$ is attained. It's easy to see that the two ways of selecting best actions give the same set of actions and any convex combination of those actions gives an optimal policy π_* (that's why π_* is not unique).

In brief, what we have done above can be concluded into following items:

- Estimating v_*, q_* is the core task in RL

- v_*, q_* satisfies non-linear Bellman optimality equations that can be solved if MDP dynamics p is known
- After getting estimates of v_*, q_*, the RL problem is solved since the best action at each turn can be figured out

While the concepts of MDP are all clear, many problems remain unsolved. Directly solving Bellman optimality equations by fixed point iteration is always not practically feasible since MDP dynamics are always unknown, especially in cases where the reward itself is random given the states and actions. Nevertheless, the method won't work for an MDP with too many states even if p is known. In fixed point iteration schemes, the vector has its length equal to the number of states, and for games like chess, there are at least 3^{361} number of states! As a result, further studies are necessary to develop various methods for the estimation of optimal policy/optimal value functions.

Gridworld Model as an Example

Let's see how optimal value functions are found in the **Gridworld** model and figure out the best actions at each state. Although no methods have been introduced to estimate optimal value function v_*, we have mentioned above that **the solution to the Bellman optimality equation can be viewed as the fixed point of a contraction mapping**. So why not use fixed point iteration to approximate v_*?

An initial value $v_0(s)$ would be constructed, and the fixed point iteration goes like:

$$\forall s \in \mathscr{S}, v_{n+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_n(s')] \tag{146}$$

It would be expected that

$$\forall s \in \mathscr{S}, v_n(s) \to v_*(s) \ (n \to \infty) \tag{147}$$

Refer to Fig. 9 for the experiment on Gridworld for the approximation result for v_*.

	0	1	2	3	4
0	14.419349	16.021499	17.801666	19.779673	21.977414
1	16.021499	17.801666	19.779673	21.977414	24.419349
2	14.419349	16.021499	17.801666	19.779673	21.977414
3	12.977414	14.419349	16.021499	17.801666	19.419349
4	11.679673	12.977414	14.419349	16.021499	17.477414

Figure 9: Fixed point iteration approximation for the optimal state value function v_* of the Gridworld model

The discount rate $\gamma = 0.9$. It's natural for (1,4) to have the highest optimal state value. Fixed point iteration of Bellman optimality equation is guaranteed to converge because of the contraction mapping property we have proved.

Based on the optimal state value function derived, it's easy for us to figure out the best actions to take at each state as stated above. The results are demonstrated in Fig. 10 (this is exactly the same as Figure 3.5 in the textbook).

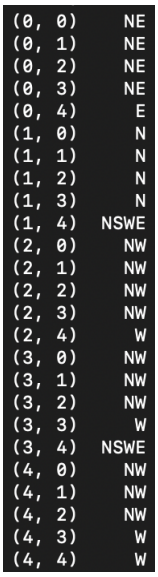

Figure 10: Best actions at each state of the Gridworld model

The first column exhibits the 25 states and the second column exhibits the best actions at that state.
N stands for North, i.e. moving upward; S stands for South, i.e. moving downward; W stands for West, i.e. moving leftward; E stands for East, i.e. moving rightward.
Multiple best actions may exist at some states, e.g. at (1,4) all four actions have the same consequence, so all four actions are the best actions at state (1,4).

Another MDP Model to Illustrate the power of Bellman Optimality Equations

Let's look at exercise 3.22 for a different MDP, shown in Fig. 11. The only two policies for choice are π_l, π_r, the policy that always choose to go left and the policy that always choose to go right. For different values of γ, how does the optimal policy change?

Let's write out the Bellman optimality equation for this model first. The top state is state s_1, the state at the left lower corner is state s_2 and the state at the right lower corner is state s_3. When at s_2 or s_3, only one action can be chosen and when at state s_1, two actions can be chosen from.

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{148}$$

Let's first write out the Bellman optimality equations for states s_2, s_3:

$$v_*(s_2) = \gamma v_*(s_1) \tag{149}$$

$$v_*(s_3) = 2 + \gamma v_*(s_1) \tag{150}$$

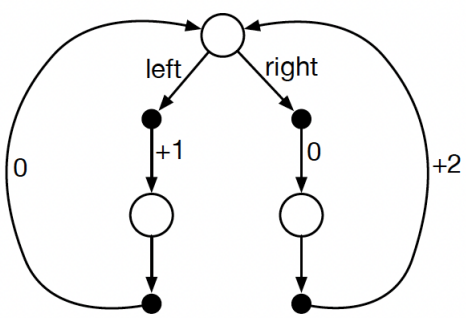

Figure 11: Another MDP

There's 3 states, but decisions only have to be made at the top state, where there are two actions to choose from. Deterministic rewards are labelled above edges. There's only two deterministic policies π_l, π_r.

The Bellman optimality equations for state s_1 has a maximum inside:

$$v_*(s_1) = \max\{1 + \gamma v_*(s_2), \gamma v_*(s_3)\} \tag{151}$$

Now, solve the Bellman optimality equations to find that the critical value of γ is γ_* such that $1 + \gamma_* v_*(s_2) = \gamma_* v_*(s_3)$.

$$1 + \gamma_*^2 v_*(s_1) = 2\gamma_* + \gamma_*^2 v_*(s_1) \tag{152}$$

$$\gamma_* = \frac{1}{2} \tag{153}$$

This is telling us that when the discount rate $\gamma = \frac{1}{2}$, both actions achieves the maximum in the Bellman optimality equations, so both π_l and π_r are optimal policies. When $0 \leq \gamma < \frac{1}{2}$, only the action of going left achieves the maximum in the Bellman optimality equations, so only π_l is the optimal policy. When $\frac{1}{2} < \gamma < 1$, only π_r is the optimal policy.

Dynamic Programming (DP) Methods

The DP methods are direct and natural extensions of the properties we have proved above. In this section regarding DP methods, it's assumed that the MDP dynamics p is always known and that there are not too many states (so sweeping through all states would be practically feasible).

Policy Evaluation

The policy evaluation task refers to the task that we hope to get value functions v_π, q_π given a policy π. Let's first deal with the task of getting v_π from scratch.

It's natural to immediately recall the Bellman equations

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{154}$$

We are happy to see that this equation is a linear system w.r.t. $v_\pi(s)$ and it has a unique solution. This equation certainly can be organized in its matrix form and be solved easily. However, the Bellman equation here provides us with a natural iteration scheme of equation solving.

Note that this equation is actually a fixed-point equation for $v_\pi(s)$, so why not use fixed point iteration? Let's first do some analysis to find out whether fixed point iteration is a good scheme for this problem. (Similar to what we have done for Bellman optimality equations)

Set the operator $T : \mathbb{R}^{|\mathscr{S}|} \to \mathbb{R}^{|\mathscr{S}|}$ to be that

$$Tv(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')] \tag{155}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v(S_{t+1})|S_t = s] \tag{156}$$

The Bellman equation becomes

$$v_\pi(s) = Tv_\pi(s) \tag{157}$$

Now view T as a linear operator on metric space $\left(\mathbb{R}^{|\mathscr{S}|}, ||\cdot||_\infty\right)$,

$$||Tv(s) - Tv'(s)||_\infty = \gamma ||\mathbb{E}_\pi[v(S_{t+1}) - v'(S_{t+1})\Big|S_t = s]||_\infty \tag{158}$$

$$\leq \gamma ||v - v'||_\infty \tag{159}$$

Once again it's a contraction mapping since $0 \leq \gamma < 1$, that's why the fixed point iteration will always converge. The **iterative policy evaluation** is stated as:

$$v_{n+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_n(s')] \tag{160}$$

and we would expect to see

$$\forall s, v_n(s) \to v_\pi(s) \ (n \to \infty) \tag{161}$$

So what about the action value function q_π? Can we still use the similar method and will the convergence still be guaranteed? Recall the Bellman equation for q_π:

$$q_\pi(s,a) = \sum_{r,s'} p(s', r|s, a)[r + \sum_{a'} \gamma \pi(a'|s') q_\pi(s', a')] \tag{162}$$

with the **iterative policy evaluation**:

$$q_{n+1}(s,a) = \sum_{r,s'} p(s', r|s, a)[r + \sum_{a'} \gamma \pi(a'|s') q_n(s', a')] \tag{163}$$

Set the operator $T : \mathbb{R}^{|\mathscr{S}|} \times \mathbb{R}^{|\mathscr{A}|} \to \mathbb{R}^{|\mathscr{S}|} \times \mathbb{R}^{|\mathscr{A}|}$ to be that

$$Tq(s,a) = \sum_{r,s'} p(s', r|s, a)[r + \sum_{a'} \gamma \pi(a'|s') q(s', a')] \tag{164}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{a'} \pi(a'|S_{t+1}) q(S_{t+1}, a')|S_t = s, A_t = a] \tag{165}$$

The Bellman equation becomes

$$q_\pi(s,a) = Tq_\pi(s,a) \tag{166}$$

Now view T as a linear operator on metric space $\left(\mathbb{R}^{|\mathscr{S}|\times|\mathscr{A}|}, ||\cdot||_\infty\right)$,

$$||Tq(s,a) - Tq'(s,a)||_\infty = \gamma ||\mathbb{E}_\pi[\sum_{a'} \pi(a'|S_{t+1})[q(S_{t+1}, a') - q'(S_{t+1}, a')]\Big| S_t = s, A_t = a]||_\infty \tag{167}$$

$$\leq \gamma ||q - q'||_\infty \mathbb{E}_\pi[\sum_{a'} \pi(a'|S_{t+1})\Big| S_t = s, A_t = a] \tag{168}$$

$$= \gamma ||q - q'||_\infty \tag{169}$$

For the same reason, T is a contraction mapping, convergence is ensured and we would expect to see

$$\forall s, a, q_n(s,a) \to q_\pi(s,a) \ (n \to \infty) \tag{170}$$

One last remark is that when setting initial values for the iteration, if the task is an episodic task, the terminal state must be given value 0 (means that no reward will be further provided). For all states in continuing tasks or normal states in episodic tasks, this restriction does not exist.

Policy Iteration

As stated at the section of the policy improvement theorem, for any policy π, $\pi'(s) = \arg\max_a q_\pi(s,a)$ is always a deterministic policy as the improvement of π and it's thus called the **greedy policy w.r.t.** π. In the case where π is already the optimal policy, $\pi'(s) = \arg\max_a q_*(s,a)$ has also been proved to be the optimal policy, though not necessarily the same. As a result, this provides a way to iteratively improve the policy until the policy is optimal.

For any initial policy π_0, policy evaluation is applied to figure out v_{π_0}. After that, the greedy policy π_1 is constructed w.r.t. q_{π_0} as an improvement of π_0. Iteratively, the policy iteration ensures that the policy is always being strictly improved until it becomes the optimal policy and the iteration stops. There's one question left: how to figure out the greedy policy w.r.t. π from v_π instead of q_π? The answer is easy since q_π can always be represented by v_π and the dynamics p as $q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$.

To conclude, the **policy iteration for** v_π proceeds like:

- For the current policy π_k
- Policy evaluation to get v_{π_k}: $v_{n+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_n(s')]$ as a fixed point iteration based on the Bellman equations, have to wait for the convergence of v_n to v_{π_k}
- Policy improvement to get a new improved deterministic policy π_{k+1} with $\pi_{k+1}(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$
- Repeat the operations above until convergence $\pi_k \to \pi_*$ happens. Note that setting the stopping criteria as $\pi_k = \pi_{k+1}$ is not sufficient since it's possible for the optimal policy to jump back and forth between two different optimal policies. Make use of the uniqueness of v_* and a good stopping criteria would be $||v_{\pi_k} - v_{\pi_{k+1}}||_\infty < \varepsilon$.

The convergence of such algorithm is ensured completely by the policy improvement theorem. If π is not optimal, the greedy policy w.r.t. π must be a strict improvement.

Actually, q_π can also be used to conduct policy iteration and the frame is very much alike. The **policy iteration for** q_π proceeds like:

- For the current policy π_k
- Policy evaluation to get q_{π_k}: $q_{n+1}(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \sum_{a'} \gamma \pi(a'|s') q_n(s',a')]$ as a fixed point iteration based on the Bellman equations, have to wait for the convergence of q_n to q_{π_k}
- Policy improvement to get a new improved deterministic policy π_{k+1} with $\pi_{k+1}(s) = \arg\max_a q_\pi(s,a)$
- Repeat the operations above until convergence $\pi_k \to \pi_*$ happens. Note that setting the stopping criteria as $\pi_k = \pi_{k+1}$ is not sufficient since it's possible for the optimal policy to jump back and forth between two different optimal policies. Make use of the uniqueness of q_* and a good stopping criteria would be $\forall s, ||q_k(s,\cdot) - q_{k+1}(s,\cdot)||_\infty < \varepsilon$.

Value Iteration

Actually, we have already made use of value iteration techniques in the Gridworld example for Bellman optimality equations. It's just a fixed point iteration of the non-linear Bellman optimality equations and we will directly get v_*. The algorithm proceeds like:

$$\forall s \in \mathscr{S}, v_{n+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_n(s')] \tag{171}$$

The convergence is ensured by the property of contraction mapping proved above. Similarly, q_* can be derived according to its Bellman optimality equations:

$$\forall s \in \mathscr{S}, a \in \mathscr{A}(s), q_{n+1}(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_n(s', a')] \tag{172}$$

The advantage of value iteration method is that it costs less time than policy iteration generally since policy iteration has to do policy evaluation within the loop.

Car Rental Example

Two locations for a rental company. When a customer comes, if a car is available, rent out and get 10 (dollars), if no car available, lose business. Cars are available the day after returned. Overnight movings between two locations are allowed, pay 2 per car. The numbers of rental requests per day in the first and second locations are $P(3), P(4)$ random variables, and the number of returns per day are $P(3), P(2)$ random variables. There can be no more than 20 cars at each location at any time (additional cars just disappear), overnight movings for at most 5 cars.

First let's turn this problem into a MDP by specifying the states, actions and rewards. The rewards are just the amounts of profit and the time is counted by days. The states are tuples consisting of the number of cars in the first location and the number of cars in the second location at the end of the day before overnight moving happens, i.e. $\mathscr{S} = \{(s_1, s_2)|s_1, s_2 \in \mathbb{Z}, 0 \leq s_1, s_2 \leq 20\}$. The actions are the decisions made about how many cars (net) to move from one location to another, i.e. $\mathscr{A} = \{-5, -4, ..., -1, 0, 1, ..., 5\}$ where 1 means 1 car is moved from the first to the second location and negative integers means moving in the opposite direction. If the action space is written in a form with a dependency on the current state, $\mathscr{A}(s_1, s_2) = \{a|a \in \mathscr{A}, -s_2 \leq a \leq s_1\}$.

The action changes the state is the way that if the current state is $S_t = (S_t^1, S_t^2)$, the action is A_t, the number of rental requests on day t for location i is RT_t^i and the number of returned cars on day t for location i is RE_t^i, then the next state would be affected by overnight movings, next day's rented cars and today's returned cars. On the next morning, there will be

$$\left(\min\left\{S_t^1 - A_t + RE_t^1, 20\right\}, \min\left\{S_t^2 + A_t + RE_t^2, 20\right\}\right) \tag{173}$$

cars in the first and the second locations. As a result, the rental will bring with overall revenues

$$10 \times [\min\left\{\min\left\{S_t^1 - A_t + RE_t^1, 20\right\}, RT_{t+1}^1\right\} + \min\left\{\min\left\{S_t^2 + A_t + RE_t^2, 20\right\}, RT_{t+1}^2\right\}] \tag{174}$$

Note that a number of $-2|A_t|$ would be added to the reward of this turn as the cost of overnight moving. As a result, the reward is

$$R_{t+1} = 10 \times [\min\left\{\min\left\{S_t^1 - A_t + RE_t^1, 20\right\}, RT_{t+1}^1\right\} + \min\left\{\min\left\{S_t^2 + A_t + RE_t^2, 20\right\}, RT_{t+1}^2\right\}] - 2|A_t| \tag{175}$$

and the next state should be

$$RENTED_{t+1}^1 = \min\left\{\min\left\{S_t^1 - A_t + RE_t^1, 20\right\}, RT_{t+1}^1\right\} \tag{176}$$

$$RENTED_{t+1}^2 = \min\left\{\min\left\{S_t^2 + A_t + RE_t^2, 20\right\}, RT_{t+1}^2\right\} \tag{177}$$

$$S_{t+1} = (\min\left\{S_t^1 - A_t + RE_t^1, 20\right\} - RENTED_{t+1}^1, \min\left\{S_t^2 + A_t + RE_t^2, 20\right\} - RENTED_{t+1}^2) \tag{178}$$

This car rental problem is organized as a continuing finite MDP. Although it's presented in the textbook as an example of policy iteration methods, I think it would already be difficult to figure out the MDP dynamics p despite the simplicity of this problem. The difficulty lies in the boundary conditions, e.g. the maximum number of cars allowed, the comparison between available cars and rental requests, and the randomness of rewards and state evolution, e.g. both the rewards and the next state involves Poisson random variables in a non-linear way.

As we have expected, if the model is simple and the dynamics p can be easily derived, e.g. the Gridworld model, DP methods work pretty well and are extremely simple to implement. However, for slightly more complicated problems, it's practically impossible to compute p, and DP methods can do nothing without knowing p.

Gambler's Problem

The gambler's problem is much nicer for DP methods to apply. A gambler gambles on a sequence of coin flips. The game ends if the gambler has no money (fails) or if the gambler has got 100 (wins). For each flip, the gambler decide an integer as the stake. If the coin comes up heads, wins as much as his stake. If the coin comes up tails, lose all stakes. Note that the gambler's stake has to satisfy the condition that the amount of money he owns now plus the stake does not surpass 100.

To specify the problem as a MDP, the reward is NOT the amount of money BUT whether the gambler wins the game (reward is 0 on all transitions and 1 iff gambler wins the game) and the time is the number of turn. The state set would be $\mathscr{S} = \{0, 1, ..., 99, 100\}$ indicating the amount of money that gambler currently owns before putting down the stake. Note that here $0, 100$ are two special terminal states, 0 is always given state value 0 and 100 is always given state value 1 (consistent with the reward setting). The set of actions is $\mathscr{A}(s) = \{0, 1, ..., \min\{s, 100 - s\}\}$.

Under such undiscounted setting, state value function $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\mathbb{I}_{win}|S_t = s] = \mathbb{P}_\pi(win|S_t = s)$ has the meaning of the **probability** of winning given policy π and the current amount of money. The only parameter for this game is p_h, the probability to get heads in a single coin flip.

This game is an **episodic, nondiscounted**, finite MDP and its dynamics $p(s', r|s, a)$ can be easily found. If heads appears, $s' = s + a$ and $r = 1$ together with the end of the game come if and only if $s' = 100$. If tails appears, $s' = s - a$ and $r = 0$, and it's the end of the game if $s' \leq 0$.

$$p(s + a, 0|s, a) = p_h \ (s + a < 100) \tag{179}$$
$$p(s + a, 1|s, a) = p_h \ (s + a = 100) \tag{180}$$
$$p(s - a, 0|s, a) = 1 - p_h \tag{181}$$
$$p(s, 0|s, 0) = 1 \tag{182}$$

The optimal state value function and the optimal policy derived using value iteration method is exhibited in Fig. 12, 13, 14 for different values of p_h. Note that the optimal policy is not necessarily unique, so all possible best actions are printed out for each state.

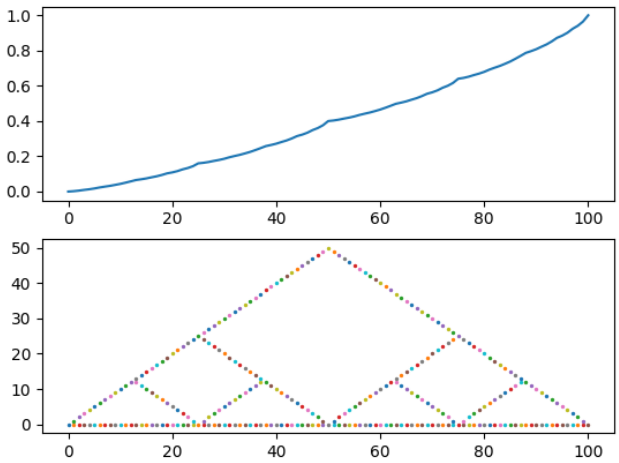

Figure 12: Optimal state value function and optimal policy for gambler's problem with $p_h = 0.4$

For the upper subplot, the x-axis is the state s and the y-axis is the state value $v_*(s)$.
For the lower subplot, the x-axis is the state s and the y-axis is the best action at this state, i.e. the action that achieves maximum in Bellman optimality equation for v_*.

Note that the lower subplot of Fig. 12 contains the deterministic optimal policy exhibited in Figure 4.3 in the textbook. As stated in the earlier context, this enables us to find the set of all optimal policies. (just have to make sure that if $\pi_*(a|s) > 0$ for some state s then action a achieves the maximum in the Bellman optimality equation for v_*)

The interesting fact is that when $p_h < \frac{1}{2}$, the best actions appear like the lower subplot in Fig. 12, while if $p_h = \frac{1}{2}$, all available actions are best actions, and the best actions appear as a left-tailed biased shape if $p_h > \frac{1}{2}$.

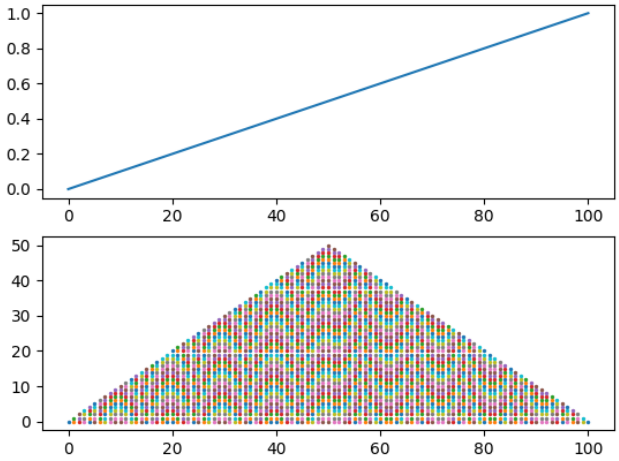

Figure 13: Optimal state value function and optimal policy for gambler's problem with $p_h = 0.5$

For the upper subplot, the x-axis is the state s and the y-axis is the state value $v_*(s)$.
For the lower subplot, the x-axis is the state s and the y-axis is the best action at this state, i.e. the action that achieves maximum in Bellman optimality equation for v_*.

For $p_h = \frac{1}{2}$, the expected return of each gamble is always 0 regardless of the action chosen. As a result, all actions are "fair bets" and there should be no preferences.

When $p_h < \frac{1}{2}$, it's harder to win the bet in each turn, so whenever there's a chance to win the game directly in one turn (when state is no less than 50), it's always the best action. How shall we explain the phenomenon that there may exist multiple best actions for the same state? For example, if the current state is 51, why is action 1 also the best action? The answer might sound weird but by taking action 1, we are hoping to lose the bet this turn in order to get to state 50. After getting to state 50, we are able to take all money as stake and to see whether we will win or lose in a single bet. It's important to understand that when $p_h < \frac{1}{2}$, we hope the game to end as soon as possible since longer time causes the law of large numbers to work more effectively, lowering the possibility to win the game. At this point, it should be easy to understand why for all $p_h < \frac{1}{2}$, the set of best actions are all the same as that in Fig. 12.

To get a little deeper into the optimal policy in Fig. 12, ignoring action 0, whenever there are multiple best actions for a single state, the largest action always aims to win the bet in the turn while the smallest action always aims to lose the bet in the turn. The "critical states" are 87.5, 75, 62.5, 50, 25, 37.5, 12.5. 50 is easy to understand since we are able to bet in a single turn to see the final result of the game. 25 and 75 are also natural since they are midpoints of the intervals $[0, 50], [50, 100]$ and by reaching them, it's possible for us to reach 50, reducing the number of turns in this game as much as possible. A natural question is that why don't we take the midpoint of $[0, 12.5]$ and do further bisections of the intervals? (This is a question worth thinking about, recall that the objective is to end the game as soon as possible, so it's a good idea to do further partitions and compare with the current scheme)

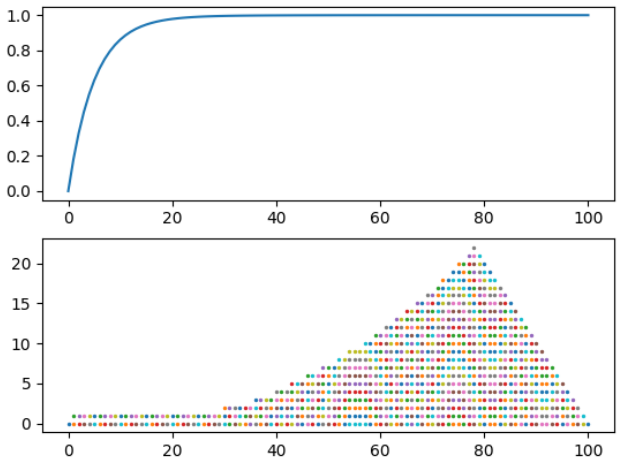

Figure 14: Optimal state value function and optimal policy for gambler's problem with $p_h = 0.55$

For the upper subplot, the x-axis is the state s and the y-axis is the state value $v_*(s)$.
For the lower subplot, the x-axis is the state s and the y-axis is the best action at this state, i.e. the action that achieves maximum in Bellman optimality equation for v_*.

When $p_h > \frac{1}{2}$, it's exactly the opposite. We hope that the game would last longer so the law of large number works. That's why taking action 1 is always the best action for all state. As long as the stake is small enough, law of large numbers finally leads us to the winning of the game. When the state gets larger, we would have more best actions since the ability to take risk also grows. It's natural to expect that different p_h would result in different set of best actions for each state, since for larger p_h, we are more confident that we cannot lose the game and we should be willing to put more money on the desk even if we are not having much money with us for the time being.

Summary

Mainly two simple DP methods: policy iteration and value iteration.

Policy iteration tries to find the optimal policy and optimal state value function by focusing on improving the policy. Firstly, we have to do policy evaluation to find c_π for policy π (fixed point iteration for Bellman equation of v_π), then policy improvement, finding a greedy deterministic policy for v_π is found (the action that achieves maximum of q_π for fixed current state) and updated as the new policy. Actually, it's like competition & cooperation between the policy and the value function. **Generalized policy iteration (GPI)** refers to the policy iteration that enables the the interaction between policy evaluation and policy improvement, i.e. improving the policy before policy evaluation is completely done or evaluating the new policy before policy improvement is completely done. Such methods are expected to work more efficiently and still has the convergence property.

Value iteration is a fixed point iteration of Bellman optimality equation for v_*. After figuring out v_*, the set of best actions is constructed by finding out which of those actions achieve the maximum in the Bellman optimality

equation.

The DP methods are efficient for simple RL problems with small state space and concise dynamics. However, the main drawbacks are that sometimes the dynamics is too complicated to be written out in a clear style and that state space is always large and it's impossible to sweep through all states.

Monte Carlo (MC) Methods

After introducing DP methods, it shall be quite obvious that the advantage of MC methods is that **experience** is the only thing needed. In other words, we don't have to figure out the dynamics p, we don't have to sweep through all states, we just need to construct the described RL process and do simulations. Experience will tell us everything we want to know. Note that since MC can deal with any expectation estimation problems and v_π, q_π are both conditional expectations, MC perfectly fits with the situation in RL.

First-visit MC

Actually, we have talked about using MC to approximate the value function v_π for a certain policy π in the previous context already. Fig. 8 is just produced by applying the every-visit MC method. The visit of an action or a state just means the appearance of an action or a state in the simulation process. A first visit of state s just means the first time a simulation generates state s.

The **first-visit MC** method for estimating v_π proceeds as:

- Do simulations for the RL problem with policy π
- For each state s, if it's the first time to visit state s, i.e. $T \overset{def}{=} \inf\{t|S_t = s\}$, take record of the return G_T at that time T (the simulation has to be stopped before the return can be calculated since the return is a discounted sum of future rewards)
- The sample mean of all G_T recorded is just an estimate for $\mathbb{E}_\pi[G_t|S_t = s] = v_\pi(s)$

The **every-visit MC** method is just recording all visits to state s, getting the returns and taking sample average.

Actually, the first-visit MC is the traditional MC since different observations of the state, action, reward chains are independent and the convergence is ensured by the law of large numbers. However, the every-visit MC also has a guaranteed convergence property.

By adopting a similar scheme, q_π can also be approximated by first-visit/every-visit MC in the sense that the state-action pair (s, a) is kept track of.

Blackjack

The blackjack game aims to get as close to 21 as possible with poker cards. All face cards are 10 and the ace can be 1 or 11. When the game starts, the dealer and the player each gets 2 cards, the dealer has one card facing up and another facing down. If player gets 21, he wins immediately except the case where the dealer also gets 21. The player can choose to hit (draw an additional card) or stick (stop drawing card). The player loses the game as soon as the sum exceeds 21 and when he sticks the dealer begins to do the same thing. The dealer loses the game as soon as the sum exceeds 21 and when he sticks he compares the sum with the player to see whether it's a draw/win/lose.

Blackjack can be formulated as a undiscounted episodic finite MDP. Let's assume that the dealer sticks on any sum greater or equal to 17 otherwise hits (the most conservative strategy), so the problem now becomes an RL problem for the player. Rewards are given based on draw/win/lose to be 0/1/-1 respectively and any transitions

within an episode receive no rewards. Assume there are infinitely many cards and ace is said to be usable if it works as 11 and nonusable if it works as 1. Note that the ace in the dealer's first two cards must work as 11 unless there are two aces that exceeds 21, and all drawn aces of the dealer work as 1. The player, however, can decide what value each ace takes. It's quite obvious that if there is ace in the first two cards of the player, at least one of them should work as 11 since there is no risk of exceeding 21 if the sum is less than 12. (when the sum is 11, if we get an ace, the ace should be nonuseable, so the sum can't exceed 21)

In detail, the state space should be $\mathscr{S} = \{(s, d)|s, d \in \mathbb{N}, 12 \leq s \leq 21, 1 \leq d \leq 10\}$, where s stands for the sum of cards in the player's hand and d stands for the first card the dealer is showing to us. Note that s has minimal value 12 because when the sum is less than 12, we just hit without thinking. The action space contains two actions: $\mathscr{A} = \{hit, stick\}$. Assume we are adopting the policy that the player sticks if and only if the sum is 20 or 21 (a risky strategy). Let's figure out the estimated v_π for this strategy π, refer to Fig. 16 and **??** for details.

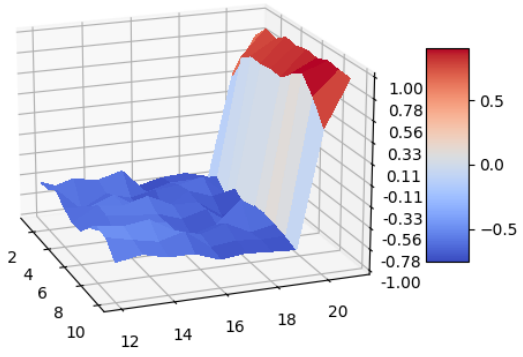

Figure 15: State value function estimated by Monte Carlo

The x-axis stands for the first card of the dealer facing up (1 means an ace), the y-axis stands for the sum of cards in the player's hands, the z-axis stands for the estimated state value $v_\pi(s)$ for such state tuple.
10000 iterations for Monte Carlo estimation, the estimation is coarse.

Some observations can be made based on these two figures. The state value function has a sudden jump when the player's sum is at 20 since the policy is that the player continues hitting unless the sum is no less than 20. As a result, when the player chooses to hit with a sum near 20, it's often the case that he will go busted and lose the game. It can also be seen that there's a drop in the state value where the first card of the dealer is an ace. This is natural since an ace for the dealer is more conditionally likely for him to win immediately (note that the probability of drawing 10 is $\frac{4}{13}$ in Blackjack, and an ace with 10 is a Blackjack). As a result, the moment we see that the dealer has the first card as ace, we are already more likely to lose given this fact.

In the Blackjack example, the first-visit MC actually has no difference from the every-visit MC since for each

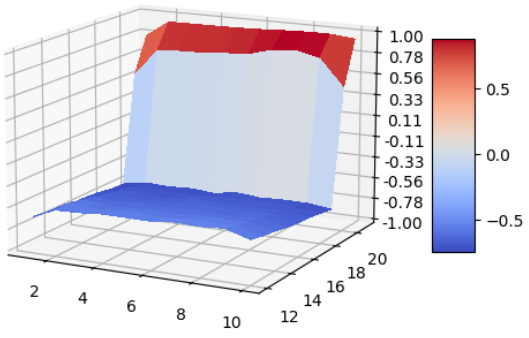

Figure 16: State value function estimated by Monte Carlo

The x-axis stands for the first card of the dealer facing up (1 means an ace), the y-axis stands for the sum of cards in the player's hands, the z-axis stands for the estimated state value $v_\pi(s)$ for such state tuple.
500000 iterations for Monte Carlo estimation, the estimation is pretty good.

game the state is always changing and never repeats (the player's sum is strictly increasing).

Some Tips for MC Methods

We have mentioned above for the MC scheme estimating action value function q_π. The fact is that so far we have always been estimating the state value function or the optimal state value function to find the best action because of its simplicity. However, by recalling the procedure we would find out that searching for best actions based on v_* would require us knowing the MDP dynamics p (the action that achieves the maximum in the Bellman optimality equation for v_*). What a bad news! MC has its advantage that it does not need to know dynamics p but now we are depending on p once more.

However, we immediately notice that **the best action can be figured out with an estimate for q_* without knowing dynamics** p. For each state s, the best action is just the action a such that $q_*(s, a)$ is the largest. Besides, the MC scheme can be easily extended for the estimation of q_π by recording both the states and the actions. That's the exact reason why we will be focusing on estimating the action value function in the following context with MC methods.

The another problem is that when evaluating value functions, there may be possibility that there's no observations of a certain action when the policy only places a very little probability on such action. That is, we have to **maintain exploration** as mentioned above to ensure that there is enough knowledge for each state-action pair. One way is to adopt the **exploring start**, i.e. each state-action pair has a positive probability of being the initial pair of state-action. By doing this, each pair is guaranteed to be visited infinitely often if we are doing infinitely

many simulations.

MC Control

Now that we want to estimate q_* in order to solve the RL problem, a GPI thought can be combined with MC. Since each GPI has the policy evaluation and the policy improvement parts, MC will help us with policy evaluation (which is done by DP in the former section), and a greedy deterministic policy is still selected as the improved policy.

So now the algorithm proceeds like:

- Fix a policy π
- For each generated episode $S_0, A_0, R_1, ...$, calculate the returns at each time, then find out the time of the first visit to state-action pair (s, a) and record all returns at such time
- Do the simulation above for many times as MC simulations to estimate $q_\pi(s, a)$ for each state s and action a
- Construct the greedy deterministic policy $\pi'(s) = \arg\max_a q_\pi(s, a)$ as an improvement of π
- Iterate until $\pi \to \pi_*, q_\pi \to q_*$

Some obvious problems with this algorithm are that: (i): the algorithm generates deterministic policies as improvements, so there's a large possibility that some states or actions are never visited in the MC simulation (ii): the algorithm costs to much time by performing the complete MC simulation for each time of policy evaluation, it may take a long time to converge.

The first problem is easily solved by adding an exploring start, i.e. assign positive probability for each state-action pair to appear as the initial state and the initial action. The second problem requires inspirations coming from GPI. Since policy evaluation now costs much time, why don't we improve the policy before policy evaluation is completed? An existing example would be the value iteration method. Although we are viewing value iteration as fixed-point iteration in the previous context, let's recall the value iteration:

$$v_{n+1}(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_n(s')] \tag{183}$$

and the policy evaluation as DP:

$$v_{n+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_n(s')] \tag{184}$$

It's natural to find that if we replace the policy π in the policy evaluation by a greedy deterministic policy $\pi'(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r+\gamma v_n(s')]$, it just becomes the value iteration. In detail, the maximized expression $\sum_{s',r} p(s', r|s, a)[r + \gamma v_n(s')]$ is just the expression within a single iteration in the policy evaluation. To sum up, the **value iteration can be viewed as GPI** because it evaluates and improves the policy at the same time. To be specific, for policy π, policy evaluation is done only for **one step towards** v_π (so the error is still large, since such evaluation would require a period of time to converge) and a policy improvement follows, finding the greedy policy

w.r.t. such poorly estimated "value function". By recognizing that value iteration is actually one-step evaluation plus one-step improvement, it would be surprising that such method actually is a fixed point iteration and converges really quickly. However, same strategy can be applied with policy evaluation techniques replaced by MC simulation.

As a result, **the first-visit MC ES for estimating** π_* goes like (ES for exploring start and policy evaluation is done for only one step for each policy):

- Exploring start, each state-action pair has positive probability of being selected as the initial state-action
- For a fixed policy π, generate episode $S_0, A_0, R_1, ...$, calculate the returns at each time, then find out the time of the first visit to state-action pair (s, a) and record all returns at such time in a list
- Update the estimate for $q_\pi(s, a)$ as the average of all values in the list (only one-step evaluation since only one return value for (s, a) is added to the list)
- Construct the greedy deterministic policy $\pi'(s) = \arg\max_a q_\pi(s, a)$ as an improvement of π
- Iterate until $\pi \to \pi_*, q_\pi \to q_*$

The difference is that originally for each policy π, a complete MC procedure is used to **accurately** evaluate q_π and now for each policy, a single episode is generated to **push the estimate for** q_π **toward the true** q_π **a little bit**. Note that here a list to store the return value for each state-action pair (s, a) is actually not necessary since what we care about is just the mean and it could be incrementally recorded, i.e. let $m_t(s, a)$ denote the sample mean of the return of the first-visit to (s, a) up to and include time t, $N_t(s, a)$ denote the number of appearances of the first-visit to (s, a) up to and include time t and $G(s, a)$ denote the return of the first-visit to (s, a) newly found. Then it's obvious that

$$m_t(s, a) = \frac{N_{t-1}(s, a)}{N_t(s, a)} m_{t-1}(s, a) + \frac{1}{N_t(s, a)} G(s, a) \tag{185}$$

This MC ES algorithm is ensured not to converge to any suboptimal policy. Otherwise, the estimate for q_π would be very close to the action value function of that suboptimal policy. However, the greedy policy is proved to be a strict improvement if the policy is not yet optimal, providing a better policy. The convergence property of MC ES has not been proved yet, which means that may exist possibility that this algorithm does not converge.

Policy Gradient Method

Policy gradient methods focus on updating the policy π instead of forming any estimation on the value function. This is especially useful when one has to deal with a very large state space and action space when the estimation of value function becomes infeasible (at least one estimation has to be made for each state). In order to form a large enough family of policies to explore, one always parametrize the policy as $\pi(a|s,\theta)$ with parameter $\theta \in \mathbb{R}^{d'}$.

Policy Gradient Theorem

The question here is how to find a way to update θ such that the policy converges to the optimal policy. Note that by assuming that the initial state is deterministic $S_0 = s_0$ and that we stick to policy $\pi(a|s,\theta)$, a good θ shall maximize

$$J(\theta) = v_\pi(s_0) = \mathbb{E}_\pi(G_0|S_0 = s_0) \tag{186}$$

i.e. the overall return of the MDP which is always our goal in RL setting. The policy gradient theorem below gives a useful representation of the gradient of $J(\theta)$ w.r.t. θ.

Theorem 2. ***(Policy Gradient Theorem, Episodic Case)***

$$\nabla J(\theta) \propto \sum_{s\in S} \mu(s) \sum_{a\in\mathscr{A}} q_\pi(s,a)\nabla\pi(a|s,\theta) \tag{187}$$

where μ is the ***on-policy distribution*** *as a probability measure on S (defined in the proof).*

Proof. All gradients below are taken w.r.t. θ

$$\begin{aligned}
\nabla J(\theta) &= \nabla v_\pi(s_0) && (188)\\
&= \nabla \sum_{a\in\mathscr{A}} \pi(a|s_0,\theta) q_\pi(s_0,a) && (189)\\
&= \sum_{a\in\mathscr{A}} [\nabla\pi(a|s_0,\theta) q_\pi(s_0,a) + \pi(a|s_0,\theta)\nabla q_\pi(s_0,a)] && (190)
\end{aligned}$$

use the Bellman equation that $q_\pi(s,a) = \sum_{s'\in S, r\in R} p(s',r|s,a)\cdot[r+\gamma v_\pi(s')]$ proved above to see

$$\begin{aligned}
\nabla v_\pi(s_0) &= \sum_{a\in\mathscr{A}} \left(\nabla\pi(a|s_0,\theta) q_\pi(s_0,a) + \pi(a|s_0,\theta)\nabla \sum_{s'\in S, r\in R} p(s',r|s_0,a)\cdot[r+\gamma v_\pi(s')] \right) && (191)\\
&= \sum_{a\in\mathscr{A}} \left(\nabla\pi(a|s_0,\theta) q_\pi(s_0,a) + \pi(a|s_0,\theta)\gamma \sum_{s'\in S} p(s'|s_0,a)\cdot\nabla v_\pi(s') \right) && (192)
\end{aligned}$$

an iterative relationship in $\nabla v_\pi(s)$, so let's replace the gradient of value function on RHS with such formula iteratively

to see

$$\nabla v_\pi(s_0) \tag{193}$$
$$= \sum_{a\in\mathscr{A}} \nabla\pi(a|s_0,\theta) q_\pi(s_0,a) + \sum_{a\in\mathscr{A}} \pi(a|s_0,\theta)\gamma \sum_{s'\in S} p(s'|s_0,a)\cdot \tag{194}$$
$$\sum_{a'\in\mathscr{A}} \left(\nabla\pi(a'|s',\theta) q_\pi(s',a') + \pi(a'|s',\theta)\gamma \sum_{s''\in S} p(s''|s',a')\cdot \nabla v_\pi(s'') \right) \tag{195}$$

for simplicity, denote $f(s) = \sum_{a\in\mathscr{A}} \nabla\pi(a|s,\theta)\cdot q_\pi(s,a)$ so

$$\nabla v_\pi(s_0) \tag{196}$$
$$= f(s_0) + \sum_{a\in\mathscr{A}} \pi(a|s_0,\theta)\gamma \sum_{s'\in S} p(s'|s_0,a)\cdot \sum_{a'\in\mathscr{A}} \nabla\pi(a'|s',\theta) q_\pi(s',a') \tag{197}$$
$$+ \sum_{a\in\mathscr{A}} \pi(a|s_0,\theta)\gamma \sum_{s'\in S} p(s'|s_0,a) \sum_{a'\in\mathscr{A}} \pi(a'|s',\theta)\gamma \sum_{s''\in S} p(s''|s',a')\cdot \nabla v_\pi(s'') \tag{198}$$
$$= f(s_0) + \gamma \sum_{a\in\mathscr{A},s'\in S} f(s')\pi(a|s_0,\theta)p(s'|s_0,a) \tag{199}$$
$$+ \gamma^2 \sum_{s''\in S}\sum_{s'\in S}\sum_{a\in\mathscr{A}} \pi(a|s_0,\theta)p(s'|s_0,a) \sum_{a'\in\mathscr{A}} \pi(a'|s',\theta)p(s''|s',a')\cdot \nabla v_\pi(s'') \tag{200}$$
$$= f(s_0) + \gamma \sum_{s'\in S} f(s')\mathbb{P}_\pi\left(S_1 = s'|S_0 = s_0\right) \tag{201}$$
$$+ \gamma^2 \sum_{s'',s'\in S} \mathbb{P}_\pi\left(S_1 = s'|S_0 = s_0\right)\mathbb{P}_\pi\left(S_2 = s''|S_1 = s', S_0 = s_0\right)\nabla v_\pi(s'') \tag{202}$$
$$= f(s_0) + \gamma \sum_{s'\in S} f(s')\mathbb{P}_\pi\left(S_1 = s'|S_0 = s_0\right) + \gamma^2 \sum_{s''\in S} \mathbb{P}_\pi\left(S_2 = s''|S_0 = s_0\right)\nabla v_\pi(s'') \tag{203}$$

here we use the Markov property that $\mathbb{P}_\pi\left(S_2 = s''|S_1 = s', S_0 = s_0\right) = \mathbb{P}_\pi\left(S_2 = s''|S_1 = s'\right)$ and that such MDP is time-homogeneous so the transition kernel does not depend on time t. If we keep doing such iteration, we will see that

$$\nabla v_\pi(s_0) = f(s_0) + \gamma \sum_{s'\in S} f(s')\mathbb{P}_\pi\left(S_1 = s'|S_0 = s_0\right) + \gamma^2 \sum_{s''\in S} \mathbb{P}_\pi\left(S_2 = s''|S_0 = s_0\right)\nabla v_\pi(s'') \tag{204}$$
$$= ... \tag{205}$$
$$= \sum_{s\in S} f(x) \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_\pi\left(S_k = s|S_0 = s_0\right) \tag{206}$$
$$= \sum_{s\in S}\sum_{k=0}^{\infty} \gamma^k \mathbb{P}_\pi\left(S_k = s|S_0 = s_0\right) \sum_{a\in\mathscr{A}} \nabla\pi(a|s,\theta)\cdot q_\pi(s,a) \tag{207}$$
$$= \sum_{s\in S} \eta(s) \sum_{a\in\mathscr{A}} \nabla\pi(a|s,\theta)\cdot q_\pi(s,a) \tag{208}$$

define $\eta(s) \overset{def}{=} \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right)$, and $\mu(s) \overset{def}{=} \frac{\eta(s)}{\sum_{s' \in S} \eta(s')}$ to be the on-policy distribution, a probability measure on S. We have the representation that

$$\nabla J(\theta) = \left(\sum_{s' \in S} \eta(s')\right) \cdot \sum_{s \in S} \mu(s) \sum_{a \in \mathscr{A}} \nabla \pi(a|s, \theta) \cdot q_\pi(s, a) \tag{209}$$

$$\propto \sum_{s \in S} \mu(s) \sum_{a \in \mathscr{A}} \nabla \pi(a|s, \theta) \cdot q_\pi(s, a) \tag{210}$$

□

Remark. *To illustrate the definition of $\eta(s), \mu(s)$, let's first assume that $\gamma = 1$ which is always taken in the setting of episodic games. Then*

$$\eta(s) = \sum_{k=0}^{\infty} \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right) \tag{211}$$

is ***the expected amount of time MDP has spent in state*** *s if the deterministic initial state is s_0 and policy π is taken. As a result, $\mu(s)$ is the normalization of $\eta(s)$ and it stands for* ***the proportion of time MDP has spent in state*** *s if the deterministic initial state is s_0 and policy π is taken. That's why it's called on-policy distribution since $\mu(s)$ actually means* ***how much we care for state*** *s. One might be wondering if the normalization is well-defined, i.e. whether it's true that $\sum_{s \in S} \eta(s) < \infty$, let's do some calculations by noticing that in episodic games there exists an ending criteria so the terminal time T is formed as a random stopping time and the sum in the definition of $\eta(s)$ is actually from 0 to $T - 1$*

$$\sum_{s \in S} \eta(s) = \sum_{s \in S} \sum_{k=0}^{T-1} \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right) \tag{212}$$

$$= \sum_{k=0}^{T-1} \sum_{s \in S} \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right) \tag{213}$$

$$= T \tag{214}$$

so ***the proportional constant if actually T in the episodic case*** *and everything works well if $T < \infty$ a.s.. At this point, we can understand that $\eta(s) \overset{def}{=} \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right)$ is just the discounted version of $\eta(s)$ mentioned above.*

This concept is also used in the approximation of value function where one forms ***the approximated value function as*** *$\hat{v}(s, w)$* ***with weight*** *w* ***as parameter****. The mean square error of value function approximation is given by*

$$\overline{VE}(w) \overset{def}{=} \sum_{s \in S} \mu(s) \left[v_\pi(s) - \hat{v}(s, w)\right]^2 \tag{215}$$

and one typically wants to choose weight w to minimize this error.

The formulations above provide intuition for the on-policy distribution. Generally, we are not restricted to deterministic initial state any longer, assume $h(s)$ is a probability measure on S such that $h(s) = \mathbb{P}(S_0 = s)$, then $\eta(s)$ is defined as

$$\eta(s) \overset{def}{=} \mathbb{E}_{S_0 \sim h} \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_\pi (S_k = s | S_0) \tag{216}$$

$$= \sum_{s' \in S} h(s') \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_\pi (S_k = s | S_0 = s') \tag{217}$$

$$= \mathbb{E}_{S_0 \sim h} \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k \mathbb{I}_{S_k = s} \Big| S_0 \right) \tag{218}$$

$$= \mathbb{E}_{S_0 \sim h, \pi} \sum_{k=0}^{\infty} \gamma^k \mathbb{I}_{S_k = s} \tag{219}$$

$$= \sum_{k=0}^{\infty} \gamma^k \mathbb{P}_{S_0 \sim h, \pi} (S_k = s) \tag{220}$$

*has **the following property** that*

$$\forall s \in S, \eta(s) = \mathbb{P}_{S_0 \sim h, \pi} (S_0 = s) + \gamma \sum_{k=1}^{\infty} \gamma^{k-1} \mathbb{P}_{S_0 \sim h, \pi} (S_k = s) \tag{221}$$

$$= h(s) + \gamma \sum_{l=0}^{\infty} \gamma^l \mathbb{P}_{S_0 \sim h, \pi} (S_{l+1} = s) \tag{222}$$

$$= h(s) + \gamma \sum_{l=0}^{\infty} \gamma^l \sum_{s' \in S} \mathbb{P}_{S_0 \sim h, \pi} (S_l = s') \mathbb{P}_{S_0 \sim h, \pi} (S_{l+1} = s | S_l = s') \tag{223}$$

$$= h(s) + \gamma \sum_{l=0}^{\infty} \gamma^l \sum_{s' \in S} \mathbb{P}_{S_0 \sim h, \pi} (S_l = s') \mathbb{P}_{S_0 = s', \pi} (S_1 = s) \tag{224}$$

$$= h(s) + \gamma \sum_{s' \in S} \eta(s') \mathbb{P}_{S_0 = s', \pi} (S_1 = s) \tag{225}$$

$$= h(s) + \gamma \sum_{s' \in S} \eta(s') \sum_{a \in \mathscr{A}} \pi(a|s') \mathbb{P}_{S_0 = s', \pi} (S_1 = s | A_0 = a) \tag{226}$$

$$= h(s) + \gamma \sum_{s' \in S} \eta(s') \sum_{a \in \mathscr{A}} \pi(a|s') p(s|s', a) \tag{227}$$

the interpretation of this equation is that if $S_0 = s$ then it contributes 1 to $\eta(s)$ immediately, otherwise the contribution to $\eta(s)$ comes from the transition from other states back to s.

REINFORCE

Now we apply the policy gradient theorem to provide a family of policy gradient algorithms called REINFORCE. Since we want to pick the θ that maximizes $J(\theta)$, a natural idea comes from gradient ascent that θ should be updated in the way that

$$\theta \leftarrow \theta + \alpha \nabla J(\theta^t) \tag{228}$$

where $\alpha > 0$ is a positive parameter for the step size in gradient ascent. By policy gradient theorem,

$$\nabla J(\theta) \propto \sum_{s \in S} \mu(s) \sum_{a \in \mathscr{A}} \nabla \pi(a|s, \theta) \cdot q_\pi(s, a) \tag{229}$$

so very naturally the **gradient ascent** gives us the update

$$\theta \leftarrow \theta + \alpha \sum_{s \in S} \mu(s) \sum_{a \in \mathscr{A}} \nabla \pi(a|s, \theta) \cdot q_\pi(s, a) \tag{230}$$

Remark. *Notice that the proportional constant in the policy gradient theorem can be absorbed into the step size parameter α so we don't have to worry about it!*

However, we still have two problems here with this update scheme. The first problem is that we don't know $\mu(s)$ in practice since it's the on-policy distribution as the proportional of time MDP has spent staying in each state. In order to design a model-free algorithm (which is the basic requirement for the algorithm to be useful), we never want to compute $\mu(s)$ analytically. The second problem here is that we don't know $q_\pi(s, a)$, the state-action value under policy π.

The first problem is not hard to solve by noticing that μ is a probability measure so we can find a random variable that has distribution μ. Let's first consider the case where $\gamma = 1$. By Kolmogorov's cycle trick,

$$\mu(s) = \frac{\sum_{k=0}^{\infty} \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right)}{\sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}_\pi \left(S_k = s | S_0 = s_0\right)} \tag{231}$$

is the stationary measure (and stationary distribution) of Markov chain $\{S_t\}$ given terminal time $T < \infty$ and s_0 as a recurrent state. That is to say, by assuming that we start from the initial state following the stationary distribution $S_0 \sim \mu$, it's always true that $\forall t, S_t \sim \mu$.

Remark. *The policy gradient algorithm adopts the* ***assumption that the initial state*** S_0 ***already follows the stationary distribution*** μ*. Although this seems not reasonable in practice since we are often starting from some "arbitrary" initial state, the asymptotic behavior of Markov chain ensures the convergence of the distribution $S_t \xrightarrow{d} \mu$ $(t \to \infty)$ if $\{S_t\}$ is nice enough (it's an ergodic Markov chain).*

As argued above, for the case where $\gamma = 1$ and the assumption that $S_0 \sim \mu$, we have

$$\sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\nabla\pi(a|s,\theta)\cdot q_\pi(s,a) = \mathbb{E}_\pi\sum_{a\in\mathscr{A}}\nabla\pi(a|S_t,\theta)\cdot q_\pi(S_t,a) \tag{232}$$

the structure as expectation of random variables! Now let's deal with the state-action value $q_\pi(S_t, a)$. Notice that

$$q_\pi(S_t,a) = \mathbb{E}_\pi(G_t|S_t, A_t = a) \tag{233}$$

$$q_\pi(S_t,A_t) = \mathbb{E}_\pi(G_t|S_t, A_t) \tag{234}$$

so it's a good idea to exploit again the structure of expectation contained in this state-action value. In order to make the sum of all actions $a \in \mathscr{A}$ an expectation w.r.t. A_t, we need a probability measure on the action space and naturally we notice that the policy itself is a probability measure on the action space. This leads to the transformation that

$$\mathbb{E}_\pi\sum_{a\in\mathscr{A}}\nabla\pi(a|S_t,\theta)\cdot q_\pi(S_t,a) = \mathbb{E}_\pi\sum_{a\in\mathscr{A}}\pi(a|S_t,\theta)\frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}\cdot q_\pi(S_t,a) \tag{235}$$

$$= \mathbb{E}_\pi\mathbb{E}_{A_t\sim\pi(\cdot|S_t,\theta)}\left(\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}\cdot q_\pi(S_t,A_t)\Big|S_t\right) \tag{236}$$

$$= \mathbb{E}_\pi\left(\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}\cdot q_\pi(S_t,A_t)\right) \tag{237}$$

$$= \mathbb{E}_\pi\left[\nabla\log\pi(A_t|S_t,\theta)\cdot\mathbb{E}_\pi(G_t|S_t,A_t)\right] \tag{238}$$

$$= \mathbb{E}_\pi\left[G_t\cdot\nabla\log\pi(A_t|S_t,\theta)\right] \tag{239}$$

After solving those two problems, we see that the gradient ascent update of θ is given by

$$\theta \leftarrow \theta + \alpha\cdot\mathbb{E}_\pi\left[G_t\cdot\nabla\log\pi(A_t|S_t,\theta)\right] \tag{240}$$

although we do not know the value of the exact gradient (the expectation) here, it's easy to randomly sample from a distribution whose expectation is equal to the exact gradient. By adopting the **stochastic gradient ascent** idea presented in bandit gradient algorithms, we get the **policy gradient update** that

$$\theta \leftarrow \theta + \alpha\cdot G_t\cdot\nabla\log\pi(A_t|S_t,\theta) \tag{241}$$

Remark. *The importance of policy gradient algorithm is that in each step only the future discounted rewards G_t and the gradient of the log policy at current state-action pair is used. As a result, we don't have to worry about* ***large state space or action space*** *any longer! Besides, policy gradient algorithm provide the possibility of* ***converging to a stochastic policy as the optimal policy****. In value based methods, the optimal policy is always formed as the deterministic policy putting all probability mass on the action that maximizes state-action value for the observed state.*

Remark. *The policy gradient update presented above only holds for $\gamma = 1$. However, if one hopes to generalize the update formula for $\gamma \in (0, 1)$ we can adopt the following idea. The problem we are facing at stage k is always equivalent to the same problem we are facing at stage $k+1$ with all rewards discounted by multiplying γ. As a result, we would perform the same policy gradient update at stage $k+1$ as what we have done at stage k and the only difference is that there's one more discount factor γ in the G_t. So the* ***discounted policy gradient update*** *is given by*

$$\theta \leftarrow \theta + \alpha \cdot \gamma^t G_t \cdot \nabla \log \pi(A_t|S_t, \theta) \tag{242}$$

performed at time *t* ***within each single episode****. Notice that here we actually use the assumption that the MDP has infinite time horizon.*

At this point, let's state the most fundamental policy gradient algorithm **REINFORCE**. The algorithm starts with an initial parameter value θ and a given form of the policy $\pi(a|s, \theta)$. For each episode, the algorithms trains the parameter θ in the following way

- Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$ following the current policy under parameter θ.
- At each time t from 0 to $T-1$ within this single episode, build up G_t incrementally and perform the update

$$\theta \leftarrow \theta + \alpha \cdot \gamma^t G_t \cdot \nabla \log \pi(A_t|S_t, \theta) \tag{243}$$

Remark. *We briefly talk about how to set up the policy with a specific form $\pi(a|s, \theta)$ here. Alike the situation in the bandit gradient algorithm, the simple way is to set up the preference $h(s, a, \theta)$ for each state-action pair (s, a) and form the policy as the softmax image of the preference, i.e.*

$$\pi(a|s, \theta) = \frac{e^{h(s,a,\theta)}}{\sum_{a' \in \mathscr{A}} e^{h(s,a',\theta)}} \tag{244}$$

and the preference can be chosen as a function linear in θ, i.e.

$$h(s, a, \theta) = \theta^T x(s, a) \tag{245}$$

under such setting, the ***eligibility vector*** *is*

$$\nabla \log \pi(a|s, \theta) = x(s, a) - \sum_{a' \in \mathscr{A}} \pi(a'|s, \theta) \cdot x(s, a') \tag{246}$$

In more general cases or practical usage, such policy $\pi(a|s, \theta)$ is always approximated by a neural network with θ to be the collection of all parameters of the NN. The output is ensured to be a policy by setting the output layer of the NN as a softmax layer.

REINFORCE with Baseline

REINFORCE makes use of the stochastic gradient ascent and the sample of the gradient has expectation as the true gradient by the policy gradient theorem proved above. However, although such sampling is not biased, there might be a huge variance that significantly slows down the convergence of the algorithm. As shown in the example of bandit gradient algorithm, adding a baseline would be a good variance reduction technique. The reason why we can add a baseline without interfering with the policy gradient update can be seen from the policy gradient theorem that

$$\sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\nabla\pi(a|s,\theta)\cdot q_\pi(s,a)=\sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\nabla\pi(a|s,\theta)\cdot[q_\pi(s,a)-b(s)] \tag{247}$$

this is because

$$\sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\nabla\pi(a|s,\theta)\cdot b(s)=\sum_{s\in S}\mu(s)\cdot b(s)\cdot\nabla 1=0 \tag{248}$$

embedding this baseline into policy gradient update provides the **policy gradient update with baseline** b

$$\theta\leftarrow\theta+\alpha\cdot\gamma^t[G_t-b(S_t)]\cdot\nabla\log\pi(A_t|S_t,\theta) \tag{249}$$

note that **such baseline can only depend on the current state but not current action** such that all previously derived conclusions still work.

In bandit gradient algorithm, we choose the baseline to be the sample mean of all past rewards. Here we can do the similar thing and set

$$b(S_t)=\mathbb{E}_{\pi^*}(G_t|S_t)=v_*(S_t) \tag{250}$$

the state value of S_t. Naturally, we maintain $\hat{v}(S_t,w)$ as an estimate of the state value with weight parameter w and set it as the baseline. In the algorithm, we shall update both parameters θ and w with the information provided simultaneously. For the update of w, we minimize the weighted mean square error as previously described

$$\min_w\overline{\mathrm{VE}}(w)=\sum_{s\in S}\mu(s)\left[v_\pi(s)-\hat{v}(s,w)\right]^2 \tag{251}$$

with gradient descent. Compute the gradient w.r.t. w that

$$\nabla_w\overline{\mathrm{VE}}(w)=\sum_{s\in S}\mu(s)\nabla\left[v_\pi(s)-\hat{v}(s,w)\right]^2 \tag{252}$$

$$=2\sum_{s\in S}\mu(s)\left[\hat{v}(s,w)-v_\pi(s)\right]\cdot\nabla_w\hat{v}(s,w) \tag{253}$$

$$\propto\mathbb{E}_\pi\left[\hat{v}(S_t,w)-v_\pi(S_t)\right]\cdot\nabla_w\hat{v}(S_t,w) \tag{254}$$

here we adopt the same assumption in policy gradient update that we start from initial state following the stationary distribution $S_0 \sim \mu$ to represent the sum as an expectation w.r.t. S_t, then

$$\nabla_w \overline{\text{VE}}(w) \propto \mathbb{E}_\pi \left[\hat{v}(S_t, w) - \mathbb{E}_\pi(G_t|S_t)\right] \cdot \nabla_w \hat{v}(S_t, w) \tag{255}$$

$$= \mathbb{E}_\pi \left[\hat{v}(S_t, w) - G_t\right] \cdot \nabla_w \hat{v}(S_t, w) \tag{256}$$

the gradient has the structure of the expectation so the stochastic gradient descent update of w for state value approximation is

$$w \leftarrow w + \alpha \left[G_t - \hat{v}(S_t, w)\right] \cdot \nabla_w \hat{v}(S_t, w) \tag{257}$$

where $\alpha > 0$ is the step size parameter.

Now let's give the **REINFORCE with baseline** algorithm. It starts with an initial parameter value θ, w and a given form of the policy $\pi(a|s, \theta)$ and the state value approximation $\hat{v}(s, w)$. For each episode, the algorithms trains the parameter θ and w in the following way

- Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$ following the current policy under parameter θ.
- At each time t from 0 to $T-1$ within this single episode, build up G_t incrementally and perform the updates (α^θ is the step size parameter in the update of θ and α^w is the step size parameter in the update of w)

$$w \leftarrow w + \alpha^w \cdot \left[G_t - \hat{v}(S_t, w)\right] \cdot \nabla_w \hat{v}(S_t, w) \tag{258}$$

$$\theta \leftarrow \theta + \alpha^\theta \cdot \gamma^t [G_t - \hat{v}(S_t, w)] \cdot \nabla_\theta \log \pi(A_t|S_t, \theta) \tag{259}$$

Remark. *There's many possible choices for the form of $\hat{v}(s, w)$. One can set it to be linear in w or to set up a neural network with parameter w. Practically setting $\alpha^w = \frac{0.1}{\mathbb{E}_{S_t \sim \mu} ||\nabla \hat{v}(S_t, w)||^2}$ would be the best choice but setting a good α^θ should depend on different problems.*

Actor-Critic Methods

In the REINFORCE with baseline, we maintain an estimate of state-value function and update its weight w with stochastic gradient descent to form a baseline. However, if a state transition from state s_1 to state s_2 happens, such state-value estimate only takes into account the state value of s_1 but has not yet taken into account the state value of s_2. By including the value estimate of both states in a state transition, we would be able to build up the **temporal difference (TD) error** as a criterion of how well we are doing in estimating the state value. Actor-critic methods allow ideas in TD methods to combine with the policy gradient ideas, which provides us with some more useful results.

The **actor** in our setting refers to the policy since it keeps generating actions while the **critic** refers to the estimated state value function since it assesses the actions taken. The main idea here is to replace the G_t in REINFORCE update with $G_{t:t+1}$, the one-step return at time t formed as

$$G_{t:t+1} = R_{t+1} + \gamma G_{t+1} \tag{260}$$

since $G_t = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + ...) = R_{t+1} + \gamma G_{t+1}$. Now the unknown G_{t+1} can be approximated by

$$\hat{v}(S_{t+1}, w) \tag{261}$$

so the term $G_t - \hat{v}(S_t, w)$ in REINFORCE with baseline update should be replaced with

$$R_{t+1} + \gamma \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w) \tag{262}$$

which is just the TD error at time t.

Remark. *The intuitive explanation of such TD error is that $R_{t+1} + \gamma \hat{v}(S_{t+1}, w)$ is the latest and better estimate of the state value of S_t while $\hat{v}(S_t, w)$ is the former estimate of the state value of S_t. As a result, the TD error at each time measures the error in the estimate made at that time.*

One main advantage of replacing the overall return with one-step return is that the algorithm now can be formed as totally online and incremental. In the previous REINFORCE training, one always have to wait for an episode to terminate since the overall return G_t is required in the update. For one-step actor-critic methods, the update only requires R_{t+1}, S_t, S_{t+1} that can all be directly observed at time t. As a result, now we are able to train the RL model when the game is still on play.

Remark. *The idea of using one-step return comes from TD learning algorithms. Although one-step return has bias, typically it enjoys much less variance and enables the algorithm to update incrementally so generally it's a better object to use compared to the overall return.*

It's immediate that one may generalize such one-step method to n-step and λ-return algorithms. However, the main idea is always the same so here we just state the simplest one-step method.

Now let's give the **one-step actor-critic** algorithm. It starts with an initial parameter value θ, w and a given

form of the policy $\pi(a|s,\theta)$ and the state value approximation $\hat{v}(s,w)$. The algorithm is **totally online** so the training can be done simultaneously as the game proceeds.

- Start the episode with an initial state, set up $I \leftarrow 1$ (to record γ^t, the discount factor).
- When the current state S is not the terminal state, sample an action from the policy $A \sim \pi(\cdot|S,\theta)$. Take such action A so one gets reward R and the state transits to S'.
- Set up the TD error

$$\delta \leftarrow R + \gamma \hat{v}(S', w) - \hat{v}(S, w) \tag{263}$$

  where $\hat{v}(S', w)$ is set as 0 if S' is the terminal state.

- Do the following update that (α^θ is the step size parameter in the update of θ and α^w is the step size parameter in the update of w)

$$w \leftarrow w + \alpha^w \cdot \delta \cdot \nabla_w \hat{v}(S_t, w) \tag{264}$$

$$\theta \leftarrow \theta + \alpha^\theta \cdot I\delta \cdot \nabla_\theta \log \pi(A_t|S_t,\theta) \tag{265}$$

- Move to the next state and do such update iteratively

$$I \leftarrow \gamma I \tag{266}$$

$$S \leftarrow S' \tag{267}$$

Extension for Continuing Tasks

One might notice that the algorithms and theorems above only work for episodic games. However, **continuing games** are also prevalent and crucial to solve. Those games do not have a terminal criterion it's possible to last forever. For policy gradient methods, the largest difference from that in episodic case is the formation of the performance $J(\theta)$ defined as

$$J(\theta) \overset{def}{=} \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}_\pi(R_t|S_0) \tag{268}$$

Remark. *Such performance is called **the average rate of reward per time step**. Intuitively, if one starts from initial state S_0 and follows policy π, one would get on average $\mathbb{E}_\pi(R_t|S_0)$ amount of reward at time $t-1$. $J(\theta)$ is just the asymptotic average of the expected amount of reward one can get at time $0, 1, ..., h-1$ (h is large enough). Unlike the episodic case where one always have an almost surely finite terminal time, in the continuing case the game might never stop so it's not reasonable to define $J(\theta)$ as $\mathbb{E}_\pi(G_0|S_0)$ any longer (there's typically no discount rate in continuing game setting).*

Notice that the expectation can be calculated in the following way that

$$\mathbb{E}_\pi(R_t|S_0) = \sum_{s\in S, a\in\mathscr{A}} \mathbb{P}_\pi\left(S_t = s, A_t = a\right) \cdot \mathbb{E}_\pi(R_t|S_0, S_t = s, A_t = a) \tag{269}$$

$$= \sum_{s\in S, a\in\mathscr{A}} \mathbb{P}_\pi\left(S_t = s\right) \cdot \pi(a|s) \cdot \sum_{s'\in S, r\in R} p(s', r|s, a) \cdot r \tag{270}$$

now assume $S_t \sim \mu$ follows the on-policy distribution under policy π, we see that

$$\mathbb{E}_\pi(R_t|S_0) \sim \sum_{s\in S} \mu(s) \sum_{a\in\mathscr{A}} \pi(a|s) \sum_{s'\in S, r\in R} p(s', r|s, a) \cdot r \ (t\to\infty) \tag{271}$$

for large enough t under the assumption that the Markov chain $\{S_t\}$ is ergodic since $S_t \xrightarrow{d} \mu$ $(t\to\infty)$ for stationary distribution μ. Recall that we have proved in the previous context that when there's no discount factor the on-policy distribution μ is just the stationary distribution.

As a result, we are actually saying that

$$\lim_{t\to\infty} \mathbb{E}_\pi(R_t|S_0) = \sum_{s\in S} \mu(s) \sum_{a\in\mathscr{A}} \pi(a|s) \sum_{s'\in S, r\in R} p(s', r|s, a) \cdot r \tag{272}$$

and notice that the existence of such limit implies

$$J(\theta) = \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}_\pi(R_t|S_0) = \lim_{t\to\infty} \mathbb{E}_\pi(R_t|S_0) \tag{273}$$

$$= \sum_{s\in S} \mu(s) \sum_{a\in\mathscr{A}} \pi(a|s) \sum_{s'\in S, r\in R} p(s', r|s, a) \cdot r \tag{274}$$

different characterizations of the performance $J(\theta)$.

Correspondingly, the value functions in the continuing game setting is different from those in the episodic game. Since $J(\theta)$ stands for the average rate of reward per time step, the return G_t is now formed as **the differential return, i.e. the sum of excessive future return compared to** $J(\theta)$

$$G_t \stackrel{def}{=} [R_{t+1} - J(\theta)] + [R_{t+2} - J(\theta)] + ... \tag{275}$$

and the value functions are defined similarly as $v_\pi(s) = \mathbb{E}(G_t|S_t = s), q_\pi(s,a) = \mathbb{E}(G_t|S_t = s, A_t = a)$. Now let's state that the policy gradient theorem still holds for such formulation so the previously stated algorithms still work for continuing tasks up to a slight modification.

Theorem 3. ***(Policy Gradient Theorem, Continuing Case)***

$$\nabla J(\theta) = \sum_{s\in S} \mu(s) \sum_{a\in\mathscr{A}} q_\pi(s,a)\nabla\pi(a|s,\theta) \tag{276}$$

where μ is the stationary distribution of Markov chain $\{S_t\}$.

Proof. The Bellman equation in the continuing case becomes

$$\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi(G_t|S_t = s, A_t = a) && (277)\\
&= \sum_{s'\in S} \mathbb{P}\left(S_{t+1} = s'|S_t = s, A_t = a\right) \mathbb{E}_\pi(G_t|S_{t+1} = s', S_t = s, A_t = a) && (278)\\
&= \sum_{s'\in S} p(s'|s,a)\mathbb{E}_\pi(G_{t+1} + R_{t+1} - J(\theta)|S_{t+1} = s', S_t = s, A_t = a) && (279)\\
&= \sum_{s'\in S, r\in R} p(s', r|s, a)[v_\pi(s') + r - J(\theta)] && (280)
\end{aligned}$$

now get the gradient of state value function w.r.t. θ (same calculations as before)

$$\nabla v_\pi(s) = \sum_{a\in\mathscr{A}} \left[\nabla\pi(a|s,\theta)q_\pi(s,a) + \pi(a|s,\theta)\left(-\nabla J(\theta) + \sum_{s'\in S} p(s'|s,a)\nabla v_\pi(s')\right)\right] \tag{281}$$

and represent $\nabla J(\theta)$ with $\nabla v_\pi(s)$ to see

$$\nabla J(\theta) = \sum_{a\in\mathscr{A}} \left[\nabla\pi(a|s,\theta)q_\pi(s,a) + \pi(a|s,\theta)\sum_{s'\in S} p(s'|s,a)\nabla v_\pi(s')\right] - \nabla v_\pi(s) \tag{282}$$

Since $J(\theta)$ does not depend on the state s, the RHS also does not depend on the state s so we can take weighted

average of both sides w.r.t. the stationary distribution μ to get

$$\nabla J(\theta) = \sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\left[\nabla\pi(a|s,\theta)q_\pi(s,a) + \pi(a|s,\theta)\sum_{s'\in S}p(s'|s,a)\nabla v_\pi(s')\right] - \sum_{s\in S}\mu(s)\nabla v_\pi(s) \tag{283}$$

$$= \sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\nabla\pi(a|s,\theta)q_\pi(s,a) + \sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\pi(a|s,\theta)\sum_{s'\in S}p(s'|s,a)\nabla v_\pi(s') - \sum_{s\in S}\mu(s)\nabla v_\pi(s) \tag{284}$$

$$= \sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\nabla\pi(a|s,\theta)q_\pi(s,a) \tag{285}$$

since $\sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\pi(a|s,\theta)\sum_{s'\in S}p(s'|s,a)\nabla v_\pi(s') = \sum_{s\in S}\mu(s)\nabla v_\pi(s)$. This is due to the fact that μ is the stationary distribution so

$$\forall s'\in S, \mu(s') = \sum_{s\in S}\mu(s)\sum_{a\in\mathscr{A}}\pi(a|s,\theta)p(s'|s,a) \tag{286}$$

□

Let's end the part for continuing task extension with the slight modification of the one-step actor-critic algorithm. Now the one-step return $G_{t:t+1}$ is formed such that

$$G_t = G_{t+1} + R_{t+1} - J(\theta) \tag{287}$$

so the estimation of the overall return by one-step return shall have one more term $\overline{R}$ included, correspondent to $J(\theta)$. As a result, the TD error shall be formed as

$$\delta \leftarrow \hat{v}(S', w) + R - \overline{R} - \hat{v}(S, w) \tag{288}$$

and one shall update $\overline{R}$ as the estimate for $J(\theta)$, for example

$$\overline{R} \leftarrow \overline{R} + \alpha^{\overline{R}}\delta \tag{289}$$

for some $\alpha^{\overline{R}} > 0$ as the step size parameter.

Extensions for Continuous Action Parameterization

The policy gradient algorithms appear naturally to deal with problems with a large state space where action value and state value approximation becomes infeasible. However, there might also be the case where the action space is too large to be viewed as a set of finitely many elements. In such case, treating $\pi(a|s,\theta)$ as the output of softmax function acted on the preference/feature does not work well since each action is only assigned an infinitesimal small amount of probability mass.

The parameterization of actions is then necessary and the policy is always formed as a **parametric family of distribution**. Let's take the Gaussian distribution $N(\mu,\sigma^2)$ as an example. The action A shall be sampled from the distribution

$$A \sim N(\mu(s,\theta),\sigma^2(s,\theta)) \tag{290}$$

where $\mu(s,\theta),\sigma(s,\theta)$ are the mean and standard deviation of the sampling Gaussian and they depend on the current state s and the parameter θ. In other words,

$$\pi(a|s,\theta) = \frac{1}{\sqrt{2\pi}\sigma(s,\theta)} e^{-\frac{[a-\mu(s,\theta)]^2}{2\sigma^2(s,\theta)}} \tag{291}$$

Now the parameter θ consists of two parts: θ_μ as the parameters in $\mu(s,\theta)$ and θ_σ as the parameters in $\sigma(s,\theta)$. The mean can be formed as a linear function in θ_μ and the standard deviation can be formed as the exponential of a linear function in θ_σ to ensure its positivity

$$\mu(s,\theta) = \theta_\mu^T x_\mu(s), \sigma(s,\theta) = e^{\theta_\sigma^T x_\sigma(s)} \tag{292}$$

where x_μ, x_σ are features extracted based on the states.

In **Gaussian policy parameterization**, by calculations under the settings above, the eligibility vector w.r.t. θ_μ has the following form that

$$\nabla_{\theta_\mu} \log \pi(a|s,\theta) = \frac{\nabla_{\theta_\mu}\pi(a|s,\theta)}{\pi(a|s,\theta)} \tag{293}$$

$$= \frac{\frac{\partial}{\partial\mu(s,\theta)}\pi(a|s,\theta)\cdot\frac{\partial}{\partial\theta_\mu}\mu(s,\theta)}{\pi(a|s,\theta)} \tag{294}$$

$$= \frac{\pi(a|s,\theta)\cdot\frac{2(a-\mu(s,\theta))}{2\sigma^2(s,\theta)}\cdot x_\mu(s)}{\pi(a|s,\theta)} \tag{295}$$

$$= \frac{(a-\mu(s,\theta))}{\sigma^2(s,\theta)}\cdot x_\mu(s) \tag{296}$$

and now consider the eligibility vector w.r.t. θ_σ

$$\nabla_{\theta_\sigma} \log \pi(a|s,\theta) = \frac{\nabla_{\theta_\sigma} \pi(a|s,\theta)}{\pi(a|s,\theta)} \tag{297}$$

$$= \frac{\frac{\partial}{\partial \sigma(s,\theta)} \pi(a|s,\theta) \cdot \frac{\partial}{\partial \theta_\sigma} \sigma(s,\theta)}{\pi(a|s,\theta)} \tag{298}$$

$$= \frac{\left(-\frac{1}{\sigma(s,\theta)} \pi(a|s,\theta) + \pi(a|s,\theta)[a-\mu(s,\theta)]^2 \sigma^{-3}(s,\theta)\right) \cdot \sigma(s,\theta) x_\sigma(s)}{\pi(a|s,\theta)} \tag{299}$$

$$= \left(\frac{[a-\mu(s,\theta)]^2}{\sigma^2(s,\theta)} - 1 \right) x_\sigma(s) \tag{300}$$

so in the policy gradient update we will update $\theta_\mu, \theta_\sigma$ respectively with those eligibility vectors.

Remark. *Consider an example given in the Barto-Sutton book for* ***Bernoulli logistic unit*** *as a stochastic neuron-like unit with its input $x(S_t)$ as a feature vector of states and its output A_t as an action taking values 0 or 1 such that $A_t \sim B(1, P_t)$. There are preferences toward two states $h(s,0,\theta), h(s,1,\theta)$ when one is at state s with parameter θ and their difference is given by a weighted sum of the unit's input, i.e. $h(s,1,\theta) - h(s,0,\theta) = \theta^T x(s)$.*

Now if the policy is formed as softmax policy based on those two preferences, then

$$P_t = \pi(1|S_t,\theta_t) = \frac{e^{h(S_t,1,\theta)}}{e^{h(S_t,0,\theta)} + e^{h(S_t,1,\theta)}} \tag{301}$$

$$= \frac{1}{1+e^{-\theta^T x(s)}} \tag{302}$$

Next, we derive the Monte-Carlo REINFORCE update of θ upon the receipt of return G_t. By previously mentioned policy gradient update,

$$\theta \leftarrow \theta + \alpha \gamma^t G_t \cdot \nabla \log \pi(A_t|S_t,\theta) \tag{303}$$

where $\alpha > 0$ is some step size parameter in stochastic gradient ascent. Now let's plug in the value of P_t derived above to see that

$$\theta \leftarrow \theta + \alpha \gamma^t G_t \cdot (-1)^{A_t} [1 - \pi(A_t|S_t,\theta)] e^{-\theta^T x(s)} \cdot x(s) \tag{304}$$

after calculations for the eligibility vector. This gives the REINFORCE update without baseline.

Reinforcement Learning Theory: MDP

In the context above we have covered some of the most important concepts in RL, but mostly without a rigorous framework and proof. In the sections below we are going to introduce the theory behind those concepts to fill in the gaps.

We refer to the book *Reinforcement Learning: Theory and Algorithms by Agarwal, Jiang, Kakade, Sun (AJKS)* for the RL theory part. I will be mentioning the important theorems and proving the exercises left (lemmas) in the book but will not cover all the details. The readers are welcome to go through this nice book on their own for more details.

Infinite Horizon MDP Setting

There are many nice results for infinite-horizon MDP, i.e. the MDP has unbounded time horizon $t = 0, 1, 2, ...$ still infinity. Let's stick to the notations in the AJKS book that $\mathscr{S}$ denotes state space, $\mathscr{A}$ denotes action space, $\Delta(A)$ denotes the set of all possible probability measures on A and $P : \mathscr{S} \times \mathscr{A} \to \Delta(\mathscr{S})$ is the transition kernel. We often denote $P(s'|s, a)$ as the probability of transitioning into state s' on taking action a in state s. The reward function $r : \mathscr{S} \times \mathscr{A} \to [0, 1]$, different from that in previous context, is assumed to be **bounded and deterministic**, so $r(s, a)$ is just the reward for taking action a in state s. $\gamma \in [0, 1)$ is the discount factor and $\mu \in \Delta(\mathscr{S})$ is the initial state distribution, i.e. $S_0 \sim \mu$.

For a given MDP, τ_t denotes the trajectory of the MDP till time t, i.e. $\tau_t = (s_0, a_0, r_0, ..., s_t, a_t, r_t)$. We typically use capitalized letters for random variables and non-capitalized letters for realizations. Set $\mathscr{H}$ consists of all possible trajectories of all lengths and the policy is defined as $\pi : \mathscr{H} \to \Delta(\mathscr{A})$.

Remark. *In previous context, we naturally considered the Markovian policy or the* ***stationary policy*** $\pi : \mathscr{S} \to \Delta(\mathscr{A})$*, i.e. on seeing state $S_t = s_t$, $\pi(s_t) \in \Delta(\mathscr{A})$ gives a probability measure on the action space and A_t is sampled from this measure. However, there are definitely more general definition of policies that allows dependence on the whole history of trajectory. The policy defined above refers to this kind of policy denoted $\pi(S_0, A_0, ..., S_t)$ and it's still a probability measure on $\mathscr{A}$ but A_t shall be sampled from this measure.*

The state value function is defined as

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \Big| S_0 = s \right] \tag{305}$$

and the action value function is defined as

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \Big| S_0 = s, A_0 = a \right] \tag{306}$$

they do not depend on time t and our goal is to achieve the optimal state value $\max_\pi V^\pi(s)$ for given initial state s.

For stationary policies, we have the following Bellman consistency equation telling us the connection between state value and action value function.

Theorem 4. ***(Bellman Consistency Equation)*** *Let π be stationary policy, then $\forall s \in \mathscr{S}, a \in \mathscr{A}$,*

$$V^\pi(s) = Q^\pi(s, \pi(s)) \tag{307}$$

$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a),\pi} V^\pi(S') \tag{308}$$

where $\pi(s)$ is actually a probability measure on $\mathscr{A}$ and $Q^\pi(s, \pi(s))$ means the expectation under such probability measure, i.e. $Q^\pi(s, \pi(s)) = \mathbb{E}_{A \sim \pi(\cdot|s)} Q^\pi(s, A)$.

Proof. By the law of total probability,

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(S_t, A_t) \Big| S_0 = s \right] \tag{309}$$

$$= \sum_{a \in \mathscr{A}} \mathbb{P}_\pi \left(A_0 = a | S_0 = s \right) \cdot \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(S_t, A_t) \Big| S_0 = s, A_0 = a \right] \tag{310}$$

$$= \sum_{a \in \mathscr{A}} \pi(a|s) \cdot Q^\pi(s,a) = Q^\pi(s, \pi(s)) \tag{311}$$

To prove the second equation, notice the decomposition $\sum_{t=0}^\infty \gamma^t r(S_t, A_t) = r(S_0, A_0) + \gamma \sum_{t=0}^\infty \gamma^t r(S_{t+1}, A_{t+1})$ so by considering what values S_1 might take,

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(S_t, A_t) \Big| S_0 = s, A_0 = a \right] \tag{312}$$

$$= r(s,a) + \gamma \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(S_{t+1}, A_{t+1}) \Big| S_0 = s, A_0 = a \right] \tag{313}$$

$$= r(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathbb{P}_\pi \left(S_1 = s' | S_0 = s, A_0 = a \right) \cdot \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(S_{t+1}, A_{t+1}) \Big| S_0 = s, A_0 = a, S_1 = s' \right] \tag{314}$$

$$= r(s,a) + \gamma \sum_{s' \in \mathscr{S}} P(s'|s,a) \cdot \mathbb{E}_\pi \left[\sum_{t=0}^\infty \gamma^t r(S_{t+1}, A_{t+1}) \Big| S_1 = s' \right] \tag{315}$$

$$= r(s,a) + \gamma \sum_{s' \in \mathscr{S}} P(s'|s,a) \cdot V^\pi(s') \tag{316}$$

$$= r(s,a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a),\pi} V^\pi(S') \tag{317}$$

by Markov property of MDP. □

Typically, we would like to organize transition kernels and value functions as matrices and vectors for compact notations. View $V^\pi \in \mathbb{R}^{|\mathscr{S}|}$ as a vector, $Q^\pi, r \in \mathbb{R}^{|\mathscr{S}| \cdot |\mathscr{A}|}$ as vectors and the transition kernel $P \in \mathbb{R}^{(|\mathscr{S}| \cdot |\mathscr{A}|) \times |\mathscr{S}|}$ as a matrix. Notice that the transition kernel does not provide us with the transition of the distribution of the MDP,

so we can incorporate the policy into the transition kernel and define the matrix

$$P^\pi \in \mathbb{R}^{(|\mathscr{S}|\cdot|\mathscr{A}|)\times(|\mathscr{S}|\cdot|\mathscr{A}|)}, P^\pi_{(s,a),(s',a')} \stackrel{def}{=} P(s'|s,a)\cdot\pi(a'|s') \tag{318}$$

so $P^\pi_{(s,a),(s',a')}$ as the entry of such matrix is actually the probability of seeing the transition of state-action pair from (s,a) to (s',a') under stationary policy π. The following lemma provides a representation of the value functions.

Lemma 2. ***(Compact Form of Bellman Consistency Equation)*** *Under stationary policy π, the following equations hold*

$$Q^\pi = r + \gamma P V^\pi \tag{319}$$
$$Q^\pi = r + \gamma P^\pi Q^\pi \tag{320}$$
$$Q^\pi = (I - \gamma P^\pi)^{-1} r \tag{321}$$

Proof. Let's prove that each entry on both sides of the equation is the same. By the Bellman consistency equation,

$$\forall s\in\mathscr{S}, a\in\mathscr{A}, Q^\pi_{(s,a)} = r_{(s,a)} + \gamma \mathbb{E}_{S'\in P(\cdot|s,a),\pi} V^\pi_{(S')} \tag{322}$$
$$= r_{(s,a)} + \gamma \sum_{s'\in\mathscr{S}} P_{(s,a),s'}\cdot V^\pi_{(s')} \tag{323}$$
$$= r_{(s,a)} + \gamma P V^\pi \tag{324}$$

is just the compact form of the Bellman consistency equation.

For the second equation, $PV^\pi \in \mathbb{R}^{|\mathscr{S}|\cdot|\mathscr{A}|}$ so compute the entries

$$\forall s\in\mathscr{S}, a\in\mathscr{A}, (PV^\pi)_{(s,a)} = \sum_{s'\in\mathscr{S}} P(s'|s,a)\cdot V^\pi(s') \tag{325}$$
$$= \sum_{s'\in\mathscr{S}} P(s'|s,a)\cdot Q^\pi(s',\pi(s')) \tag{326}$$
$$= \sum_{s'\in\mathscr{S}} P(s'|s,a) \sum_{a'\in\mathscr{A}} Q^\pi(s',a')\pi(a'|s') \tag{327}$$
$$= \sum_{s'\in\mathscr{S},a'\in\mathscr{A}} P^\pi_{(s,a),(s',a')} Q^\pi_{(s',a')} \tag{328}$$
$$= (P^\pi Q^\pi)_{(s,a)} \tag{329}$$

where we still use the Bellman consistency equation that $V^\pi(s) = Q^\pi(s,\pi(s))$.

To prove the last equation, just need to prove $I-\gamma P^\pi$ is invertible, i.e. $(I-\gamma P^\pi)x = 0$ implies $x = 0$ for $\forall x \in \mathbb{R}^{|\mathscr{S}|\cdot|\mathscr{A}|}$. We can prove that whenever $x\neq 0$, $||I-\gamma P^\pi||_\infty > 0$. This can be done by applying triangle inequality and noticing the fact that $||x||_\infty \geq ||P^\pi x||_\infty$ since the sum of entries of P^π's each row is always 1.

□

Lemma 3. ***(Interpretation of*** $(I-\gamma P^\pi)^{-1}$***)***

$$\left[(1-\gamma)(I-\gamma P^\pi)^{-1}\right]_{(s,a),(s',a')} = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}_\pi\left(S_t=s', A_t=a' | S_0=s, A_0=a\right) \tag{330}$$

Proof. Consider the spectral radius $\rho(\gamma P^\pi) = \gamma\rho(P^\pi)$ with $||P^\pi||_\infty = 1$ as a transitional kernel matrix, so $\forall k \in \mathbb{N}, ||(P^\pi)^k||_\infty = 1$ by Gelfand's formula, $\rho(P^\pi) = \lim_{k\to\infty} ||(P^\pi)^k||_\infty^{\frac{1}{k}} = 1$ so $\rho(\gamma P^\pi) = \gamma < 1$.

This ensures the power series expansion

$$(I-\gamma P^\pi)^{-1} = \sum_{t=0}^{\infty}\gamma^t (P^\pi)^t \tag{331}$$

notice the Markov transition kernel structure of P^π, actually $(P^\pi)^t$ is just the t-step transition kernel of state-action pair under policy π so it's natural that

$$[(P^\pi)^t]_{(s,a),(s',a')} = \mathbb{P}_\pi\left(S_t=s', A_t=a' | S_0=s, A_0=a\right) \tag{332}$$

the conclusion is proved.

□

Remark. *Notice that if we sum the RHS w.r.t. all* $(s',a') \in \mathscr{S}\times\mathscr{A}$*, the sum is 1. In other words, each row of the matrix on the LHS can be seen as a probability distribution. It's called the induced distribution over state-action pair when following policy* π *after starting from state-action pair* (s,a)*.*

The representation $Q^\pi = (I-\gamma P^\pi)^{-1} r$ *can thus be understood as that* ***the state-action value*** $Q^\pi(s,a)$ ***is the weighted average of reward w.r.t. the induced distribution over state-action pair when following policy*** π ***and starting the MDP from state-action pair*** (s,a)*.*

Optimal Policy and Bellman Optimality Equation

Let's still consider the infinite horizon MDP and define the optimal value functions as the pointwise supreme of value functions w.r.t. all possible policies $\pi \in \Pi$ where Π **denotes the set of all non-stationary and randomized policies**

$$V^*(s) \overset{def}{=} \sup_{\pi \in \Pi} V^\pi(s), Q^*(s,a) \overset{def}{=} \sup_{\pi \in \Pi} Q^\pi(s,a) \tag{333}$$

the following theorem ensures that the optimal policy exists and surprisingly there must exist a stationary deterministic optimal policy.

Theorem 5. ***(Existence of Optimal Policy)*** *There exists a stationary and deterministic optimal policy π^* such that*

$$\forall s \in \mathscr{S}, a \in \mathscr{A}, V^{\pi^*}(s) = V^*(s), Q^{\pi^*}(s,a) = Q^*(s,a) \tag{334}$$

Proof. The proof starts with proving

$$\sup_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r(S_t, A_t) \Big| \pi, (S_0, A_0, R_0, S_1) = (s, a, r, s')\right] = \gamma V^*(s') \tag{335}$$

which means that on knowing the state at time 1, the history of trajectory does not make any difference toward the optimal value as if starting from time 1. The proof makes use of **the offset policy** $\pi_{(s,a,r)}$ w.r.t. π defined as

$$\pi_{(s,a,r)}(A_t | S_0 = s_0, A_0 = a_0, R_0 = r_0, ..., S_t = s_t) \overset{def}{=} \pi(A_t | S_0 = s, A_0 = a, R_0 = r, S_1 = s_0, A_1 = a_0, R_1 = r_0, ..., S_{t+1} = s_t) \tag{336}$$

in simple words, $\pi_{(s,a,r)}$ behaves in the same manner as π as if one has observed the state, action, reward in the first period of the MDP as s, a, r. One might notice that when π traverses through Π, the offset policy $\pi_{(s,a,r)}$ also traverses through Π for any (s,a,r) so $\sup_{\pi \in \Pi} V^{\pi_{(s,a,r)}}(s) = \sup_{\pi \in \Pi} V^\pi(s) = V^*(s)$. Using those tools with Markov property, one can easily prove this equation.

Naturally, the stationary deterministic optimal policy is constructed as

$$\tilde{\pi}(s) \overset{def}{=} \arg\sup_{a \in \mathscr{A}} \mathbb{E}\left[r(s,a) + \gamma V^*(S_1) | S_0 = s, A_0 = a\right] \tag{337}$$

whenever we are at state s, always puts all probability mass on the action that maximizes the expectation of the sum of current reward and discounted maximum possible future reward. This policy is stationary since $\tilde{\pi}$ is a function only of the current state s. By proving

$$\forall s_0 \in \mathscr{S}, V^*(s_0) \leq \mathbb{E}_{\tilde{\pi}}\left[r(s_0, A_0) + \gamma V^*(S_1)\right] \tag{338}$$

and proceed iteratively, we will see that $\forall s_0 \in \mathscr{S}, V^*(s_0) \leq V^{\tilde{\pi}}(s_0)$.

□

Remark. *The surprising point here is that* ***although we are finding the optimal policy among all non-stationary and randomized policy (a much larger space), there must exist an optimal policy to be stationary and deterministic (a much smaller space)****. This allows us to focus only on stationary policies in the infinite horizon MDP case, which greatly simplifies the problem.*

The following Bellman optimality equation provides a characterization for the optimal value function.

Theorem 6. ***(Bellman Optimality Equation)*** *Q is the optimal state-action value function if and only if*

$$\forall s \in \mathscr{S}, a \in \mathscr{A}, Q(s,a) = r(s,a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a)} \max_{a' \in \mathscr{A}} Q(S', a') \tag{339}$$

the stationary deterministic optimal policy is given by the greedy policy w.r.t. Q^* that

$$\pi^*(s) = \pi_{Q^*}(s) \overset{def}{=} \arg\max_{a \in \mathscr{A}} Q(s,a) \tag{340}$$

Proof. First prove the compact form of Bellman optimality equation that

$$V^*(s) = \max_{a \in \mathscr{A}} Q^*(s,a) \tag{341}$$

by Bellman consistency equation and the existence of optimal policy above, $V^*(s) = Q^*(s, \pi^*(s)) \leq \max_{a \in \mathscr{A}} Q^*(s,a)$. The other direction is proved by considering a perturbation of the optimal policy π^*, i.e. consider π_a as a non-stationary policy that always chooses action a at time 0 and follow π^* afterwards. It's quite clear that $V^{\pi_a}(s) = Q^{\pi^*}(s,a)$ concludes the proof.

Now we denote the Bellman optimality equation by the **Bellman optimality operator** $\mathscr{T} : \mathbb{R}^{|\mathscr{S}| \cdot |\mathscr{A}|} \to \mathbb{R}^{|\mathscr{S}| \cdot |\mathscr{A}|}$ such that $\mathscr{T}Q \overset{def}{=} r + \gamma P V_Q$ where $V_Q = \max_{a \in \mathscr{A}} Q(s,a)$ is the state value function induced by Q. It's easy to verify that

$$\forall s \in \mathscr{S}, a \in \mathscr{A}, (r + \gamma P V_Q)_{(s,a)} = r(s,a) + \gamma \sum_{s' \in \mathscr{S}} P(s'|s,a) V_Q(s') \tag{342}$$

$$= r(s,a) + \gamma \sum_{s' \in \mathscr{S}} P(s'|s,a) \max_{a' \in \mathscr{A}} Q(s', a') \tag{343}$$

$$= r(s,a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a)} \max_{a' \in \mathscr{A}} Q(S', a') \tag{344}$$

is just the RHS of Bellman optimality equation. As a result, **Bellman optimality equation holds for Q iff Q is the fixed point of Bellman optimality operator $\mathscr{T}$.**

Now if $Q = Q^*$ is the optimal value function, $Q^* = \mathscr{T}Q^*$ follows easily from Bellman consistency equation and the compact form proved above.

Conversely, if $Q = \mathscr{T}Q$, then by the representation that $Q^\pi = (I - \gamma P^\pi)^{-1} r$, we can prove $Q = Q^{\pi_Q}$ so Q is actually the value function under the greedy policy w.r.t. Q itself. By the last theorem, there exists an optimal

policy that is stationary and deterministic so we only need to prove that for any stationary deterministic policy π', $Q^{\pi_Q} - Q^{\pi'} \geq 0$. This follows from the calculation

$$Q^{\pi_Q} - Q^{\pi'} = (I - \gamma P^{\pi'})^{-1}[(I - \gamma P^{\pi'})Q^{\pi_Q} - r] \tag{345}$$

$$= (I - \gamma P^{\pi'})^{-1}[(I - \gamma P^{\pi'})Q^{\pi_Q} - (I - \gamma P^{\pi_Q})Q^{\pi_Q}] \tag{346}$$

$$= (I - \gamma P^{\pi'})^{-1}\gamma(P^{\pi_Q} - P^{\pi'})Q^{\pi_Q} \tag{347}$$

notice that each row of $\frac{1}{1-\gamma}(I - \gamma P^{\pi'})^{-1}$ is a probability distribution as proved in the lemma above and that

$$[(P^{\pi_Q} - P^{\pi'})Q^{\pi_Q}]_{(s,a)} = \sum_{s' \in \mathscr{S}, a' \in \mathscr{A}} [P^{\pi_Q}_{(s,a),(s',a')} - P^{\pi'}_{(s,a),(s',a')}]Q^{\pi_Q}(s', a') \tag{348}$$

$$= \sum_{s' \in \mathscr{S}} P(s'|s,a) \sum_{a' \in \mathscr{A}} [\pi_Q(a'|s') - \pi'(a'|s')]Q^{\pi_Q}(s', a') \tag{349}$$

$$= \sum_{s' \in \mathscr{S}} P(s'|s,a)[Q^{\pi_Q}(s', \pi_Q(s')) - Q^{\pi_Q}(s', \pi'(s'))] \geq 0 \tag{350}$$

by the definition of the greedy policy π_Q. This concludes the proof. □

Remark. *One might be able to find that the Bellman optimality equation is closely connected with the* ***dynamic programming principle (DPP)*** *in the setting of stochastic control problem. View $\{S_t\}$ as state process and π as a deterministic policy, then π is actually the control process and r is the running reward. DPP tells us that*

$$\forall s \in \mathscr{S}, V^*(s) \overset{DPP}{=} \sup_{\pi} \mathbb{E}_{A \sim \pi(\cdot|s), S' \sim P(\cdot|s,a)} [r(s, A) + \gamma V^*(S')] \tag{351}$$

$$= \sup_{\pi} \mathbb{E}_{S' \sim P(\cdot|s,\pi(s))} [r(s, \pi(s)) + \gamma V^*(S')] \tag{352}$$

$$= \sup_{\pi} \mathbb{E}_{S' \sim P(\cdot|s,\pi(s))} \left[r(s, \pi(s)) + \gamma \max_{a' \in \mathscr{A}} Q^*(S', a') \right] \tag{353}$$

if we fix the current state s and denote $\pi(s) = a$ as the action taken according to policy π, then the sup w.r.t. π is actually the sup w.r.t. action a

$$V^*(s) = \sup_{a \in \mathscr{A}} \mathbb{E}_{S' \sim P(\cdot|s,a)} \left[r(s, a) + \gamma \max_{a' \in \mathscr{A}} Q^*(S', a') \right] = \sup_{a \in \mathscr{A}} \mathbb{E}_{S' \sim P(\cdot|s,a)} [r(s, a) + \gamma V^*(S')] \tag{354}$$

gives ***the Bellman optimality equation w.r.t. optimal state value function*** *V^*. This is consistent with what we can get from the Bellman optimality equation w.r.t. optimal state-action value function Q^* since if we take sup w.r.t. action a on both sides, we will see that*

$$V^*(s) = \max_{a \in \mathscr{A}} Q^*(s, a) = \max_{a \in \mathscr{A}} \left\{ r(s, a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a)} \max_{a' \in \mathscr{A}} Q^*(S', a') \right\} = \max_{a \in \mathscr{A}} \left\{ r(s, a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a)} V^*(S') \right\} \tag{355}$$

the same equation as derived from DPP.

Finite Horizon MDP

So far we have been focusing on infinite horizon MDP and it turns out that there are many good properties, e.g. there exists a stationary deterministic optimal policy and the Bellman optimality equations characterize the optimal value functions. However, in finite horizon MDP the settings change. Firstly, we **do not introduce the discount rate** any longer since the sum of finitely many real numbers must be finite. Secondly, **time dependence** is introduced for transition kernel and reward function. Now the transition kernel is denoted P_h at time step h, similarly the reward function is r_h so at each time step we maintain a different transition kernel and reward function.

Remark. *One might be interested in the reason why time dependence in only introduced in the finite horizon case. Actually the intuition comes from stochastic control problems. In infinite horizon control problems, the value function $V(s)$ has a nice time structure so it's always organized in the form that only depends on the state s but not time t. However, in finite horizon problems one has to form the value function as $V(t,s)$ since the time structure is unknown. Simply saying, in infinite horizon problems, after one time step, one is still facing exactly the same problem as if all rewards are discounted by factor γ so the symmetricity makes the problem much easier.*

Note that the time dependence of value functions in the finite horizon setting is also one of the main reasons that time-dependent finite horizon MDP is not applied in practice.

By the reasoning above, the value functions under general policy π are naturally formed as

$$V_h^\pi(s) \overset{def}{=} \mathbb{E}_\pi\left[\sum_{t=h}^{H-1} r_t(S_t, A_t)\Big| S_h = s\right] \tag{356}$$

$$Q_h^\pi(s,a) \overset{def}{=} \mathbb{E}_\pi\left[\sum_{t=h}^{H-1} r_t(S_t, A_t)\Big| S_h = s, A_h = a\right] \tag{357}$$

where the time horizon is denoted $0, 1, ..., H-1$. This is exactly the same setting as that in the stochastic control problems, to organize value functions backwardly. Our objective is still to find the optimal policy π that maximizes $V_0^\pi(s)$ for given initial state s.

The Bellman optimality equation now also has time structure, provides characterization of the optimal value functions

$$V_h^*(s) \overset{def}{=} \sup_{\pi\in\Pi} V_h^\pi(s) \tag{358}$$

$$Q_h^*(s,a) \overset{def}{=} \sup_{\pi\in\Pi} Q_h^\pi(s,a) \tag{359}$$

with $V_H = 0, Q_H = 0$ defined at the tail (since the game has already ended).

Theorem 7. ***(Bellman Optimality Equation, Finite Horizon)*** *Q_h is the optimal value function iff*

$$Q_h(s,a) = r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} \max_{a'\in\mathscr{A}} Q_{h+1}(S', a') \tag{360}$$

and the deterministic optimal policy at time h is given by the greedy policy w.r.t. Q_h^* that

$$\pi^*(s,h) = \pi_{Q^*}(s,h) \overset{def}{=} \arg\max_{a\in\mathscr{A}} Q_h^*(s,a) \tag{361}$$

Proof. We provide the whole proof for this theorem here mimicking the proof for the infinite horizon case. Notice that the theorem of the existence of optimal policy can be generalized here to prove that **there must exist a deterministic optimal policy** taking action $\pi^*(s,h)$ at time step h.

Let's first prove **the compact form** that

$$\forall h \in [H], s \in \mathscr{S}, V_h^*(s) = \max_{a\in\mathscr{A}} Q_h^*(s,a) \tag{362}$$

where $[H] = \{0,1,...,H-1\}$. By the law of total probability,

$$V_h^*(s) = V_h^{\pi^*}(s) = \sum_{a\in\mathscr{A}} \mathbb{P}_{\pi^*}(A_h = a|S_h = s)\cdot Q_h^{\pi^*}(s,a) \leq \max_{a\in\mathscr{A}} Q_h^*(s,a) \tag{363}$$

on the other hand, consider π_a as a policy that always takes action a at the first time step (time step h) but follows π^* afterwards, we have

$$\forall a \in \mathscr{A}, V_h^*(s) \geq V_h^{\pi_a}(s) = Q_h^{\pi^*}(s,a) = Q^*(s,a) \tag{364}$$

so the connection between two optimal value functions is proved.

Now let's prove that Q_h^* satisfies the Bellman optimality equation

$$\begin{aligned}
Q_h^*(s,a) &= \max_\pi Q_h^\pi(s,a) && (365)\\
&= r_h(s,a) + \max_\pi \mathbb{E}_\pi\left[\sum_{t=h+1}^{H-1} r_t(S_t,A_t)\Big|S_h = s, A_h = a\right] && (366)\\
&= r_h(s,a) + \max_\pi \sum_{s'\in\mathscr{S}} P_h(s'|s,a)\cdot\mathbb{E}_\pi\left[\sum_{t=h+1}^{H-1} r_t(S_t,A_t)\Big|S_h = s, A_h = a, S_{h+1} = s'\right] && (367)\\
&= r_h(s,a) + \max_\pi \sum_{s'\in\mathscr{S}} P_h(s'|s,a)\cdot V_{h+1}^\pi(s') && (368)\\
&= r_h(s,a) + \max_\pi \mathbb{E}_{S'\sim P_h(\cdot|s,a)} V_{h+1}^\pi(S') && (369)\\
&= r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} V_{h+1}^*(S') && (370)\\
&= r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} \max_{a'\in\mathscr{A}} Q_{h+1}^*(S',a') && (371)
\end{aligned}$$

gets proved.

Conversely, if the Bellman optimality equation holds for Q_h, consider π_Q as the greedy policy w.r.t. Q_h defined

as

$$\pi_Q(s,h) \overset{def}{=} \arg\max_{a\in\mathscr{A}} Q_h(s,a) \tag{372}$$

so it's easy to verify that

$$\begin{aligned}
Q_h^{\pi_Q}(s,a) &= r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} V_{h+1}^{\pi_Q}(S') && (373)\\
&= r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} \sum_{a'\in\mathscr{A}} \pi_Q(a'|S',h+1)\cdot Q_{h+1}^{\pi_Q}(S',a') && (374)\\
&= r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} Q_{h+1}^{\pi_Q}(S',\pi_Q(S',h+1)) && (375)\\
&= r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} \max_{a'\in\mathscr{A}} Q_{h+1}^{\pi_Q}(S',a') = Q_h(s,a) && (376)
\end{aligned}$$

so $Q = Q^{\pi_Q}$ is actually just the value function w.r.t. the policy π_Q. Now in order to prove that π_Q is the optimal policy, it suffices to prove that for any non-stationary deterministic policy π', $Q_h^{\pi_Q} - Q_h^{\pi'} \geq 0$.

From the **Bellman consistency equation for finite horizon MDP** proved above that

$$\begin{cases} V_h^\pi(s) = \sum_{a\in\mathscr{A}} \mathbb{P}_\pi\left(A_h = a|S_h = s\right)\cdot Q_h^\pi(s,a) \\ Q_h^\pi(s,a) = r_h(s,a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} V_{h+1}^\pi(S') \end{cases} \tag{377}$$

we rebuild the representation of Q_h^π for fixed horizon $h\in[H]$ as a vector in $\mathbb{R}^{|\mathscr{S}|\cdot|\mathscr{A}|}$ that

$$\begin{cases} Q_h^\pi = r_h + P_h V_{h+1}^\pi \\ Q_h^\pi = r_h + P_h^\pi Q_{h+1}^\pi \end{cases} \tag{378}$$

where $(P_h^\pi)_{(s,a),(s',a')} \overset{def}{=} (P_h)_{(s,a),s'}\pi(a'|s',h)$. This leads to the fact that

$$\begin{aligned}
Q_h^{\pi_Q} - Q_h^{\pi'} &= r_h + P_h^{\pi_Q} Q_{h+1}^{\pi_Q} - r_h - P_h^{\pi'} Q_{h+1}^{\pi'} && (379)\\
&= P_h^{\pi_Q} Q_{h+1}^{\pi_Q} - P_h^{\pi'} Q_{h+1}^{\pi'} && (380)\\
&= P_h^{\pi_Q}(Q_{h+1}^{\pi_Q} - Q_{h+1}^{\pi'}) + (P_h^{\pi_Q} - P_h^{\pi'})Q_{h+1}^{\pi'} && (381)
\end{aligned}$$

with the first term on RHS to have positive components since $Q_{h+1}^{\pi_Q} - Q_{h+1}^{\pi'} \geq 0$ by the definition of π_Q and each row of $P_h^{\pi_Q}$ is a probability distribution. For the second term on the RHS, it's also positive since

$$\begin{aligned}
[(P_h^{\pi_Q} - P_h^{\pi'})Q_{h+1}^{\pi'}]_{(s,a)} &= \sum_{s'\in\mathscr{S},a'\in\mathscr{A}} ([P_h^{\pi_Q}]_{(s,a),(s',a')} - [P_h^{\pi'}]_{(s,a),(s',a')})\cdot Q_{h+1}^{\pi'}(s',a') && (382)\\
&= \sum_{s'\in\mathscr{S}} (P_h)_{(s,a),s'}\cdot Q_{h+1}^{\pi'}(s',\pi_Q(s',h)) - \sum_{s'\in\mathscr{S}} (P_h)_{(s,a),s'}\cdot Q_{h+1}^{\pi'}(s',\pi'(s',h)) && (383)\\
&= \mathbb{E}_{S'\sim P_h(\cdot|s,a)}[Q_{h+1}^{\pi'}(S',\pi_Q(S',h)) - Q_{h+1}^{\pi'}(S',\pi'(S',h))] \geq 0 && (384)
\end{aligned}$$

by noticing that π_Q, π' are both deterministic policies and that by the definition of π_Q again, $Q_{h+1}^{\pi'}(S', \pi_Q(S', h)) \geq Q_{h+1}^{\pi'}(S', \pi'(S', h))$. So we have proved that π_Q is the optimal policy and Q must be the optimal value function. $\square$

Remark. *The proof above is very similar to the case for infinite horizon MDP and the only difference lies in the different formulations (time dependency) of Bellman consistency equation and Bellman optimality equation stated in the proof.*

Similarly, if we restrict ourselves to deterministic policy π, RL actually becomes a discrete-time finite horizon stochastic control problem where the policy is just the control process and it's time dependent. By applying the DPP again, we see that

$$V_h^*(s) \overset{DPP}{=} \sup_{\pi} \mathbb{E}_{\pi, A\sim\pi(\cdot|s,h), S'\sim P_h(\cdot|s,A)}[r(s, A) + V_{h+1}^*(S')] \tag{385}$$

$$= \sup_{a\in\mathscr{A}} \mathbb{E}_{S'\sim P_h(\cdot|s,a)}[r(s, a) + V_{h+1}^*(S')] \tag{386}$$

since π is deterministic and we denote $a = \pi(s, h)$, the sup taken w.r.t. all deterministic policies is just the sup taken w.r.t. all possible actions. Therefore, we derive ***the Bellman optimality equation for state value function*** *V^*. To see the consistency with the Bellman optimality equation for Q^* above, take maximum w.r.t. action a to see*

$$V_h^*(s) = \max_{a\in\mathscr{A}} Q_h^*(s, a) = \max_{a\in\mathscr{A}} \left\{ r_h(s, a) + \mathbb{E}_{S'\sim P_h(\cdot|s,a)} V_{h+1}^*(S') \right\} \tag{387}$$

so even if in the time-dependent and finite horizon case, it's still consistent with DPP.

Reinforcement Learning Theory: Model-Based Methods

In this section, we introduce some model-based methods that directly follow from the Bellman optimality equation and relevant ideas stated above. Notice that those methods are not frequently used in practice since they require the knowledge of the transition kernel $P_{(s,a),s'}$, which is often unknown. We denote the infinite horizon MDP as $M = (\mathscr{S}, \mathscr{A}, P, r, \gamma)$ (without specification we consider infinite horizon case) and $L(P, r, \gamma)$ denotes the total number of bits required to specify M. Of course a feasible method should have polynomial time to figure out the optimal policy and the time complexity shall be a function of $|\mathscr{S}|, |\mathscr{A}|, L(P, r, \gamma), \gamma$. A method is called **strongly polynomial** if the time complexity is a polynomial in everything with no dependence on $L(P, r, \gamma)$.

Value Iteration

The idea of value iteration is to find out Q^* and to construct the optimal policy as the greedy policy w.r.t. Q^*. In order to figure out Q^*, notice that it is the fixed point of the Bellman optimality operator $\mathscr{T}$ so it's natural to apply fixed point iteration. As what we have shown in the previous context, $\mathscr{T}$ is actually a contraction mapping under infinity norm.

Lemma 4. ***($\mathscr{T}$ as Contraction Mapping)***

$$\forall Q, Q' \in \mathbb{R}^{|\mathscr{S}|\cdot|\mathscr{A}|}, ||\mathscr{T}Q - \mathscr{T}Q'||_\infty \leq \gamma ||Q - Q'||_\infty \tag{388}$$

Combine with the fact that $\gamma < 1$, we know that the fixed point uniquely exists and the fixed point iteration converges to Q^*. Now we can apply the same error analysis techniques as those in the fixed point iteration to get the following error bounds.

Remark. *Notice that we have actually proved that although the optimal policy is often not unique, all different optimal policies share the* ***unique optimal value function*** *Q^*, V^* from the uniqueness of the fixed point.*

Lemma 5. ***(Q-Error Amplification)***

$$\forall Q \in \mathbb{R}^{|\mathscr{S}|\cdot|\mathscr{A}|}, V^{\pi_Q} \geq V^* - \frac{2||Q - Q^*||_\infty}{1-\gamma}\vec{1} \tag{389}$$

Proof. It follows from

$$V^*(s) - V^{\pi_Q}(s) = Q^*(s, \pi^*(s)) - Q^*(s, \pi_Q(s)) + Q^*(s, \pi_Q(s)) - Q^{\pi_Q}(s, a) \tag{390}$$

then apply Bellman consistency equation to write the second difference as an expectation in terms of V and derive the bound that

$$\forall s \in \mathscr{S}, V^*(s) - V^{\pi_Q}(s) \leq 2||Q - Q^*||_\infty + \gamma ||V^* - V^{\pi_Q}||_\infty \tag{391}$$

□

Remark. *This lemma tells us that if we adopt the greedy policy w.r.t. Q, when Q is very close to Q^*, V^{π_Q} is also very close to V^*. So the sub-optimality gap in V of the greedy policy can be bounded by the sub-optimality gap in Q. This will help us figure out the shrinking rate of the sub-optimality gap in V if the greedy policy is taken.*

Theorem 8. ***(Q-Value Iteration Convergence)*** *Set up the fixed point iteration $Q^{(0)} = 0, Q^{(k+1)} = \mathscr{T}Q^{(k)}$ and set $\pi^{(k)} = \pi_{Q^{(k)}}$ as the greedy policy in the k-th iteration, then for error tolerance $\varepsilon > 0$ such that $V^{\pi^{(k)}} \geq V^* - \varepsilon\vec{1}$ we need $k \geq \frac{\log \frac{2}{(1-\gamma)^2\varepsilon}}{1-\gamma}$ number of iterations to guarantee such error tolerance.*

Proof. By the lemma above,

$$\frac{2||Q^{(k)} - Q^*||_\infty}{1-\gamma} \leq \varepsilon \tag{392}$$

implies $V^{\pi^{(k)}} \geq V^* - \varepsilon\vec{1}$ and by contraction mapping, $||Q^{(k)} - Q^*||_\infty \leq \gamma^k ||Q^{(0)} - Q^*||_\infty$ so we just need to ensure that

$$\frac{2\gamma^k}{1-\gamma}||Q^{(0)} - Q^*||_\infty \leq \varepsilon \tag{393}$$

notice that the reward is always between 0 to 1 so $||Q^*||_\infty \leq \frac{1}{1-\gamma}$ and such k suffices to ensure $\frac{2\gamma^k}{(1-\gamma)^2} \leq \varepsilon$. □

Remark. *To specify the MDP, we need $L(P, r, \gamma)$ number of bits so the $V^{\pi^{(k)}}$ from value iteration has no difference from the true V^* under the machine accuracy level if the difference is no more than $\varepsilon = 2^{-L(P,r,\gamma)}$. That's why the time complexity of value iteration is $O\left(|\mathscr{S}|^2|\mathscr{A}|\frac{L(P,r,\gamma)\log\frac{1}{1-\gamma}}{1-\gamma}\right)$ and it's not strongly polynomial. Notice that we would operate the value iteration for $O\left(\frac{L(P,r,\gamma)\log\frac{1}{1-\gamma}}{1-\gamma}\right)$ iterations but in each iteration we have to apply the Bellman optimality operator that results in $O(|\mathscr{S}|^2|\mathscr{A}|)$ calculations since for each state s and action a, the calculation of the expectation $\mathbb{E}_{S'\sim P(\cdot|s,a)} \max_{a'\in\mathscr{A}} Q(S', a')$ takes $O(|\mathscr{S}|)$ time.*

Of course, value iteration can be generalized to apply for finite horizon MDP, and one just have to change the update to the time dependent Bellman optimality equation. It's still a contraction mapping and the greedy policy shall be formed at each time step.

Policy Iteration

Policy iteration consists of two parts, policy evaluation and policy improvement. The idea is to start from some policy π_k, evaluate the value function Q^{π_k} and then form $\pi_{k+1} = \pi_{Q^{\pi_k}}$ as the greedy policy w.r.t. Q^{π_k} to be a better one. It's natural to prove that the policy is improving as proceeding in the policy iteration.

Lemma 6. ***(Policy Improvement)***

$$Q^{\pi_{k+1}} \geq \mathscr{T} Q^{\pi_k} \geq Q^{\pi_k}, ||Q^{\pi_{k+1}} - Q^*||_\infty \leq \gamma ||Q^{\pi_k} - Q^*||_\infty \tag{394}$$

Proof. $\mathscr{T} Q^\pi \geq Q^\pi$ directly follows from the Bellman consistency equation. Since π_{k+1} is greedy w.r.t. Q^{π_k}, by Bellman consistency equation again $Q^{\pi_k} \leq r + \gamma P^{\pi_{k+1}} Q^{\pi_k}$ and iterative application combined with the power series expansion of $(I - \gamma P^{\pi_{k+1}})^{-1}$ proves $Q^{\pi_{k+1}} \geq Q^{\pi_k}$. Apply such conclusion, we can prove $Q^{\pi_{k+1}} \geq \mathscr{T} Q^{\pi_k}$.

The reason we want to build the relationship between not only $Q^{\pi_k}, Q^{\pi_{k+1}}$ but also $\mathscr{T} Q^{\pi_k}$ is that it makes our life easier when building up the error shrinking rate since Q^* is the fixed point

$$||Q^{\pi_{k+1}} - Q^*||_\infty \leq ||\mathscr{T} Q^{\pi_k} - \mathscr{T} Q^*||_\infty \leq \gamma ||Q^{\pi_k} - Q^*||_\infty \tag{395}$$

by the contraction mapping. □

Theorem 9. ***(Policy Iteration Convergence)*** *Let π_0 be any initial policy to start the policy iteration stated above. For error tolerance $\varepsilon > 0$ such that $Q^{\pi^{(k)}} \geq Q^* - \varepsilon \vec{1}$ we need $k \geq \frac{\log \frac{1}{(1-\gamma)\varepsilon}}{1-\gamma}$ number of iterations to guarantee such error tolerance.*

Proof. By the lemma above,

$$||Q^{\pi_k} - Q^*||_\infty \leq \gamma^k ||Q^{\pi_0} - Q^*||_\infty \leq \frac{\gamma^k}{1-\gamma} \tag{396}$$

so we just need to ensure that

$$\frac{\gamma^k}{1-\gamma} \leq \varepsilon \tag{397}$$

and $k \geq \frac{\log \frac{1}{(1-\gamma)\varepsilon}}{1-\gamma}$ suffices. □

Remark. *A similar argument to that made for value iteration gives the following time complexity for policy iteration $O\left((|\mathscr{S}|^3 + |\mathscr{S}|^2|\mathscr{A}|)\frac{L(P,r,\gamma)\log\frac{1}{1-\gamma}}{1-\gamma}\right)$. Notice that in each iteration policy evaluation requires the calculation $Q^{\pi_k} = (I - \gamma P^{\pi_k})^{-1} r$ so it takes time $O(|\mathscr{S}|^3)$ to invert the matrix and time $O(|\mathscr{S}|^2|\mathscr{A}|)$ to compute the matrix product since π_k is deterministic (sparse structure of P^{π_k}).*

The difference here is that the number of policies will not exceed $|\mathscr{A}|^{|\mathscr{S}|}$ so we can get rid of the $L(P,r,\gamma)$ in the time complexity. For a fixed discount factor γ, policy iteration is strongly polynomial with time complexity $O\left((|\mathscr{S}|^3 + |\mathscr{S}|^2|\mathscr{A}|)\cdot \min\left\{\frac{|\mathscr{A}|^{|\mathscr{S}|}}{|\mathscr{S}|}, \frac{|\mathscr{S}|^2|\mathscr{A}|\log\frac{|\mathscr{S}|^2}{1-\gamma}}{1-\gamma}\right\}\right)$. Here $\frac{|\mathscr{S}|^2|\mathscr{A}|\log\frac{|\mathscr{S}|^2}{1-\gamma}}{1-\gamma}$ is the maximum number of iterations

needed to find the optimal policy under fixed γ and $\frac{|\mathscr{A}|^{|\mathscr{S}|}}{|\mathscr{S}|}$ is the maximum number of policies we will need to check, taking a minimum gives the number of iterations needed in the worst scenario.

UCB, LinUCB and Regret Analysis

One of the algorithms that performs pretty well for bandit problems is the UCB method. Recall that UCB algorithm tells us to first explore through all K possible actions once and then select action based on

$$A_t = \arg\max_a \left(Q_t(a) + c\sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(a)}} \right) \tag{398}$$

for some small probability $\delta \in (0,1)$, some constant $c > 0$ and $N_t(a) = \sum_{i=1}^{t-1} \mathbb{I}_{A_t=a}$. In this section, we illustrate why this bound is called the upper confidence bound, what's the interpretation of hyperparameters c, δ and why this algorithm works that well in practice. Before doing any of those, we want to build a measurement for different algorithms in bandit problem for the purpose of comparison.

Cumulative Regret

The **cumulative regret** up to time T is defined as

$$R_T = T \max_a q_*(a) - \sum_{i=1}^{T} q_*(A_i) \tag{399}$$

the term regret means that at time i on taking action A_i, we would regret on not taking the optimal action and get the maximum possible expected reward $\max_a q_*(a)$. As a result

$$R_T = \sum_{i=0}^{T-1} \Big(\max_a q_*(a) - q_*(A_i) \Big) \tag{400}$$

is just the sum of the opportunity cost at all time steps resulting from not taking the best action.

In the following context we always focus on estimating the cumulative regret of the action sequence generated by a certain algorithm. For simplicity, we always assume that the rewards are random but almost surely bounded taking value in $[0,1]$. This assumption enables us to use simpler concentration bounds for a.s. bounded random variables. WLOG, all those conclusions also work for sub-Gaussian random variables, the connection provided by Hoeffding's lemma.

UCB Algorithm Analysis

Let's provide an estimate for the cumulative regret of UCB. Assume that $T >> K$, the first K actions are devoted in exploring all available actions which result in $O(K)$ contribution to the cumulative regret. Notice that adding this exploration step provides empirically better performance but in the theoretical analysis it's not crucial thus neglected.

The analysis of UCB algorithm starts from an observation on $Q_t(a)$ that it's an unbiased estimator for $q_*(a)$ for

every action a given action sequence $A_1, ..., A_{t-1}$. To see this,

$$\mathbb{E}(Q_t(a)|A_1, ..., A_{t-1}) = \frac{\sum_{i=1}^{t-1} \mathbb{E}(R_i|A_1, ..., A_{t-1})\mathbb{I}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{I}_{A_i=a}} = \frac{\sum_{i=1}^{t-1} q_*(a)\mathbb{I}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{I}_{A_i=a}} = q_*(a) \tag{401}$$

hence we have the reason to believe that $Q_t(a)$ would concentrate near $q_*(a)$. If we fix action a,

$$X_i = \mathbb{I}_{A_i=a}(R_i - q_*(a)) \tag{402}$$

defines a sequence of random variables adapted to the filtration $\{\mathscr{F}_t\}$ where $\mathscr{F}_t$ denotes the collection of information in the game up to time t. Since UCB tells us the way to generate A_t based on all the information before time t, it's clear that $A_t \in \mathscr{F}_{t-1}$ is predictable. Simple calculation tells us

$$\mathbb{E}(X_t|\mathscr{F}_{t-1}) = \mathbb{I}_{A_t=a}[\mathbb{E}(R_t|\mathscr{F}_{t-1}) - q_*(a)] = 0 \tag{403}$$

as a result, the sequence of r.v. $\{X_t\}$ can be seen as a sequence of MG increments. Thinking of concentration inequality for bounded martingale increments, Azuma's inequality is the right tool to use.

Lemma 7 (Azuma's Inequality)**.** *Let $\{S_n\}$ be a martingale with almost surely bounded increments $\forall k, A_k \leq S_k - S_{k-1} \leq B_k$ a.s. for deterministic A_k, B_k, then*

$$\forall a > 0, \mathbb{P}\left(S_n - S_0 \geq a\right) \leq e^{-\frac{2a^2}{\sum_{i=1}^{n}(B_i - A_i)^2}} \tag{404}$$

Apply Azuma's inequality for $S_t = \sum_{i=1}^{t} X_i$ as a MG to see that

$$\mathbb{P}\left(\sum_{i=1}^{t} X_i \geq c\sqrt{\log\frac{2}{\delta} \cdot N_t(a)}\Big| N_t(a)\right) \leq e^{-\frac{2c^2 \log\frac{2}{\delta} N_t(a)}{4N_t(a)}} = e^{-\log\frac{2}{\delta}} = \frac{\delta}{2} \tag{405}$$

since X_i takes value zero if $A_i \neq a$ and $-1 \leq X_i \leq 1$ *a.s.* if $A_i = a$. Since action a has been taken for $N_t(a)$ times given the value of $N_t(a)$, $\sum_{i=1}^{t}(B_i - A_i)^2 \leq 4N_t(a)$. Taking $c^2 = 2$, we get the concentration inequality above. Written in double-sided form

$$\mathbb{P}\left(\left|\sum_{i=1}^{t} X_i\right| \geq c\sqrt{\log\frac{2}{\delta} \cdot N_t(a)}\right) \leq \delta \tag{406}$$

with high probability (at least $1 - \delta$), $\left|\sum_{i=1}^{t} X_i\right| \leq c\sqrt{\log\frac{2}{\delta} \cdot N_t(a)}$ happens.

Notice that

$$\sum_{i=1}^{t} X_i = \sum_{i=1}^{t} \mathbb{I}_{A_i=a}(R_i - q_*(a)) = N_t(a)Q_t(a) - N_t(a)q_*(a) \tag{407}$$

this tells us on fixing action a and time t, with high probability (at least $1 - \delta$) the following concentration bound

holds

$$|Q_t(a) - q_*(a)| \leq c\sqrt{\frac{\log \frac{2}{\delta}}{N_t(a)}} \tag{408}$$

taking union bound w.r.t. $\forall t \in \{1, 2, ..., T\}, a \in \{1, 2, ..., K\}$, with probability at least $1 - KT\delta$ the same bound holds. Set δ' as $\frac{\delta}{KT}$, we have proved the following result.

Lemma 8 (Concentration of $Q_t(a)$)**.** *With probability at least* $1 - \delta$*,*

$$\forall t \in \{1, 2, ..., T\}, \forall a \in \{1, 2, ..., K\}, |Q_t(a) - q_*(a)| \leq c\sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(a)}} \tag{409}$$

Now it's time to prove the regret bound for UCB.

Theorem 10 (UCB Regret Bound)**.** *With probability at least* $1 - \delta$*,*

$$R_T = O\left(\min\left\{\sqrt{KT\log\frac{2KT}{\delta}}, \sum_{a\neq a^*}\frac{\log\frac{2KT}{\delta}}{\Delta_a}\right\} + K\right) \tag{410}$$

where a^* *denotes the best action and* $\Delta_a = q_*(a^*) - q_*(a)$ *denotes the optimality gap of action* a*.*

Remark. *There are three parts in the regret bound. The first one is* K*, which results from the exploration stage and is typically negligible.*

The second part is $\sqrt{KT\log\frac{KT}{\delta}}$*, which is denoted* $\tilde{O}(\sqrt{KT})$*.* $\tilde{O}$ *means the time complexity ignoring logarithmic factors compared to the main factor.*

The third part is $\sum_{a\neq a^*}\frac{\log\frac{KT}{\delta}}{\Delta_a}$*, saying that if the expected rewards of different bandits are very different (large* Δ_a*), UCB works well and vice versa. This term represents the intrinsic complexity of the bandit problem from the perspective of the reward distribution.*

Proof. We neglect the $O(K)$ regret from the exploration stage and work on proving the remaining terms.

From the concentration inequality, with probability at least $1 - \delta$,

$$\forall a, t, q_*(a) \leq Q_t(a) + c\sqrt{\frac{\log\frac{2TK}{\delta}}{N_t(a)}} \tag{411}$$

denote $\{A_t\}$ as the sequence of actions generated by UCB, by definition of UCB,

$$\forall t, q_*(a^*) \leq Q_t(a^*) + c\sqrt{\frac{\log\frac{2TK}{\delta}}{N_t(a^*)}} \leq Q_t(A_t) + c\sqrt{\frac{\log\frac{2TK}{\delta}}{N_t(A_t)}} \tag{412}$$

so UCB generated action value $q_*(A_t)$ is always not too far away from the best action value $q_*(a^*)$

$$\forall t, q_*(a^*) - q_*(A_t) \leq Q_t(A_t) - q_*(A_t) + c\sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(A_t)}} \leq 2c\sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(A_t)}} \tag{413}$$

sum over $t \in \{1, 2, ..., T\}$ to get

$$R_T = \sum_{t=1}^{T} [q_*(a^*) - q_*(A_t)] \tag{414}$$

$$\leq 2c \sum_{t=1}^{T} \sqrt{\frac{\log \frac{2TK}{\delta}}{N_t(A_t)}} \tag{415}$$

$$= 2c\sqrt{\log \frac{2TK}{\delta}} \sum_{t=1}^{T} \frac{1}{\sqrt{N_t(A_t)}} \tag{416}$$

the summation can be simplified here since A_t can only take values in $\{1, 2, ..., K\}$

$$\sum_{t=1}^{T} \frac{1}{\sqrt{N_t(A_t)}} = \sum_{a=1}^{K} \sum_{t:A_t=a} \frac{1}{\sqrt{N_t(a)}} = \sum_{a=1}^{K} \sum_{i=1}^{N_T(a)} \frac{1}{\sqrt{i}} \tag{417}$$

the last equation is from re-ordering the terms w.r.t. the second summation index. Action a must have appeared for $N_T(a)$ times so the i-th time it's appearing, it contributes $\frac{1}{\sqrt{i}}$ to the summation. An easy integral estimation now gives

$$\sum_{a=1}^{K} \sum_{i=1}^{N_T(a)} \frac{1}{\sqrt{i}} \leq \sum_{a=1}^{K} \left(1 + \int_1^{N_T(a)} \frac{1}{\sqrt{x}}\, dx\right) \leq 2 \sum_{a=1}^{K} \sqrt{N_T(a)} \tag{418}$$

since $\sum_{a=1}^{K} N_T(a) = T$, we want to see this summation, apply Cauchy-Schwarz

$$\sum_{a=1}^{K} \sqrt{N_T(a)} \cdot 1 \leq \sqrt{\sum_{a=1}^{K} 1 \cdot \sum_{a=1}^{K} N_T(a)} = \sqrt{KT} \tag{419}$$

combine all inequalities together to see

$$R_T \leq 2c\sqrt{\log \frac{2TK}{\delta}}\sqrt{KT} \tag{420}$$

this proves one bound in the minimum.

For the other bound, the proof is simpler

$$R_T = \sum_{t=1}^{T}[q_*(a^*) - q_*(A_t)] \tag{421}$$

$$\leq \sum_{a\neq a^*}\sum_{1\leq t\leq T, A_t=a}[q_*(a^*) - q_*(a)] \tag{422}$$

$$\leq \sum_{a\neq a^*} N_T(a)\cdot\Delta_a \tag{423}$$

for $\forall a \neq a^*$, consider the last time action a is selected by UCB (e.g. at time t), then the upper confidence bound of a must exceed that of a^*

$$Q_t(a) + c\sqrt{\frac{\log\frac{2tK}{\delta}}{N_t(a)}} \geq Q_t(a^*) + c\sqrt{\frac{\log\frac{2tK}{\delta}}{N_t(a^*)}} \geq q_*(a^*) \tag{424}$$

the second inequality happens with probability at least $1-\delta$. With high probability, we also have

$$|Q_t(a) - q_*(a)| \leq c\sqrt{\frac{\log\frac{2TK}{\delta}}{N_t(a)}} \tag{425}$$

since it's the last time action a is selected, $N_t(a) = N_T(a)$ so

$$\forall a\neq a^*, 0 < \Delta_a = q_*(a^*) - q_*(a) \leq 2c\sqrt{\frac{\log\frac{2TK}{\delta}}{N_T(a)}}, N_T(a) \leq 4c^2\frac{\log\frac{2TK}{\delta}}{\Delta_a^2} \tag{426}$$

as a result,

$$R_T \leq \sum_{a\neq a^*} 4c^2\frac{\log\frac{2TK}{\delta}}{\Delta_a} \tag{427}$$

concludes the proof.

□

Remark. *This theorem shows the* ***sublinear*** $\tilde{O}(\sqrt{KT})$ *cumulative regret of UCB. In contrast,* ε*-greedy strategy still has linear cumulative regret* $O(T)$ *(try to argue this).*

From the proof, it's clear that $1-\delta$ *stands for the confidence of the bound,* c *has something to do with the sub-Gaussian constant or the width of concentration region of random variables.*

See Fig. 5, 6 that numerically shows what we have proved above.

Linear Bandit Problem and LinUCB