NOTES FOR REINFORCEMENT LEARNING

Yao Yao
realyao21002@gmail.com

1 Multi-arm Bandit: only one state

- ε-greedy. To find out the best arm by statistical analysis. At each step/round/time, choose the current best arm with probability (w.p.) $(1-\varepsilon)$ (**exploitation**) or randomly choose an arm w.p. ε (**exploration**). However, there is always a gap in regret.
- Upper Confidence Bound (UCB) method. To find out the best arm by statistical analysis, too. At each step/round/time, without a probability, choose the largest 'potential' arm that optimize the value function added with a **Chernoff-Hoeffding bound term**. This is to enable the reward gap to approach 0.
- Gradient-bandit method. To decide a distribution for pulling arms. A naive version of policy gradient method.

Elements: K-arm bandit, with mean $\mu_1,\dots,\mu_K$, action $A_t \in \mathscr{A} = \{1,\dots,K\}$ and R_t. At each time step compute estimation of mean - using rewards accumulated **before time T**

$$Q_T(a) = \frac{\Sigma_{t=1}^{T-1} R_t \mathbf{1}_{A_t=a}}{N_T(a)}, \text{ where } N_T(a) = \Sigma_{t=1}^{T-1}\mathbf{1}_{A_t=a}.$$

1.1 ε-greedy

Randomly pick $A_t \sim U\{1,\dots,K\}$ w.p. $\varepsilon \in (0,1)$, or pick $A_t = argmax_{a\in\mathscr{A}} Q_T(a)$ w.p. $1-\varepsilon$.
In application - generate an R.V.$\sim U(0,1)$ to ensure the exploration rate is $P(R.V. \leq \varepsilon) = \varepsilon$.

$$: \text{if } R.V. \leq \varepsilon, \text{ randomly choose an arm.}$$

There is an iteration in Q -

$$Q_{N+1} = \frac{(N-1)\cdot Q_N + R_N}{N} = Q_N + \frac{R_N - Q_N}{N},$$

where $N+1$ is the local time when **an arm** is being estimated, i.e. N is the last time an arm is pulled.

Bernoulli Bandit $r_i \sim Bernoulli(\mu_i)$

Define an R.V. $Y \sim U(0,1)$ and

$$X = \begin{cases} 1, & Y < \mu \\ 0, & Y \geq \mu \end{cases}$$

to ensure the reward of a Bernoulli arm is $r_i = 1$ w.p. μ_i

$$: P(X=1) = P(Y<\mu) = \mu,\ P(X=0) = P(Y \geq \mu) = 1-\mu.$$

Hyper Param Analysis

The larger ε, the more exploration, the faster it converges, but not necessarily converge to good solutions. The more exploitation, the more cumulative rewards, but the slower it converges.
Empirical good choice $\varepsilon = 0.1$ or $\varepsilon = 0.05$.

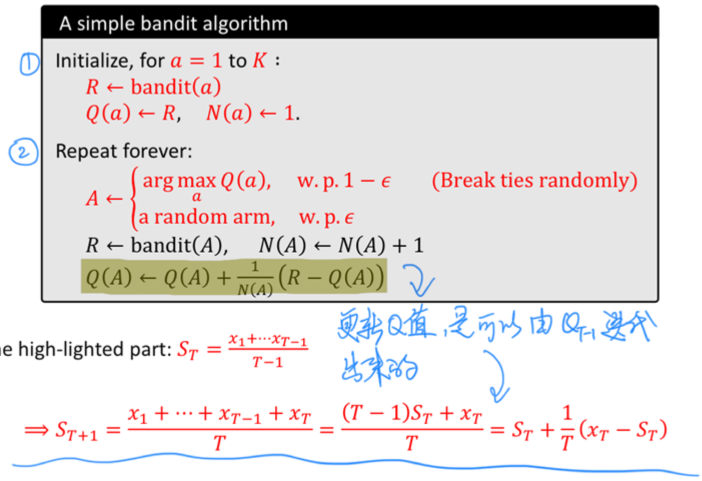

Figure 1: ε-greedy

1.2 Regret analysis: ε-greedy results in a gap

A measure of 'expected loss' compared to always pulling the optimal arm.

$$Regret(T) = T\mu^* - E[\Sigma_{k=1}^{T} R_{A_k}] = T\mu^* - E[\Sigma_{k=1}^{T} \mu_{A_k}] = T\mu^* - E[\Sigma_{a\in\mathscr{A}} N(a)\mu_a],$$

where $\mu^* = max_i\{\mu_i\}, \mu_{A_k} =$ estimated mean for arm $A_k =$ some a, $N(a) =$#times arm a is pulled.

If regret $\to 0$, then

$$Regret = T\mu^* - E[\Sigma_{k=1}^{T} \mu_{A_k}] \to 0,$$

$$AvgReward(T) = \frac{E[\Sigma_{k=1}^{T} \mu_{A_k}]}{T} \to \mu^*,$$

$$\frac{Regret(T)}{T} = \mu^* - AvgReward(T) \to 0.$$

In ε-greedy, assuming arm 10 as the true best

$$Regret = T\mu_{10} - \frac{9\varepsilon}{10} T\bar{\mu}_{other} - \frac{10 - 9\varepsilon}{10} T\mu_{10} = \frac{9\varepsilon T}{10}(\mu_{10} - \bar{\mu}_{other}),$$

$$AvgRegret(T) = \frac{Regret(T)}{T} = \frac{9\varepsilon}{10}(\mu_{10} - \bar{\mu}_{other}) > 0.$$

There is a non-diminishing gap.

A Naive Approach

Introducing **Chernoff-Hoeffding bound**:

$$P(|Q_n(a) - \mu_a| \leq \delta) \geq 1 - 2e^{-2n\delta^2},$$

where n is the number of times arm a is pulled, μ_a is the true mean reward of arm a.

Then the confidence interval $\mu_a \in [Q_n(a) - \delta, Q_n(a) + \delta]$ **is what we focus on.** Suppose the gap between the true best and the true second best is known as Δ. Let $1 - 2e^{-2n(\Delta/3)^2} \geq 1 - p$. then $n \geq \frac{9\ln(2/p)}{2\Delta^2}$ would satisfies

$$\forall a \neq 10, Q_n(a) \leq \mu_a + \Delta/3 < \mu_{10} - \Delta/3 \leq Q_n(10) \text{ w.p. at least } 1 - p,$$

to ensure the true best arm 10 has the highest $Q_n(10)$ w.p. $1-10p$. There are 10 inequalities, where any of them does not hold w.p.

$$P(\text{any does not hold}) = P(\{\text{1st does not hold}\} \cup \cdots \cup \{\text{10th does not hold}\}) \leq 10p.$$

Then the approach is **exploration-then-exploitation** - pull each time for n times, and then always choose the arm with best estimated reward for $T-nk$ times. Note that

$$\text{let } p = 1/T, \text{ then } n = [\frac{9\ln(2T)}{2\Delta^2}].$$

Such approach has drawbacks that the user needs to know T as the number of trials, which also makes the exploration and exploitation completely separated and lack flexibility, and Δ as the gap.

1.3 The UCB Approach

This approach helps the Chernoff-Hoeffding bound inequality holds, to make the

$$\frac{Regret(T)}{T} = \mu^* - AvgReward(T) = O(1/\sqrt{T}) \to 0.$$

At each time step, choose an arm to maximize the **Upper Confidence Bound** of the previous confidence interval. Here the selected $\delta_t(a)$ varies at each time across arms, and is $\sqrt{\frac{2\ln t}{N_t(a)}}$ such that

$$1 - 2e^{-2n\delta_t^2} = 1 - 1/t^4.$$

The larger the time t grows, the higher the probability that $Q_t(a) + \delta_t(a)$ provides a true upper

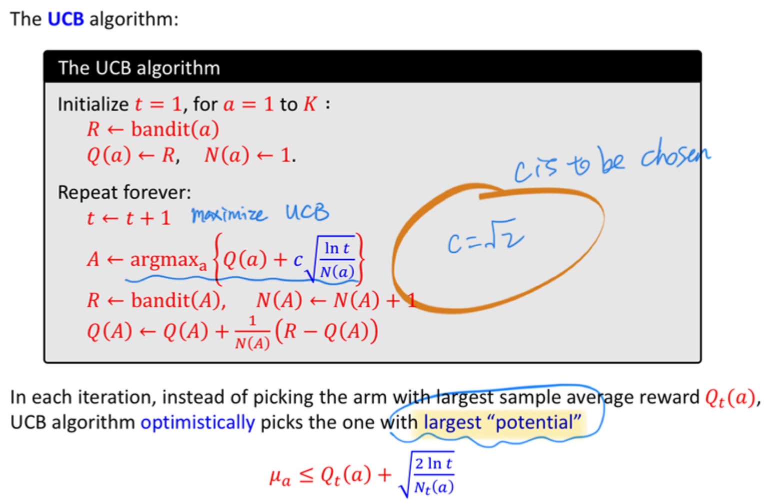

Figure 2: The UCB approach

bound for the true mean μ_a, although δ_t is changing through time.
The larger the pull number $N_t(a)$ grows, the tighter the upper bound $Q_t(a) + \delta_t(a)$ is compared with the true mean μ_a. Because $\delta_t(a)$ would decrease, letting the upper bound be closer to μ_a.
In each iteration, instead of picking the arm with largest $Q_t(a)$, UCB alg optimistically picks the one with **the largest 'potential'**.

- Every arm will be pulled infinitely. $\delta_t(a)$ gets larger even if arm a is never pulled, then arm a will be pulled until its ucb becomes the largest.

- Pulled suboptimal arm will be played less frequently, because $N_t(a)$ gives a penalty in $\delta_t(a)$.
- Overestimate rarely pulled arms. In particular, if $N_t(a) = 0$, then its ucb $= \infty$, making the unpulled arm pulled first. (However, in practice, we always initialize the first pulling of each arm.)

1.4 Gradient-Bandit

Direct optimization to optimize the distribution for pulling each arm. A naive version of the policy gradient method.

$$
\begin{aligned}
&min_\pi - \Sigma_{a=1}^K \mu_a \pi_a \\
&s.t. \pi \geq 0 \\
&\Sigma_a \pi_a = 1
\end{aligned}
$$

Where, the objective is the negative expected reward when pulling arms according to the distribution π. To ensure $\pi_a \in [0,1]$, introduce a parameter θ_a for each arm and let

$$\pi_a(\theta) = \frac{exp(\theta_a)}{\Sigma_{b=1}^K exp(\theta_b)}.$$

The objective becomes $min_\theta \{f(\theta) : -\Sigma_{a=1}^K \mu_a \pi_a(\theta)\}$.
In such method, we update the distribution using gradient descent, which utilize the unknown μ_a. Although this is unknown, we can still have an unbiased estimator of the gradient without knowing μ_a. Fortunately,

$$\frac{\partial f(\theta)}{\partial \theta_a} = \Sigma_{b \neq a}^K \mu_b \pi_a(\theta)(\pi_b(\theta) - 0) + \mu_a \pi_a(\theta)(\pi_a(\theta) - 1) = E_{b \sim \pi(\theta), R = bandit(b)}[R \cdot (\pi_a(\theta) - \mathbf{1_{b=a}})].$$

The estimator means that, when you pull an arm b, calculate the partial derivative using the expectation for each arm a.
Then, conduct gradient descent step for each $\theta_a^{t+1} = \theta_a^t - \eta_t R_b(\pi_a^t - \mathbf{1_{a=b}})$, where π_a^t is computed at the beginning of each iteration.

Gradient-bandit algorithm

The gradient-bandit algorithm

Initialize $t = 0$, $\boldsymbol{\theta}^0 \in \mathbf{R}^K$

Repeat forever:

- Compute distribution $\boldsymbol{\pi}^t$: $\pi_a^t = \frac{\exp(\theta_a)}{\sum_{b=1}^K \exp(\theta_b)}, \forall a \in \{1, \cdots, K\}$
- Sample an arm $A \sim \boldsymbol{\pi}^t$
- Get reward $R \leftarrow$ bandit(A)
- For $a = A$, update: $\theta_a^{t+1} = \theta_a^t - \eta_t R(\pi_A^t - 1)$
- For $a \neq A$, update: $\theta_a^{t+1} = \theta_a^t - \eta_t R \pi_a^t$
- $t \leftarrow t + 1$

η_t是步长，$\pi_A^t - 1_{A=a}$是估计的梯度

Figure 3: Gradient Bandit

- Taking smaller steps will converge to higher final average reward, but with slower convergence speed.
- Better than ε-greedy.

Add base-line to improve GB

The base-line is

$$\bar{R}_T = \frac{\Sigma_{t=1}^{T} R_t}{T}, \text{ to update baseline, } \bar{R}_{T+1} = \frac{\bar{R}_T \cdot T + R_{T+1}}{T+1} = \bar{R}_T + \frac{R_{T+1} - \bar{R}_T}{T+1}.$$

When conducting gradient descent,

$$\theta_a^{t+1} = \theta_a^t - \eta_t (R_b - \bar{R}_{t+1})(\pi_a^t - \mathbf{1_{a=b}}), \text{ when choosing arm } b.$$

Adding a baseline **does not change the unbiased-ness**.

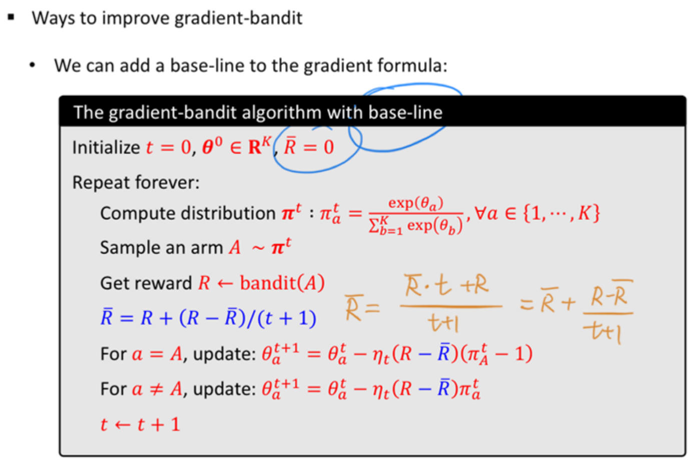

Figure 4: gradient bandit with baseline

$$\nabla f_B(\theta) = \nabla f(\theta), \text{ and } \Sigma_{a=1}^K \pi_a(\theta) = 1 \Rightarrow \Sigma_{a=1}^K \frac{\partial \pi_a(\theta)}{\partial \theta_a} = 0.$$

Which means equally shifting the rewards of each arm by a constant B will not change the relative performance of the arms and will not change the gradients.
Adding an appropriate base-line will often improve the (overall) performance.

2 Markov Decision Process

The (finite) Markov Decision Process is written as

$$\mathscr{M}(\mathscr{S}, \mathscr{A}, r, \gamma, P).$$

2.1 MDP Elements

- State space $\mathscr{S}$ s.t. $s \in \mathscr{S}$ is a set.
- Action space $\mathscr{A}$ s.t. $a \in \mathscr{A}$. To distinguish between the action choice a and the real action $\hat{a}$.
- Reward function $r(\cdot, \cdot)$. Results from $s \in \mathscr{S}$ and $a \in \mathscr{A}$, $E[R] = r(s,a)$ or simplified $R = r(s,a)$.
- Transition probability P. A probability $P(s'|s,a)$, so $s' \sim P(\cdot|s,a)$. For each action a, we can also write $P_a(s,s')$, so $s' \sim P_a(s,\cdot)$.

- Discount factor $\gamma \in (0,1)$. We generally **don't expect a terminating state**, then a discount on future reward should be conducted. We would like to maximize the **expected discounted cumulative reward**
$$E[\Sigma_{t=0}^{\infty}\gamma^t R_t] \text{ with } E[R_t|s_t,a_t] = r(s_t,a_t).$$
In most cases, γ is close to 1, e.g., $\gamma = 0.95$.

- Policy $\pi : \mathscr{S} \to \Delta_{\mathscr{A}}$ is a mapping from the state space to a **distribution over the action space**. Under a policy π, the probability of an action a given a state s gives $a \sim \pi(\cdot|s)$. Given an **initial state distribution** ξ, we would like to find the optimal policy by maximizing
$$V^{\pi}(\xi) := E[\Sigma_{t=0}^{\infty}\gamma^t \cdot r(s_t,a_t)|s_0 \sim \xi, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t,a_t)].$$

2.2 Estimating Cumulative Reward: Monte Carlo

Approach 1: generate random horizon, without computing γ

At each epoch, generate a random horizon H s.t. $H = X - 1$ with $X \sim Geom(1-\gamma)$, then $P(H \geq k) = \gamma^k$. After that, sample a finite sequence $\{s_t,a_t,R_t\}_{t=0}^{H}$ and compute $\tilde{V}_{epoch} = \Sigma_{t=0}^{H} R_t$. After epochs,take the average $\bar{V} = \frac{1}{h}\Sigma_{epoch=1}^{h}\tilde{V}_{epoch}$.
The estimator $\tilde{V}$ is unbiased,

$$E[\tilde{V}] = E[\Sigma_{t=0}^{\infty}\mathbf{1}_{t\leq H}R_t] = \Sigma_{t=0}^{\infty}P(H \geq t)E[R_t] = \Sigma_{t=0}^{\infty}\gamma^t E[R_t] = V^{\pi}(\xi).$$

Expected trajectory length is $E[H]+1 = \frac{1}{1-\gamma}$.

Alternative estimator: fixed horizon

Take a fixed horizon $H = [const \cdot \frac{ln(1/\varepsilon)}{1-\gamma}]$. Then sample a sequence $\{s_t,a_t,R_t\}_{t=0}^{H}$ and compute $\tilde{V} = \Sigma_{t=0}^{H}\gamma^t R_t$.
This estimator is biased, but the bias is very small:

$$|E[\tilde{V}] - V^{\pi}(\xi)| \leq P(\varepsilon).$$

Complimentary knowledge on horizon H

There are two types of horizon, relatively for finite and infinite MDP:

$$H = \begin{cases} \text{terminating state}, & \text{finite MDP} \\ \tau, & \text{infinite MDP} \end{cases}.$$

Fixed horizon may lead to an unexpected terminating state.

3 Value Functions

3.1 State Value Function

Definition

As the cumulative reward function is

$$V^{\pi}(\xi) = E_{s_0\sim\xi, a_t\sim\pi(\cdot|s_t), s_{t+1}\sim P(\cdot|s_t,a_t)}\Sigma_{t=0}^{\infty}\gamma^t r(s_t,a_t),$$

when the initial state s_0 is known and instead denoted by s, we can compute the state value function from such initial s.

$$V^\pi(s) = E_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t,a_t)}[\Sigma_{t=0}^\infty \gamma^t r(s_t,a_t)|s_0 = s].$$

Note that

$$V^\pi(\xi) = E_{s_0 \sim \xi}[V^\pi(s_0)].$$

Bellman Equation

$$\begin{aligned} V^\pi(s) &= \Sigma_{a\in A}\pi(a|s)\Sigma_{s'\in S}P(s'|s,a)[r(s,a)+\gamma V^\pi(s')] \\ &= \Sigma_{a\in A}\pi(a|s)[r(s,a)+\Sigma_{s'\in S}P(s'|s,a)\gamma V^\pi(s')], \forall s. \end{aligned}$$

In expectation form,

$$V^\pi(s) = E_{a\sim\pi(\cdot|s), s'\sim P(\cdot|s,a)}[r(s,a)+\gamma V^\pi(s')], \forall s.$$

$\mathbf{V}^\pi(\mathbf{s})$ in Terms of $\mathbf{Q}^\pi(\mathbf{s},\mathbf{a})$

$$V^\pi(s) = \Sigma_{a\in A}\pi(a|s)Q^\pi(s,a) = E_{a\sim\pi(\cdot|s)}Q^\pi(s,a).$$

3.2 State Action Value Function (Q Function)

Definition

$$Q^\pi(s,a) := E_{s_{t+1}\sim P(\cdot|s_t,a_t), a_t\sim\pi(\cdot|s_t)}[\Sigma_{t=0}^\infty \gamma^t r(s_t,a_t)|s_0 = s, a_0 = a], \forall s,a.$$

Bellman Equation

$$\begin{aligned} Q^\pi(s,a) &= \Sigma_{s'\in S}P(s'|s,a)\Sigma_{a'\in A}\pi(a'|s')[r(s,a)+\gamma Q^\pi(s',a')] \\ &= r(s,a)+\Sigma_{s'\in S}\Sigma_{a'\in A}P(s'|s,a)\pi(a'|s')\gamma Q^\pi(s',a'). \end{aligned}$$

In expectation form,

$$Q^\pi(s,a) = r(s,a)+\gamma E_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')}[Q^\pi(s',a')].$$

$\mathbf{Q}^\pi(\mathbf{s},\mathbf{a})$ in Terms of $\mathbf{V}^\pi(\mathbf{s})$

$$\begin{aligned} Q^\pi(s,a) &= \Sigma_{s'\in S}P(s'|s,a)[r(s,a)+\gamma V^\pi(s')] \\ &= r(s,a)+\gamma E_{s'\sim P(\cdot|s,a)}[V^\pi(s')]. \end{aligned}$$

3.3 Optimal Value Function

To obtain the optimal policy π^*, maximize the cumulative reward function

$$max_\pi V^\pi(\xi) := E_{\xi,\pi,P}[\Sigma_{t=0}^\infty \gamma^t r(s_t,a_t)].$$

The obtained π^* will **simultaneously optimize all the value functions and Q functions**.
In such a policy,

$$\pi^*(a|s) = \begin{cases} 1, & a = a^*(s) \\ 0, & a \neq a^*(s) \end{cases}.$$

Where

$$a^* = argmax_{a \in A}\{Q^*(s,a)\} = argmax_{a \in A}\{r(s,a) + \gamma \Sigma_{s' \in S} P(s'|s,a) V^*(s')\}.$$

Optimal State Value Function

$$V^*(s) = max_{a \in A}\{r(s,a) + \gamma \Sigma_{s' \in S} P(s'|s,a) V^*(s')\} = max_{a \in A}\{Q^*(s,a)\}.$$

Optimal State Action Value Function

$$\begin{aligned} Q^*(s,a) &= r(s,a) + \gamma \cdot max_{a' \in A}\{\Sigma_{s' \in S} P(s'|s,a) Q^*(s',a')\} \\ &= r(s,a) + \gamma \Sigma_{s' \in S} P(s'|s,a) \cdot max_{a' \in A}\{Q^*(s',a')\} \end{aligned}$$

3.4 Notes on Solving the Optimal Policy

Note: Solving **Bellman Equations** for $V^*(s)$ or $Q^*(s,a)$ would derive the optimal policy π^*. However, how to solve those Bellman Equations depends on premise assumptions.

- If we assume known transitions and terminating state (i.e. with a horizon), we use dynamic programming, which problem is always called **planning**. This is the focus of section 4.
- If we meet with an infinite-horizon MDP without terminating state, we use value iteration or Q iteration.

4 Finite Horizon MDP: Dynamic Programming

The finite horizon maybe 1) time horizon, or 2) terminating state. (see 2.2)
Note: When setting up, invalid action will be presented by $reward = -\infty$. In practice, although all the states share the same action set, it is OK for different state to have different action sets.

4.1 Example: Grid World Navigation - Terminating State

- State $s = (i, j)$ as the position.
- Action $a \in \{R, D\}$.
- Transition. The transition $s, a \to s'$ is deterministic, i.e. $P(s'|s,a) = 1$. Therefore, to simplify, use the transition mapping $s' = trans(s,a)$. Note that

  $$trans((i,j,),R) = (i, min\{4, j+1\}),$$

  $$trans((i,j),D) = (min\{4, i+1\}, j).$$

  Then it can be seen that state $(4,4)$ is absorbing, which state once hit will not change forever.
- Reward $r(s,a)$ should take the **penalty for invalid state** and take the **terminating state** into consideration.

  $$r((i,j),R) = w[trans((i,j),R)] - \infty \cdot \mathbf{1}_{j=4},$$

  $$r((i,j),D) = w[trans((i,j),D)] - \infty \cdot \mathbf{1}_{i=4}$$

  (where w is the reward number on a grid (i,j)). For terminating state,

  $$r((4,4),a) = 0, \forall a \in \{R, D\}.$$

  Note: once hit the terminating state, the reward received afterwards would be 0 forever.

Solving Bellman Equations

In such a problem, the probability in the original Bellman Equations would be deterministic as 1 or 0, and we also set $\gamma = 1$ -

$$\begin{aligned} V^*(s) &= max_{a\in A}\{r(s,a) + \gamma V^*(s')\} \\ &= max_{a\in A}\{r(s,a) + V^*(trans(s,a))\}. \end{aligned}$$

In order to conduct **dynamic programming**, start from the terminating state $(4,4)$ with $V^*(4,4) = 0$. Then, in the Bellman Equation above, the $r(s,a)$ and the $V^*(s')$ are all known, enabling us to solve the the previous state $V^*(s) = V^*(4,3)$ and $V^*(s) = V^*(3,4)$.
Solving those equations yield the optimal policy -

$$\pi^*(a^*|s) = 1, \forall s.$$

Note: for some state, we would yield multiple optimal action a^*, then randomly choose one. In this example, $\pi^*(\cdot|1,1)$ can be any distribution, for R and D are equally optimal.

4.2 Time Horizon Instead of Terminating State

Horizon of $t = H$ can be view as a special case of MDP with terminating state. **Time can be encoded into the states,** i.e., let $s = (\hat{s},t) \sim S = \hat{S} \times N$ where $\hat{s} \in S$ and $t \in N$.
In the previous grid world example, state incorporating time is not necessary.
Remind: state incorporating into time information is always by default in many papers, which use π^t to embrace such.
Note: in such cases, all states with $t = H$ are terminating states. Transitions become $P(\cdot|s_t,t,a_t)$. Also, take $\gamma = 1$, then the cumulative reward function will be

$$V^\pi(\xi) = E_{\xi,\pi,P}[\Sigma_{t=0}^{H-1} r(s_t,a_t)].$$

Maximizing the function above would yield an optimal policy that **depends on time** $a_t \sim \pi(\cdot|s_t,t)$.
Note: whether taking the optimal policy or not,

$$V^\pi(\hat{s},H) = V^*(\hat{s},H) = 0, \forall \hat{s} \in \hat{S};$$

$$Q^\pi(\hat{s},H,a) = Q^*(\hat{s},H,a) = 0, \forall \hat{s} \in \hat{S}, a \in A.$$

Therefore, the Bellman Equations can be solved **level by level**, i.e., **at each time step**. Knowing

$$V^*(\hat{s},H-1) = max_{a\in A}\{r(\hat{s},a) + \gamma \Sigma_{\hat{s}'\in\hat{S}} P(\hat{s}'|\hat{s},H-1,a) V^*(\hat{s}',H)\}.$$

To conduct the **dynamic programming**, also start from the terminating state, i.e., level H - $V^*(\hat{s},H) = 0, \forall \hat{s} \in \hat{S}$. Then, assuming known reward function $r(\hat{s},a)$ and transition probability $P(\hat{s}'|\hat{s},H-1,a)$, as well as the solved $V^*(\hat{s}',H)$, we can solve the previous level $H-1$ states.

4.3 Example: Inventory Management - Time Horizon

- State $s = (n,t)$ as the inventory **at end of** the month t. Assuming no inventory at the beginning of month 1, i.e., $s_0 = (0,0)$ as the initial state. Therefore, $S = \{(n,t) : n \in \{0,\ldots,4\}, t \in \{1,2,3,4\}\} \cup \{(0,0)\}$.

- Action $a \in \{0, \ldots, 5\}$ as the number of productions **at the beginning of next moth**. To satisfy the **demand and capacity constraints**,

$$a + n \geq d_{t+1}, a + n - d_{t+1} \leq K = 4.$$

- Transition is also **deterministic**. Simply use $s' = trans(s, a)$ as the transition mapping for $t = 0, 1, 2, 3$ (**not considering transition for** $t = 4$). Let

$$trans((n,t),a) = (proj_{[0,4]}(n + a - d_{t+1}), t + 1)$$

- Reward $r((n,t),a)$ in terms of $g = n + a - d_{t+1}$.

$$r((n,t),a) = -0.5 \times Proj_{[o,4]}(g) - 1 \times a - 3 \times \mathbf{1}_{a>0} - \infty \times \mathbf{1}_{g \notin [0,4]}.$$

  to **minimize the total cost**, no revenue is considered, but a penalty for over-capacity inventory is imposed.

- Discount factor $\gamma = 1$ also.
- Time horizon $H = 4$.

4.4 Example: Coin Picking Game - nondeterministic transitions

- State $s = (i, j)$ as the remaining coin from number i to number j. After each step two coins will be picked, therefore the number of remaining coins is even. $S = \{(1,2), (2,3), (3,4), (4,5), (5,6), (1,4), (2,5), (3,6), (1,$ The terminating state is the null subset O.
- Action $a \in \{L, R\}$.
- Transition is **not deterministic**. Consider from the terminating state:

$$P(O|O,a) = 1, \forall a;$$

$$P(O|(i,i+1),a) = 1, \forall a;$$

$$P((i+2,j)|(i,j),L) = P((i+1,j-1)|(i,j),L) = 0.5,$$

$$P((i,j-2)|(i,j),R) = P((i+1,j-1)|(i,j),R) = 0.5.$$

- Reward is the value of coin picked.

$$r(O,L) = r(O,R) = 0;$$

$$r((i,j),L) = v_i, r((i,j),R) = v_j.$$

The Bellman Equations with $\gamma = 1$

$$V^*(s) = max_{a \in A}\{r(s,a) + \Sigma_{s' \in S} P(s'|s,a) V^*(s')\}.$$

To conduct dynamic programming, start from the terminating state -

$$V^*(O) = 0.$$

Then for the previous states, solve all the $V^*(i, i+1)$.

5 Infinite Horizon MDP: Value and Q Iterations

The infinite-horizon MDPs are without any terminating state **(transition dynamics are still known)**, and the corresponding approach is value iteration and Q iteration.
Value iteration and Q iteration are iterations using Bellman equations and Bellman operators for

$$V^\pi(\cdot),\ V^*(\cdot),\ Q^\pi(\cdot), \text{ and } Q^*(\cdot).$$

Recall that solving V^* and Q^* would derive the optimal policy π^*.

5.1 Fixed-point problem of Bellman operators

Using Bellman operators, we try to solve fixed-point problems including

- Value iteration.

$$V^\pi = T^\pi V^\pi$$
$$V^* = T^* V^*,$$
$$\text{where } T : R^S \to R^S,$$

  and we can write the function for $[T^\pi V^\pi]_s$ and $[T^* V^*]_s$ as

$$[T^\pi V^\pi]_s := \Sigma_{a\in A}\pi(a|s)[r(s,a)+\gamma\Sigma_{s'\in S}P(s'|s,a)V^\pi_{s'}] = V^\pi_s$$
$$[T^* V^*]_s := \max_{a\in A}\{r(s,a)+\gamma\Sigma_{s'\in S}P(s'|s,a)V^*_{s'}\} = V^*_s.$$

- Q iteration.

$$Q^\pi = T^\pi Q^\pi$$
$$Q^* = T^* Q^*,$$
$$\text{where } T : R^{S\times A} \to R^{S\times A},$$

  and we can write the function for $[T^\pi Q^\pi]_{s,a}$ and $[T^* Q^*]_{s,a}$ as

$$[T^\pi Q^\pi]_{s,a} := r(s,a)+\gamma\Sigma_{s'\in S}P(s'|s,a)\Sigma_{a'\in A}\pi(a'|s')Q^\pi_{s',a'} = Q^\pi_{s,a}$$
$$[T^* Q^*]_{s,a} := r(s,a)+\gamma\Sigma_{s'\in S}P(s'|s,a)\max_{a'\in A}\{Q^*_{s'a'}\} = Q^*_{s,a}.$$

5.2 Fixed-point Solution: Contraction Property

Let $0<\gamma<1$, the mapping $f(\cdot)$ is a $\gamma-$ **contraction** under the norm $||\cdot||$ if

$$||f(x)-f(y)|| \le \gamma||x-y||, \forall x,y.$$

Subsequently, the solution to the fixed-point problem $x = f(x)$ can be obtained by the **fixed-point iteration**.
Given the fixed-point problems of value and Q iterations, it can be proved that the Bellman operators are $\gamma-$ **contraction** under the **infinity** norm, which is defined as $||x||_\infty := \max_i |x_i|$. Because they are $\gamma-$ contraction, the iteration will converge to the true solutions.

$$||x^t - x^*|| = ||f(x^{t-1}) - f(x^*)|| \le \gamma||x^{t-1}-x^*|| \le \gamma^2||x^{t-2}-x^*|| \cdots \le \gamma^t||x^0-x^*||,$$
$$\Rightarrow \lim_{t\to\infty} x^t = x^*.$$

Value Iteration

Start from arbitrary an $V^0 \in \mathbf{R}^S$ and set $t = 0$
Based the purpose, set $T = T^\pi$ or $T = T^*$

Repeat forever:

$V^{t+1} = TV^t$

$t \leftarrow t + 1$

Q Iteration

Start from arbitrary an $Q^0 \in \mathbf{R}^{S \times A}$ and set $t = 0$
Based the purpose, set $T = T^\pi$ or $T = T^*$

Repeat forever:

$Q^{t+1} = TQ^t$

$t \leftarrow t + 1$