

Artificial Intelligence

CS 165A

May 23, 2022

Instructor: Prof. Yu-Xiang Wang

Today

- Exploration (Part 2)
- Reinforcement Learning

Notes

- Instructor OH is on 3pm Tuesday from this point onwards!
- Project 3 (Learning PACMAN) on the course website!
 - The MDP part (Q1,Q2,Q3) is already ready for you to complete
 - The RL algorithm part (Q6 onwards this week's lectures / next Tuesday)
- Grading for leaderboard / Report for Project 1 almost done
- Grades for Project 2 published on Gradescope
 - You may still submit for partial credits
 - Challenge the autograder, request TA for a manual evaluation if you believe something is wrong with the autograder.

Recap: Multi-arm bandits: Problem setup

- No state. k-actions $a \in \mathcal{A} = \{1, 2, \dots, k\}$

- You decide which arm to pull in every iteration

$$A_1, A_2, \dots, A_T$$

- You collect a cumulative payoff of $\sum_{t=1}^T R_t$

- The goal of the agent is to maximize the expected payoff.
 - For future payoffs?
 - For the expected cumulative payoff?

Recap: How do we measure the performance of an **online learning agent**?

- The notion of “Regret”:
 - I wish I have done things differently.
 - Comparing to the best actions in the hindsight, how much worse did I do.

- For MAB, the regret is defined as follow

$$\underbrace{T \max_{a \in [k]} \mathbb{E}[R_t | a]}_{\text{Oracle}} - \underbrace{\sum_{t=1}^T \mathbb{E}_{a \sim \pi} [\mathbb{E}[R_t | a]]}_{\substack{\sum_{t=1}^T \mathbb{E}[R_t | A_t] \\ \text{your performance}}} = \underbrace{\frac{3}{4} T}_{\text{“no regret”}} = \underbrace{O(T)}$$

Recap: MAB Algorithms

- Idea: Plug-in estimate of the reward value
- Greedy: Regret = $O(T)$
- Explore-first: Regret = $O(T^{2/3})$
- More algorithms today

Recap: Hoeffding's inequality and how quickly empirical averages converge to the mean.

$$\bar{X}_n \xrightarrow{n \rightarrow \infty} \mathbb{E}[X_1]$$

- Statistics is about using **samples** from a distribution to infer the properties of the distribution itself (**population**)

- $X_1, X_2, X_3, \dots, X_n \sim P$

- (Simplified version of the) Hoeffding's Inequality:**

- with high probability,

$$\left| \frac{1}{n} \sum_{i=1}^n X_i - \mathbb{E}[X_1] \right| = O(1/\sqrt{n})$$

ϵ
 $n \geq \frac{1}{\epsilon^2}$
 $(\bar{X} - \frac{\epsilon}{\sqrt{n}}, \bar{X} + \frac{\epsilon}{\sqrt{n}})$

*If you try each arm multiple times, then you have a good idea what the expected reward is.

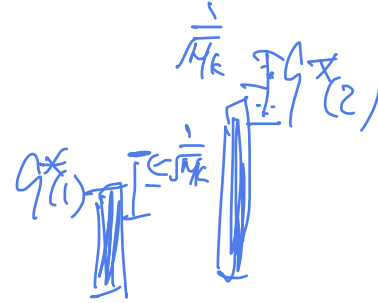
This lecture

- More algorithms for exploration under multi-armed bandits
- Contextual bandits
- Reinforcement Learning algorithms

Recap: How does Explore First work?

- First try out all arms each for N/k times.

$$\forall a \quad \left| \underset{\substack{\uparrow \\ \text{estimate}}}{\hat{q}(a)} - \underset{\substack{\uparrow \\ \text{true}}}{q^*(a)} \right| \leq \frac{C}{\sqrt{N/k}}$$



- Then commit to one of the arm

$$\hat{a}^* = \underset{a \in \{1, 2, \dots, k\}}{\text{argmax}} \hat{q}(a), \text{ play } a^* \text{ for } t = N+1, N+2, \dots, T$$

- Key message: You *per-step* regret is proportional to how well you can estimate the value of that arm.

$$N + (T-N) \cdot O\left(\frac{1}{\sqrt{N/k}}\right) = O\left(T^{\frac{2}{3}} k^{\frac{1}{3}}\right)$$

$$N = T^{\frac{2}{3}} k^{\frac{1}{3}}$$

ϵ -Greedy strategy: one way to balance exploration and exploitation

- You choose with probability $1 - \epsilon$

$$A_t \doteq \operatorname{argmax}_a Q_t(a),$$

Handwritten annotations:
- A blue arrow points from R_t to the Q_t term in the equation.
- A blue box is drawn around the $Q_t(A_{t-1})$ term, with the handwritten text "update $Q_t(A_{t-1})$ " written below it.
- A blue arrow points from the Q_t term to the a term.

- With probability ϵ , choose an action **uniformly at random!**
 - Including the argmax.

- Carefully choose ϵ parameter.

Let's analyze ϵ -Greedy!

- By Hoeffding's inequality, at every t
 - w.h.p. each arm is chosen at least $0.5 \epsilon t / k$ times.
 - w.h.p.,

$$0.5 \frac{\epsilon t}{k}$$

$k a$
 $k t$

- Regret bound is

Let's analyze ϵ -Greedy!

- By Hoeffding's inequality, at every t
 - w.h.p. each arm is chosen at least $0.5 \epsilon t / k$ times.
 - w.h.p.,

$$Q_t(a) \approx q_*(a)$$

- Regret bound is

Let's analyze ϵ -Greedy!

- By Hoeffding's inequality, at every t
 - w.h.p. each arm is chosen at least $0.5 \epsilon t / k$ times.
 - w.h.p.,

$$Q_t(a) \approx \underline{q_*(a)} \pm C \sqrt{\frac{k}{\epsilon t}}$$

Error bar

of times
 $\sim \frac{\epsilon t}{k}$

- $\left[\begin{array}{l} \epsilon \\ \text{Regret bound} \end{array} \right]$ is

$$T \cdot \epsilon + \sum_{t=1}^T C \sqrt{\frac{k}{\epsilon t}}$$

$$C(1-\epsilon) \sqrt{\frac{k}{\epsilon}} \underbrace{\sum_{t=1}^T \sqrt{\frac{1}{t}}}$$

Let's analyze ϵ -Greedy!

- By Hoeffding's inequality, at every t
 - w.h.p. each arm is chosen at least $0.5 \epsilon t / k$ times.
 - w.h.p.,

$$Q_t(a) \approx q_*(a) \pm C \sqrt{k/\epsilon t}$$

- Regret bound is

$$\epsilon T$$

Let's analyze ϵ -Greedy!

- By Hoeffding's inequality, at every t
 - w.h.p. each arm is chosen at least $0.5 \epsilon t / k$ times.
 - w.h.p.,

$$Q_t(a) \approx q_*(a) \pm C \sqrt{k/\epsilon t}$$

- Regret bound is

$$\epsilon T + \sum_{t=1}^T C \sqrt{\frac{k}{\epsilon t}}$$

Convergent and divergent series

$$\leq 1 + \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \dots + \leq 2$$

$$1 + 1/2^2 + 1/3^2 + \dots + 1/T^2 \stackrel{T \rightarrow \infty}{=} \frac{\pi^2}{6}$$

Harmonic series

$$1 + 1/2 + 1/3 + \dots + 1/T =? \underline{O(\log T)}$$

On the side, check out Riemann's Zeta Function...

Convergent and divergent series

$$1 + 1/2^2 + 1/3^2 + \dots + 1/T^2 = ?$$

$$1 + 1/2 + 1/3 + \dots + 1/T = ?$$

$$1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}} + \dots + \frac{1}{\sqrt{T}} = ? \leq 2\sqrt{T}$$

$$\sum_{n=1}^T \frac{1}{\sqrt{n}} \leq 2\sqrt{T}$$

$$\int_0^T \frac{1}{\sqrt{x}} dx = [2\sqrt{x}]_0^T = 2\sqrt{T}$$

On the side, check out Riemann's Zeta Function... 11

Let's analyze ϵ -Greedy!

$$\epsilon T \approx N$$

- Regret bound

$$\epsilon T + \sum_{t=1}^T C \sqrt{\frac{k}{\epsilon t}} \leq \epsilon T + O\left(\sqrt{\frac{Tk}{\epsilon}}\right)$$

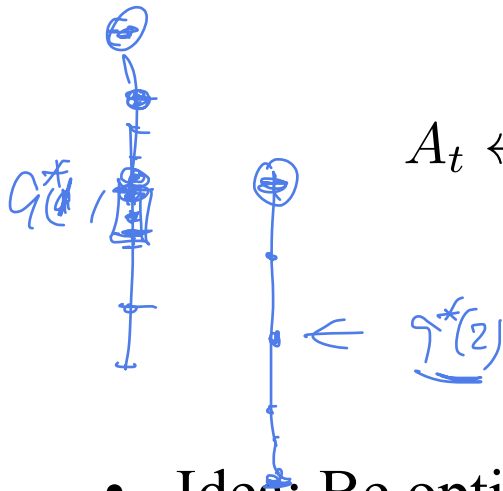
- Choose the optimal ϵ to minimize the bound?
- Work it out yourself that you get $T^{2/3}$ just like in Explore-First.

$$O\left(T^{2/3} k^{1/3}\right)$$

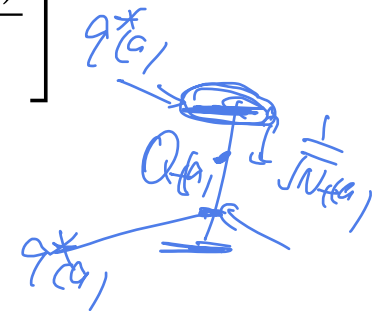
Upper Confidence Bound algorithm (UCB)

- At time t , choose the action

OPTIMAL



$$A_t \leftarrow \operatorname{argmax}_a \left[\underline{Q}_t(a) + c \sqrt{\frac{\log(1+t)}{N_t(a)}} \right]$$



- Idea: Be optimistic
 - Choose an option that maximizes the upper confidence bound.

- The proof is out of the scope of this course. For those who are interested, please look up. It's not difficult.

Upper Confidence Bound algorithm (UCB)

- At time t , choose the action

$$A_t \leftarrow \operatorname{argmax}_a \left[Q_t(a) + c \sqrt{\frac{\log(1+t)}{N_t(a)}} \right]$$

$$\sum_a N_t(a) = t-1$$

- Idea: Be optimistic
 - Choose an option that maximizes the upper confidence bound.

$$\mathbb{E}[\text{Regret}] = O(\sqrt{Tk})$$

- The proof is out of the scope of this course. For those who are interested, please look up. It's not difficult.

Example: Two-Armed Bandits

- $k=2$. Expected reward $r(a = 1) = 0.8$, $r(a = 2) = 0.5$

• Question: what is the Q-function for this problem?

- If episodic with horizon $T = 1$.
- If discounted with $0 \leq \gamma < 1$

$$Q(s,a) = \begin{cases} 0.8 + \gamma \frac{0.8}{1-\gamma} & a=1 \\ 0.5 + \gamma \frac{0.5}{1-\gamma} & a=2 \end{cases}$$

Handwritten derivation of the Bellman optimality equation:

$$Q^*(s,a) = r(s,a) + \gamma \max_{a'} Q^*(s',a')$$

The maximum value is shown to be 0.8.

• What is the optimal policy?

Handwritten derivation of the optimal policy:

$$Q^*(1) = 0.8 + \gamma Q^*(1)$$

$$Q^*(1) = \frac{0.8}{1-\gamma}$$

$$Q^*(2) = 0.5 + \gamma Q^*(2)$$

$$Q^*(2) = \frac{0.5}{1-\gamma}$$

Since $\frac{0.8}{1-\gamma} > \frac{0.5}{1-\gamma}$, the optimal policy is to choose action 1.

Example: UCB on a Two Armed Bandit

- A run of UCB algorithm with $c = 2$.
 1. Initialize $Q_1(a) = \infty, \forall a \in \{1,2\}$. UCB: $\bar{Q}_1(a) = \infty$.
 2. Pick action $A_1 = 1$. (break ties arbitrarily), receive a reward of 0.
 - a. Update $N_2(a) \leftarrow N_1(a) + 1(A_1 = a)$ $[N_2(1)=1, N_2(2)=0]$
 - b. Update $Q_2(1) = 0/1 = 0, Q_2(2) = \infty$
 - c. Update $\bar{Q}_2(1) \leftarrow Q_2(1) + 2\sqrt{\frac{\log 2}{1}} \approx 0 + 1.67, \bar{Q}_2(2) = \infty$
 3. Pick action $a = 2$ (because of what?), receive a reward of 1.
 - a. Update $N_3(2) \leftarrow N_2(2) + 1, Q_3(2) \leftarrow 1$
 - b. Update $\bar{Q}_3(2) \leftarrow Q_3(2) + 2\sqrt{\frac{\log 2}{1}} \approx 1 + 1.67$
 - c. Keep $N_3(1), Q_3(1), \bar{Q}_3(1)$ unchanged.
 4. Which action to pick at next? $\bar{Q}_2(1) = 1.67, \bar{Q}_3(2) = 2.67$
 5. ... pick $A_3 = 2$

A 10-armed bandits benchmark

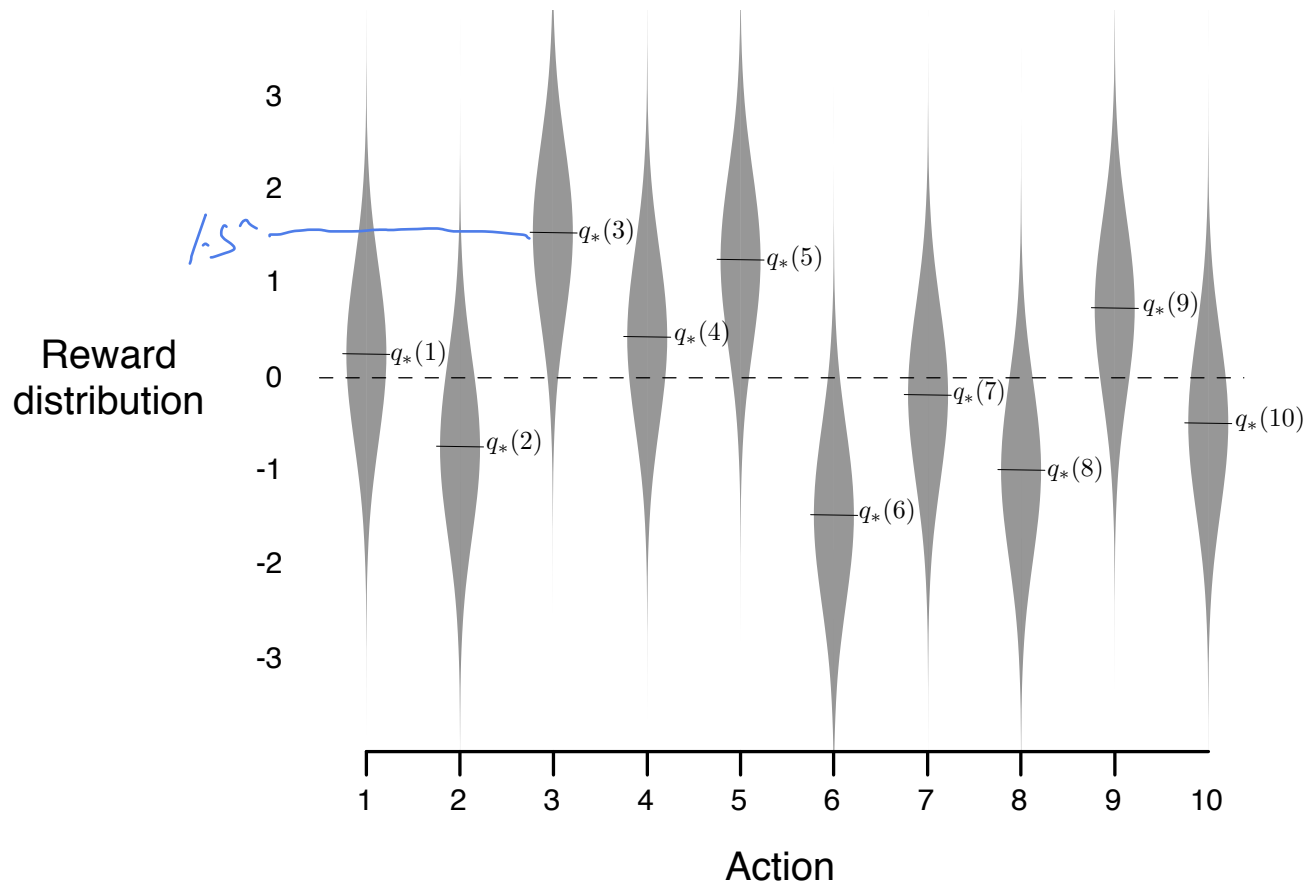
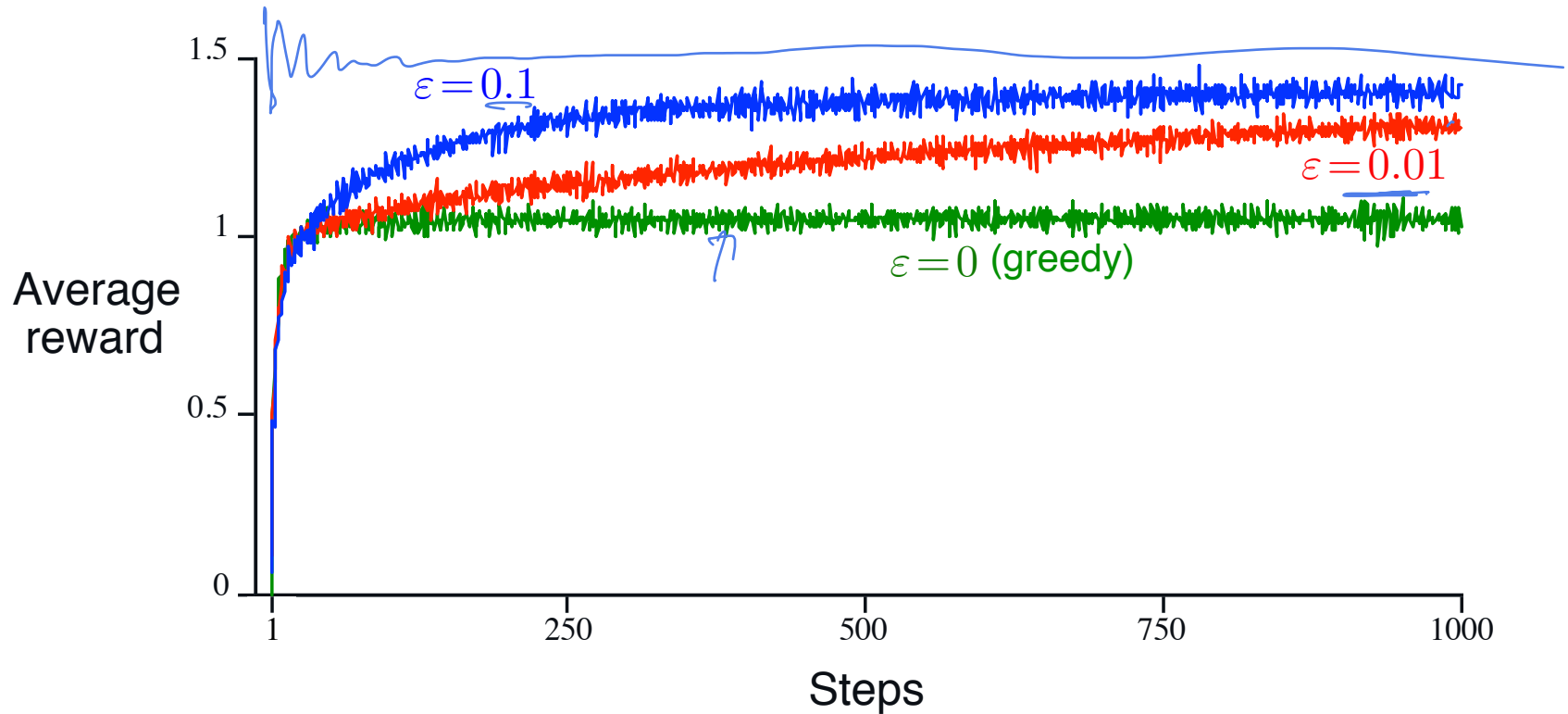
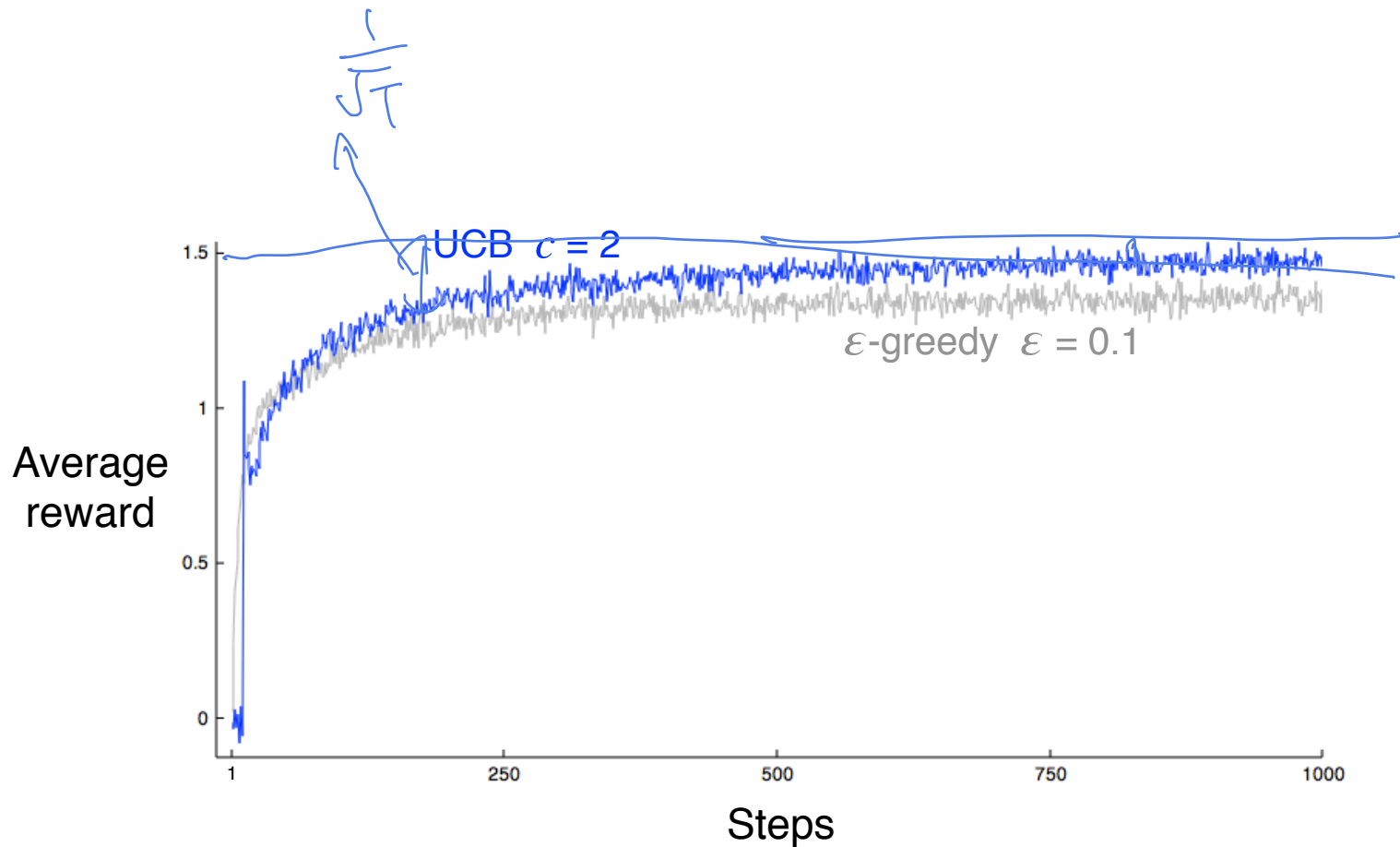


Figure 2.1: An example bandit problem from the 10-armed testbed. The true value $q_*(a)$ of each of the ten actions was selected according to a normal distribution with mean zero and unit variance, and then the actual rewards were selected according to a mean $q_*(a)$ unit variance normal distribution, as suggested by these gray distributions.

Comparing the different algorithms

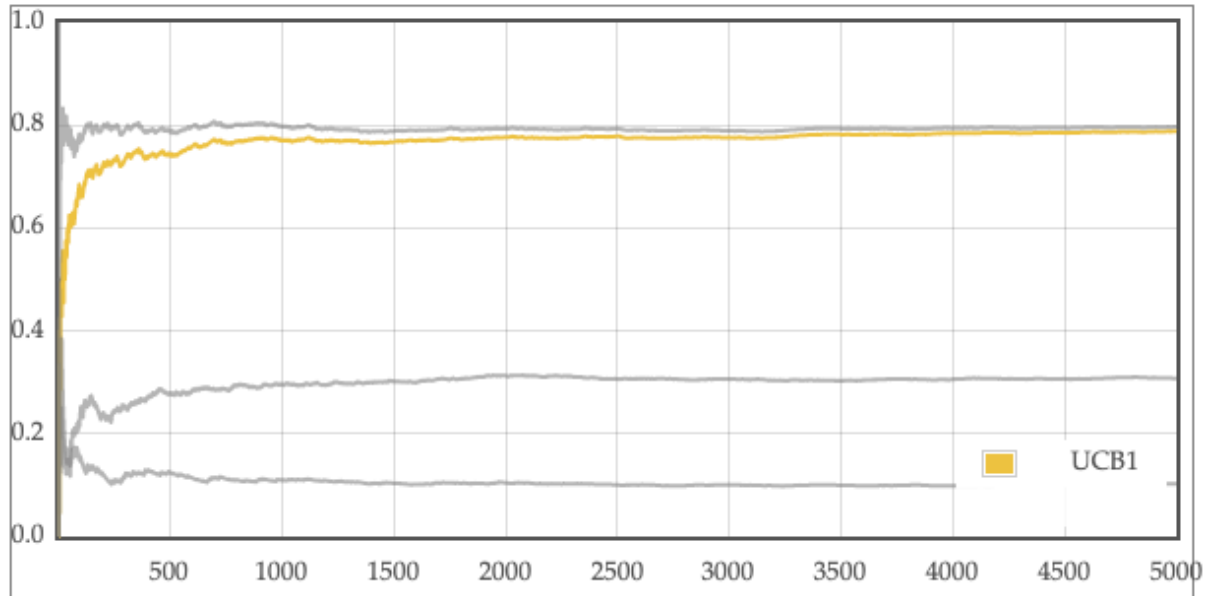


UCB vs. ϵ -Greedy



Live Demo of an Agent Solving Multi-Armed Bandits

Bandits		Agents	
1. Bernoulli	0.1	1. UCB1	Run game for 5000
2. Bernoulli	0.3	+ create	steps.
3. Bernoulli	0.8		<input type="button" value="Go!"/>
+ create			Average: <input checked="" type="checkbox"/>



Credit Mark Reid: <http://mark.reid.name/code/bandits/>

Variants of Bandits problems

- Online Learning from Expert Advice
 - Adversarial chooses the outcome
 - You observe outcome of other arms as well
 - Compare against the best arm in the hindsight
- Adversarial k-Armed Bandits
 - Same as above. But you observe only your arm.
- Nonstationary Bandits
 - Stochastic but the reward distribution changes over time.
 - Compare against the best arm for each time.
- Contextual bandits: you have a state in each time point.

$T \log k$

\sqrt{TK}

Variants of Bandits problems

- Online Learning from Expert Advice
 - Adversarial chooses the outcome
 - You observe outcome of other arms as well
 - Compare against the best arm in the hindsight
- Adversarial k-Armed Bandits
 - Same as above. But you observe only your arm.
- Nonstationary Bandits
 - Stochastic but the reward distribution changes over time.
 - Compare against the best arm for each time.
- Contextual bandits: you have a state in each time point.

Remark: In all these problems, there are algorithms with provably low-regret.

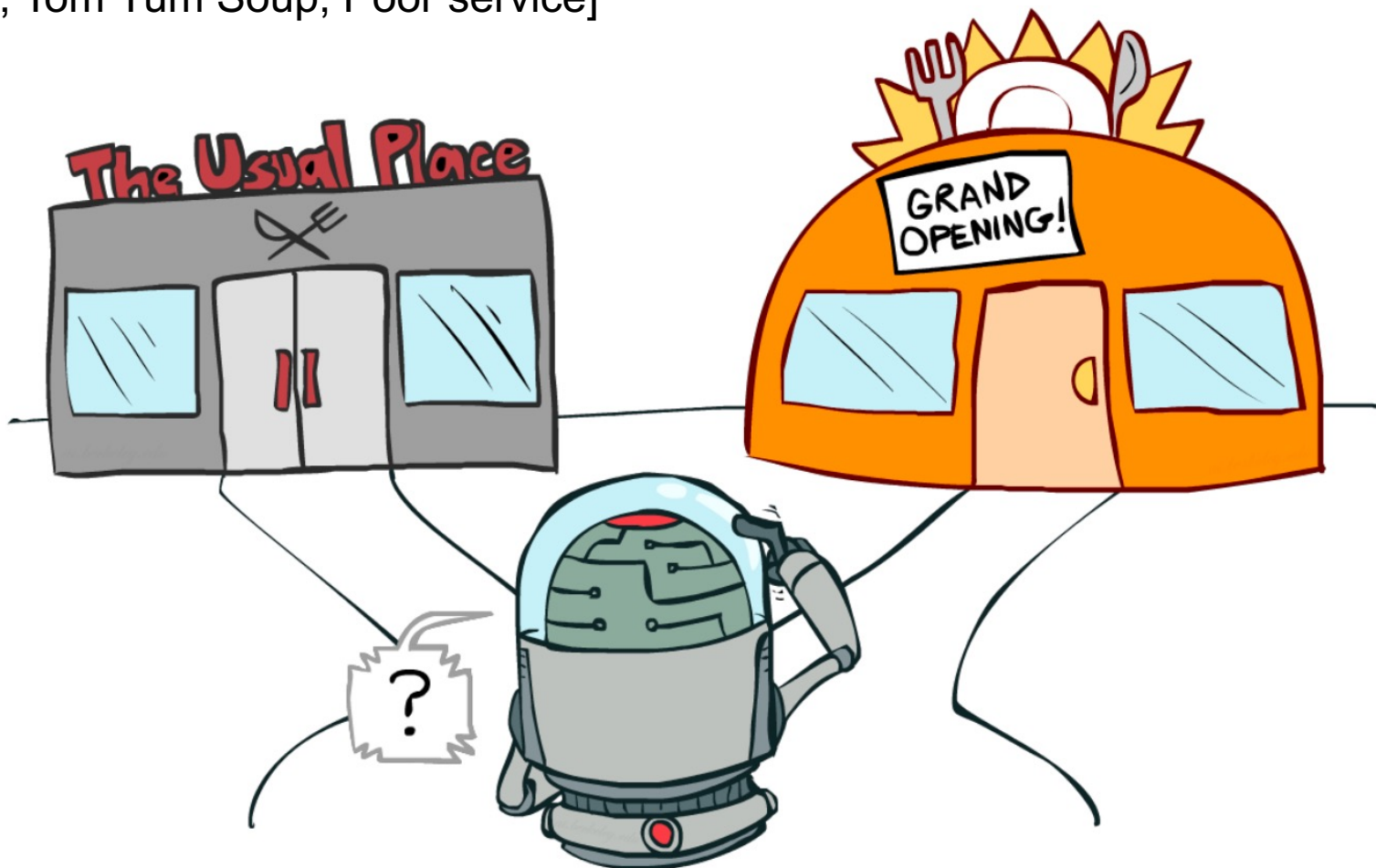
Do I have to try, if I have features?

Features:

[Noodles, Tom Yum Soup, Poor service]

Features:

[Burger, Fries, Onion Ring, Fried Chicken]



(Illustration from Dan Klein and Pieter Abbeel's course in UC Berkeley)

We know how to use with features, don't we?

- Classifier agent
 - Take features of a restaurant as input
 - Output a prediction of “will I like the food?”
- Train with supervised learning
 - Using the my previous visits to the restaurants
 - Using Yelp reviews

We know how to use with features, don't we?

- Classifier agent
 - Take features of a restaurant as input
 - Output a prediction of “will I like the food?”
- Train with supervised learning
 - Using the my previous visits to the restaurants
 - Using Yelp reviews

Why can't we just use that?

We know how to use with features, don't we?

- Classifier agent
 - Take features of a restaurant as input
 - Output a prediction of “will I like the food?”
- Train with supervised learning
 - Using the my previous visits to the restaurants
 - Using Yelp reviews

Why can't we just use that?

How to explore?

Contextual Bandits: Problem Setup

- For each round $t = 1, 2, 3, \dots, T$:
 - A context $x_t \sim$ unknown distribution i.i.d.
 - Agent picks an action $a_t = 1, 2, 3, \dots, K$
 - Reward $r_t \sim D(\cdot | x_t, a_t)$

- Agent's goals:

- Learn the best policy out of many policies
- Minimize the cumulative regret

A finite family of policies

Π

$$T \cdot \max_{\pi \in \Pi} \mathbb{E}_{\pi} [r_t(x_t, a_t)] - \mathbb{E}_{\text{Agent's policy}} \left[\sum_{t=1}^T r_t(x_t, a_t) \right]$$

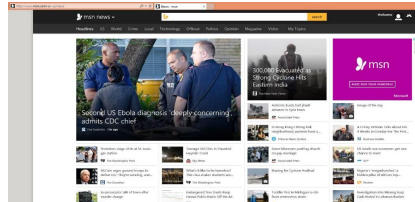
Reward from the best policy

Reward collected by the Agent

Applications of Contextual Bandits

Applications of Contextual Bandits

Personalized news?



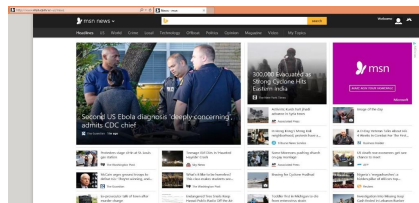
Repeatedly:

1. Observe features of user+articles
2. Choose a news article.
3. Observe click-or-not

Goal: Maximize fraction of clicks

Applications of Contextual Bandits

Personalized news?

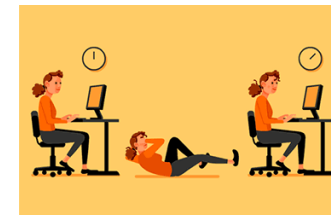


Repeatedly:

1. Observe features of user+articles
2. Choose a news article.
3. Observe click-or-not

Goal: Maximize fraction of clicks

Health advice?



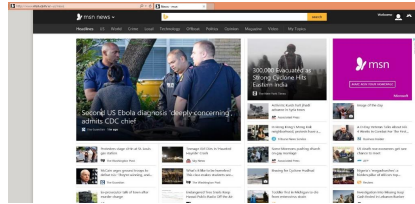
Repeatedly:

1. Observe features of user+advice
2. Choose an advice.
3. Observe steps walked

Goal: Healthy behaviors (e.g. step count)

Applications of Contextual Bandits

Personalized news?

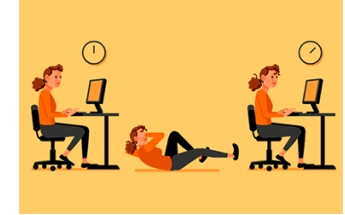


Repeatedly:

1. Observe features of user+articles
2. Choose a news article.
3. Observe click-or-not

Goal: Maximize fraction of clicks

Health advice?



Repeatedly:

1. Observe features of user+advice
2. Choose an advice.
3. Observe steps walked

Goal: Healthy behaviors (e.g. step count)

The Amazon logo, consisting of the word "amazon" in a bold, black, sans-serif font with a curved orange arrow underneath it.

Recommendations



buy or not buy



Exploration vs. Exploitation in Contextual Bandits.

Exploration vs. Exploitation in Contextual Bandits.

- Challenging because:
 - **Infinite state space**, never see the same context again.
 - **Exponentially large** policy space

Exploration vs. Exploitation in Contextual Bandits.

- Challenging because:
 - **Infinite state space**, never see the same context again.
 - **Exponentially large** policy space
- Ideas:
 - ExploreFirst, ϵ -Greedy $O(T^{2/3})$


Exploration vs. Exploitation in Contextual Bandits.

- Challenging because:
 - **Infinite state space**, never see the same context again.
 - **Exponentially large** policy space
- Ideas:
 - ExploreFirst, ϵ -Greedy $O(T^{2/3})$
 - UCB? But how do we construct Confidence Interval for an exponentially large set of policies?

Exploration vs. Exploitation in Contextual Bandits.

- Challenging because:
 - **Infinite state space**, never see the same context again.
 - **Exponentially large** policy space
- Ideas:
 - ExploreFirst, ϵ -Greedy $O(T^{2/3})$
 - UCB? But how do we construct Confidence Interval for an exponentially large set of policies?
- Optimal regret:

$$O(\sqrt{KT \log |\Pi|})$$


 d

Remainder of the lecture today

- Reinforcement learning for MDPs
 - Model-based vs model-free algorithms
 - Online policy iterations
 - Temporal difference learning
- Readings:
 - AIMA Ch. 21.1-21.3 (Ch 22.1- 22.3 in 4th Edition)
 - Sutton and Barto: Ch 4-6
 - Maybe: Sutton and Barto: Ch 6, Ch 13

Let us tackle different aspects of the RL problem one at a time

- Markov Decision Processes:
 - Dynamics are given no need to learn
- Bandits: Explore-Exploit in simple settings
 - RL without dynamics
- **Full Reinforcement Learning**
 - Learning MDPs

Recap: Tabular MDP

- **Discrete** State, **Discrete** Action, Reward and Observation

$$S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad \text{---} \quad \cancel{O_t \in \mathcal{O}}$$

- Policy:

– When the state is observable: $\pi : \mathcal{S} \rightarrow \mathcal{A}$

~~– Or when the state is not observable~~

$$\text{---} \quad \cancel{\pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A}}$$

- Learn the best policy that maximizes the expected reward

– Finite horizon (episodic) RL: $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[\sum_{t=1}^T R_t \right]$ **T: horizon**

– Infinite horizon RL: $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \right]$

γ : discount factor

Recap: Policy Iterations and Value Iterations

- What are these algorithms for?
 - Algorithms of computing the V^* and Q^* functions from MDP parameters

- Policy Iterations

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

- Value iterations

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k(s')]$$

- How do we make sense of them?
 - Recursively applying the Bellman equations until convergence.

Recap: Policy Iterations and Value Iterations

- What are these algorithms for?
 - Algorithms of computing the V^* and Q^* functions from MDP parameters

- Policy Iterations

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

- Value iterations

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k(s')]$$

- How do we make sense of them?
 - Recursively applying the Bellman equations until convergence.

*These methods are called “Dynamic Programming” approaches in Chap 4 of Sutton and Barto.

Revisit the dynamic programming approach

- Policy Evaluation

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k^{\pi}(s')]$$

- Policy improvement

$$\begin{aligned} \pi'(s) &= \arg \max_a Q^{\pi}(s, a) \\ &= \arg \max_a \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k^{\pi}(s')] \end{aligned}$$

- Value iterations

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k(s')]$$

Revisit the dynamic programming approach

- Policy Evaluation

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \cancel{P(s'|s, a)} [\cancel{r(s, a, s')} + \gamma V_k^{\pi}(s')]$$

- Policy improvement

$$\begin{aligned} \pi'(s) &= \arg \max_a Q^{\pi}(s, a) \\ &= \arg \max_a \sum_{s'} \cancel{P(s'|s, a)} [\cancel{r(s, a, s')} + \gamma V_k^{\pi}(s')] \end{aligned}$$

- Value iterations

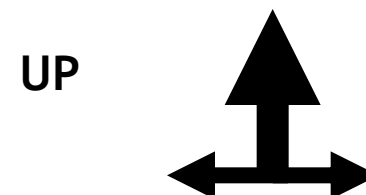
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \cancel{P(s'|s, a)} [\cancel{r(s, a, s')} + \gamma V_k(s')]$$

***We do not have the MDP parameters in RL!**

Example: Frozen Lake

			+1
			-1
START			

actions: UP, DOWN, LEFT, RIGHT



80%

move UP

10%

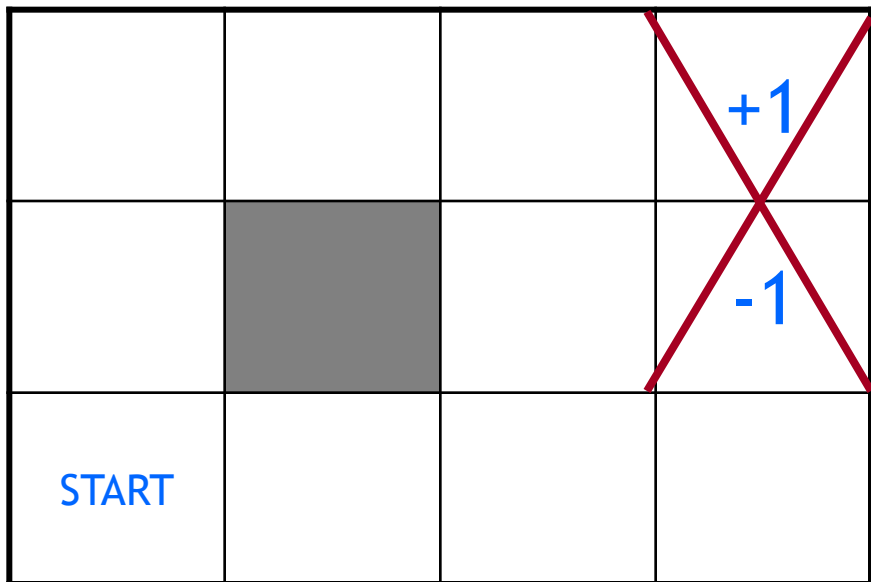
move LEFT

10%

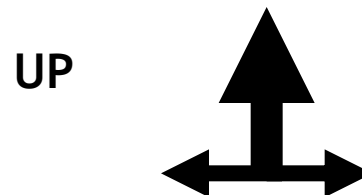
move RIGHT

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- what's the strategy to achieve max reward?

Example: Frozen Lake



actions: UP, DOWN, LEFT, RIGHT



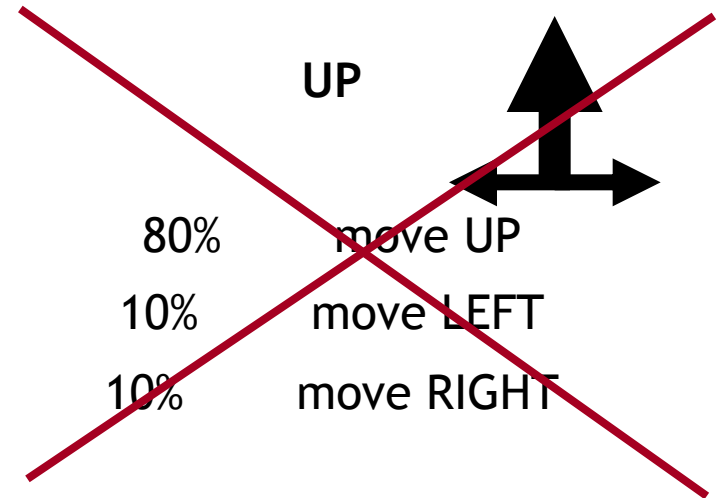
80% move UP
10% move LEFT
10% move RIGHT

- ~~reward +1 at [4,3], -1 at [4,2]~~
- ~~reward -0.04 for each step~~
- what's the strategy to achieve max reward?

Example: Frozen Lake

			+1
			-1
START			

actions: UP, DOWN, LEFT, RIGHT

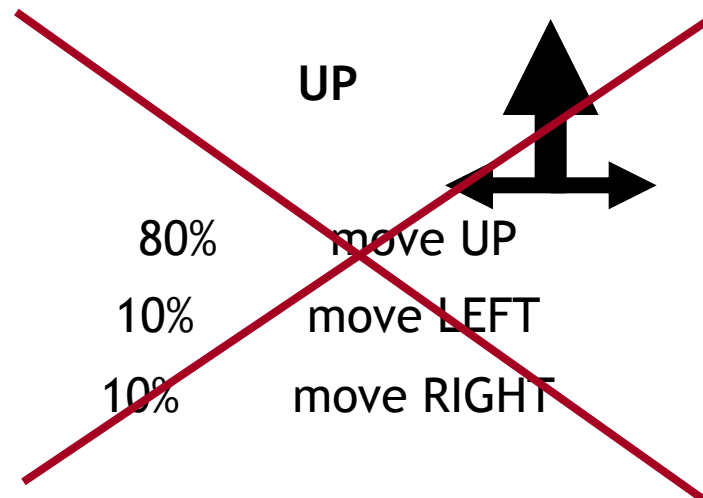


- ~~• reward +1 at [4,3], -1 at [4,2]~~
- ~~• reward -0.04 for each step~~
- what's the strategy to achieve max reward?

Example: Frozen Lake

			+1
			-1
START			

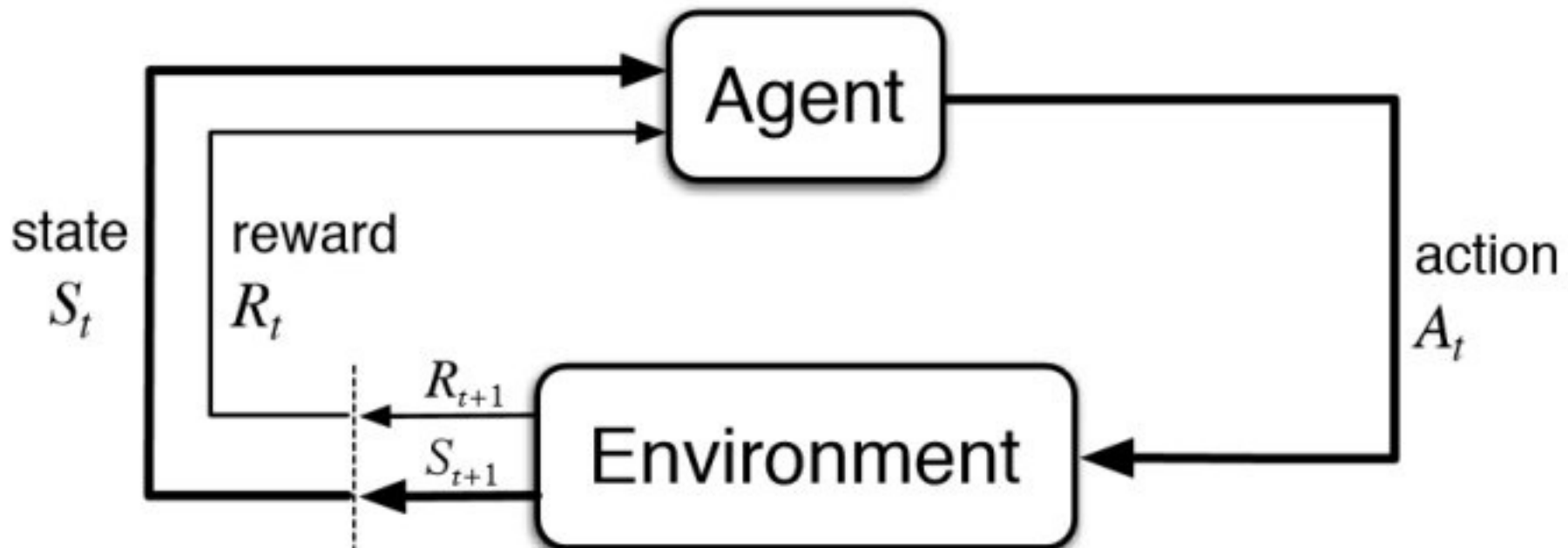
Action 1, Action 2, Action 3, Action 4
actions. ~~UP, DOWN, LEFT, RIGHT~~



- ~~reward +1 at [4,3], -1 at [4,2]~~
- ~~reward -0.04 for each step~~
- what's the strategy to achieve max reward?

Instead, reinforcement learning agents have “online” access to an environment

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents can “act” and “experiment”, rather than only doing offline planning.



Idea 1: Model-based Reinforcement Learning

- Model-based idea
 - Let's approximate the model based on experiences
 - Then solve for the values as if the learned model were correct
- Step 1: Get data by running the agent to explore
 - Many data points of the form:
 $\{(s_1, a_1, s_2, r_1), \dots, (s_N, a_N, s_{N+1}, r_N)\}$
- Step 2: Estimate the model parameters
 - $\hat{P}(s'|s, a)$ --- again this is a CPT we need to observe the transition many times for each s, a
 - $\hat{r}(s', a, s)$ --- this is an estimate of the empirical rewards.

Then we can plug in these estimates and then use dynamic programming for policy evaluation / improvements.

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$\pi' \leftarrow \arg \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k(s')]$$

Then we can plug in these estimates and then use dynamic programming for policy evaluation / improvements.

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$\pi' \leftarrow \arg \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k(s')]$$

Then we can plug in these estimates and then use dynamic programming for policy evaluation / improvements.

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$\pi' \leftarrow \arg \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k(s')]$$

* Note the “**hat**”. Usually it indicates empirical estimates.

Then we can plug in these estimates and then use dynamic programming for policy evaluation / improvements.

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

$$\pi' \leftarrow \arg \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k^{\pi}(s')]$$

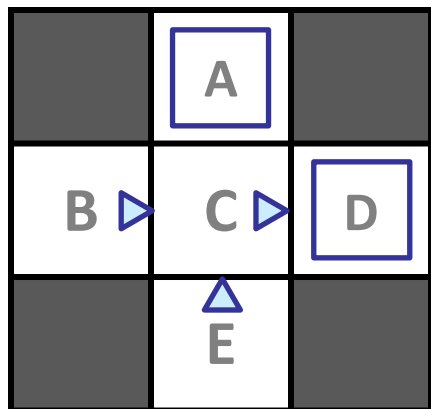
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} \hat{P}(s'|s, a) [\hat{r}(s, a, s') + \gamma V_k(s')]$$

* Note the “**hat**”. Usually it indicates empirical estimates.

* These iterations will produce \hat{V}^* and \hat{Q}^* functions, and then $\hat{\pi}^*$

Example: Model-Based RL (2 min exercise)

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

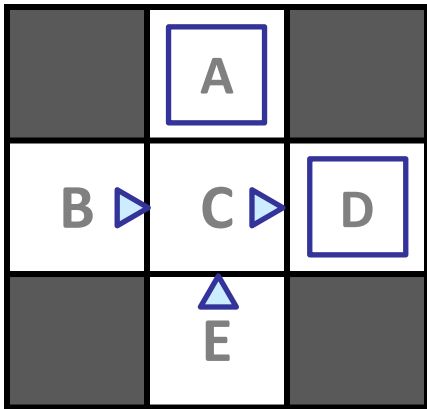
Learned Model

$$\hat{P}(s'|s, a)$$

$$\hat{r}(s, a, s')$$

Example: Model-Based RL (2 min exercise)

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

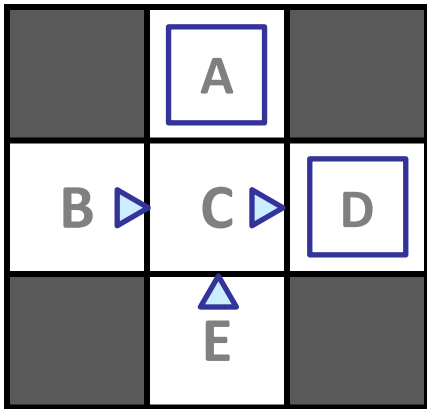
Learned Model

$$\hat{P}(s'|s, a)$$

$$\hat{r}(s, a, s')$$

Example: Model-Based RL (2 min exercise)

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
 C, east, D, -1
 D, exit, x, +10

Episode 2

B, east, C, -1
 C, east, D, -1
 D, exit, x, +10

Episode 3

E, north, C, -1
 C, east, D, -1
 D, exit, x, +10

Episode 4

E, north, C, -1
 C, east, A, -1
 A, exit, x, -10

Learned Model

$$\hat{P}(s'|s, a)$$

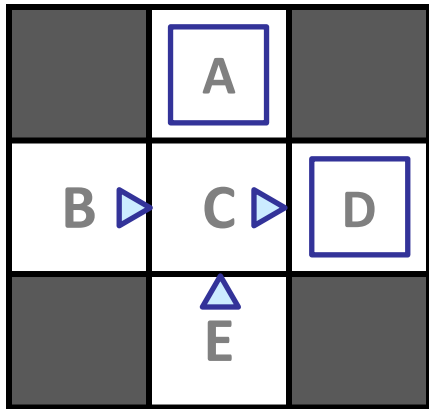
T(B, east, C) =
 T(C, east, D) =
 T(C, east, A) =
 ...

$$\hat{r}(s, a, s')$$

R(B, east, C) =
 R(C, east, D) =
 R(D, exit, x) =
 ...

Example: Model-Based RL (2 min exercise)

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

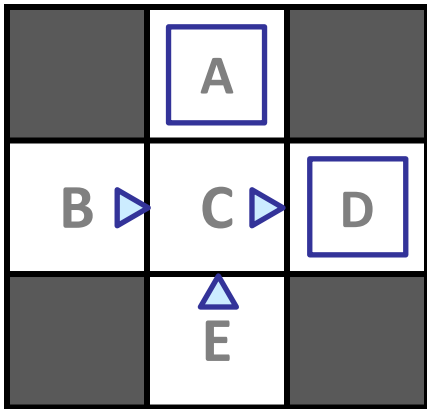
Learned Model

$$\hat{P}(s'|s, a)$$

$$\hat{r}(s, a, s')$$

Example: Model-Based RL (2 min exercise)

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

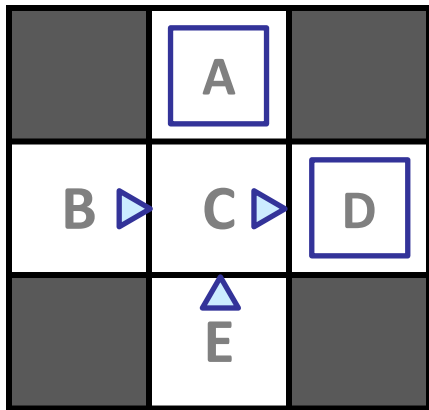
Learned Model

$$\hat{P}(s'|s, a)$$

$$\hat{r}(s, a, s')$$

Example: Model-Based RL (2 min exercise)

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$$\hat{P}(s'|s, a)$$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
...

$$\hat{r}(s, a, s')$$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
...

This is simply the “Exploration-First” strategy! But there are complications.

This is simply the “Exploration-First” strategy! But there are complications.

- In bandits problems
 - Uniformly sample the actions for N rounds.
 - Guarantees that each choice is explored $O(N/k)$ times.

This is simply the “Exploration-First” strategy! But there are complications.

- In bandits problems
 - Uniformly sample the actions for N rounds.
 - Guarantees that each choice is explored $O(N/k)$ times.
- For MDPs
 - Often we need to take a carefully chosen sequence of actions to reach a state
 - The chance of randomly running into a state can be **exponentially small**.

This is simply the “Exploration-First” strategy! But there are complications.

- In bandits problems
 - Uniformly sample the actions for N rounds.
 - Guarantees that each choice is explored $O(N/k)$ times.
- For MDPs
 - Often we need to take a carefully chosen sequence of actions to reach a state
 - The chance of randomly running into a state can be **exponentially small**.
 - **Question: What is an example of this?**

This is simply the “Exploration-First” strategy! But there are complications.

- In bandits problems
 - Uniformly sample the actions for N rounds.
 - Guarantees that each choice is explored $O(N/k)$ times.
- For MDPs
 - Often we need to take a carefully chosen sequence of actions to reach a state
 - The chance of randomly running into a state can be **exponentially small**.
 - **Question: What is an example of this?**

*Need to somehow update the “exploration policy” on the fly!

More caveats

-

?

More caveats

- The fitted model is just an approximation of the environment.

-

?

More caveats

- The fitted model is just an approximation of the environment.
- How does the error in the fitted MDP translate into the error in the estimated value functions V^* and Q^* ?

-

?

More caveats

- Q^* ?
- The fitted model is just an approximation of the environment.
- How does the error in the fitted MDP translate into the error in the estimated value functions V^* and Q^* ?
- How does the error in the estimated Q^* function affect the suboptimality of the policy that maximizes Q^* ?
 Q^* ?

More caveats

- Q^* ?
- The fitted model is just an approximation of the environment.
- How does the error in the fitted MDP translate into the error in the estimated value functions V^* and Q^* ?
- How does the error in the estimated Q^* function affect the suboptimality of the policy that maximizes Q^* ?
- Answered by “Simulation Lemma” (Kearns and Singh, 2002)
 - Resurgence of research on this more recently: Yin and W. (2020), Yin, Bai and W. (2020)

Even more caveats

- How many free parameters are there to represent an MDP?

Even more caveats

- How many free parameters are there to represent an MDP?
 - Ans: $O(S^2A)$

Even more caveats

- How many free parameters are there to represent an MDP?
 - Ans: $O(S^2A)$
- S is often large
 - 9-puzzle, Tic-Tac-Toe: $9! = 362,800$, $S^2 = 1.3 \cdot 10^{11}$

Even more caveats

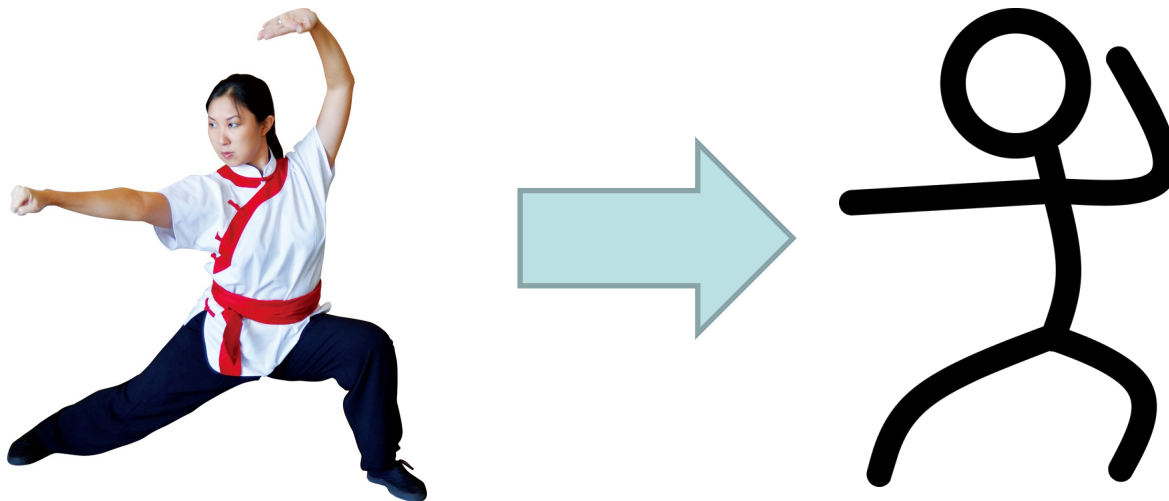
- How many free parameters are there to represent an MDP?
 - Ans: $O(S^2A)$
- S is often large
 - 9-puzzle, Tic-Tac-Toe: $9! = 362,800$, $S^2 = 1.3 \cdot 10^{11}$
 - PACMAN with 20 by 20 grid. $S = O(2^{400})$, $S^2 = O(2^{800})$

Even more caveats

- How many free parameters are there to represent an MDP?
 - Ans: $O(S^2A)$
- S is often large
 - 9-puzzle, Tic-Tac-Toe: $9! = 362,800$, $S^2 = 1.3 \cdot 10^{11}$
 - PACMAN with 20 by 20 grid. $S = O(2^{400})$, $S^2 = O(2^{800})$
- In practice, we often have to use an approximate model.

Even more caveats

- How many free parameters are there to represent an MDP?
 - Ans: $O(S^2A)$
- S is often large
 - 9-puzzle, Tic-Tac-Toe: $9! = 362,880$, $S^2 = 1.3 \cdot 10^{11}$
 - PACMAN with 20 by 20 grid. $S = O(2^{400})$, $S^2 = O(2^{800})$
- In practice, we often have to use an approximate model.



Idea 2: **Model-free** Reinforcement Learning

- Do we need the model? Can we learn the Q function directly?

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

Idea 2: Model-free Reinforcement Learning

- Do we need the model? Can we learn the Q function directly?
 - **How many free parameters are there to represent the Q-function?**

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

Idea 2: Model-free Reinforcement Learning

- Do we need the model? Can we learn the Q function directly?
 - **How many free parameters are there to represent the Q-function?**
 - **Ans: $SA \ll O(S^2A)$**

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

Idea 2: Model-free Reinforcement Learning

- Do we need the model? Can we learn the Q function directly?
 - **How many free parameters are there to represent the Q-function?**
 - **Ans: $SA \ll O(S^2A)$**
- Recall: Policy iterations

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

Idea 2: Model-free Reinforcement Learning

- Do we need the model? Can we learn the Q function directly?
 - **How many free parameters are there to represent the Q-function?**
 - **Ans: $SA \ll O(S^2A)$**
- Recall: Policy iterations

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

- **Maybe we can do policy evaluation without estimating the model?**

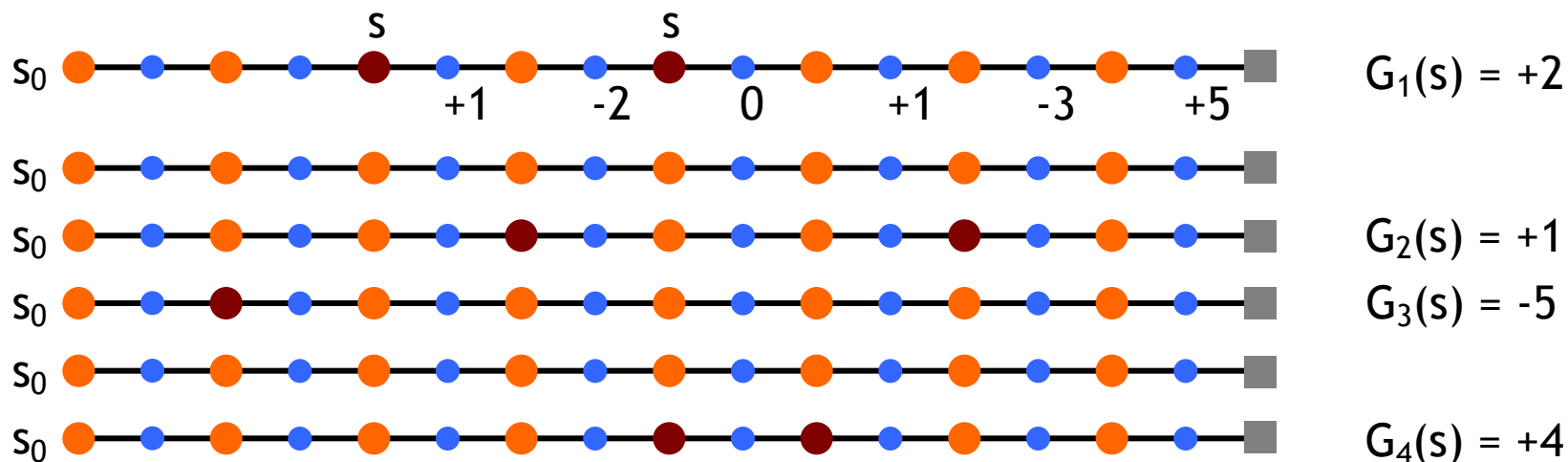
Monte Carlo Policy Evaluation (Prediction)

- want to estimate $V^\pi(s)$
 - = expected return starting from s and following π
 - estimate as average of observed returns in state s
- We can execute the policy π
- first-visit MC
 - average returns following the first visit to state s



Monte Carlo Policy Evaluation (Prediction)

- want to estimate $V^\pi(s)$
 - = expected return starting from s and following π
 - estimate as average of observed returns in state s
- We can execute the policy π
- first-visit MC
 - average returns following the first visit to state s

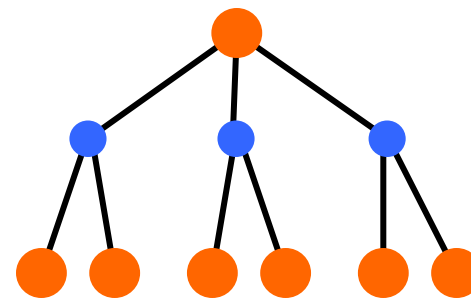


$$V^\pi(s) \approx (2 + 1 - 5 + 4) / 4 = 0.5$$

Monte Carlo Policy Optimization (Control)

- V^π not enough for policy improvement
 - need exact model of environment

- estimate $Q^\pi(s,a)$
 $\pi'(s) = \arg \max_a Q^\pi(s, a)$



- MC control

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} Q^*$$

- update after each episode

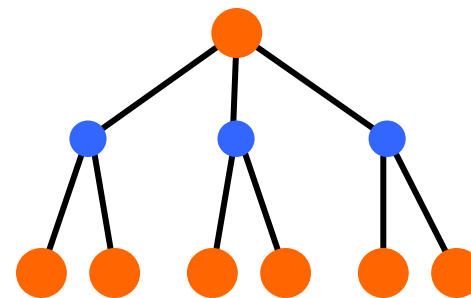
- Two problems

- greedy policy won't explore all actions
- Requires many independent episodes for the estimated value function to be accurate.

Monte Carlo Policy Optimization (Control)

- V^π not enough for policy improvement
 - need exact model of environment

- estimate $Q^\pi(s,a)$
 $\pi'(s) = \arg \max_a Q^\pi(s, a)$



- MC control

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} Q^*$$

- update after each episode

- Two problems

- greedy policy won't explore all actions **eps-greedy!**
- Requires many independent episodes for the estimated value function to be accurate.

Improved Monte-Carlo Q-function estimate using Bellman equations

- Recall:

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

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

Improved Monte-Carlo Q-function estimate using Bellman equations

- Recall:

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

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

- We can use the empirical (Monte Carlo) estimate.

$$\widehat{Q}^\pi(s, a) = \widehat{r}^\pi(s, a) + \gamma \widehat{\mathbb{E}}_{s' \sim P(s'|s, a)} [\widehat{V}^\pi(s')]$$

Improved Monte-Carlo Q-function estimate using Bellman equations

- Recall:

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

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

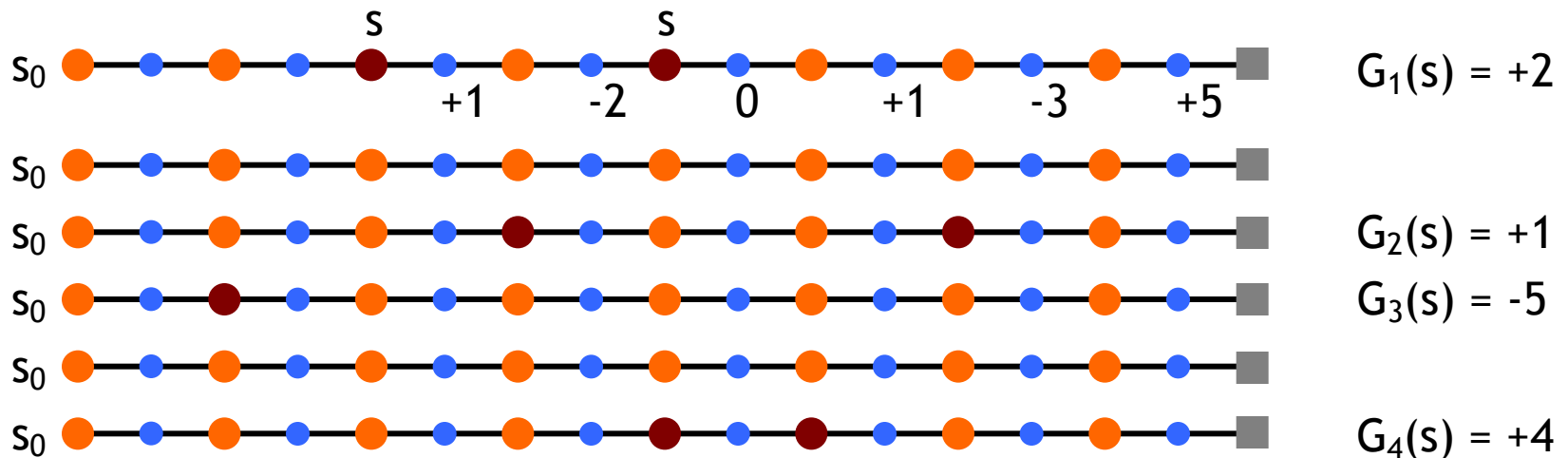
- We can use the empirical (Monte Carlo) estimate.

$$\widehat{Q}^\pi(s, a) = \widehat{r}^\pi(s, a) + \gamma \widehat{\mathbb{E}}_{s' \sim P(s'|s, a)} [\widehat{V}^\pi(s')]$$

*No need to estimate $P(s' | s, a)$ or $r(s, a, s')$ as intermediate steps.

*Require only $O(SA)$ space, rather than $O(S^2A)$

Online averaging representation of MC



$$V^\pi(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$$

- Alternative, *online averaging* update

$$V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)], \quad \text{where } \alpha = 1/N_{S_t}$$

So far, in RL algorithms

- Model-based approaches
 - Estimate the MDP parameters.
 - Then use policy-iterations, value iterations.
- Monte Carlo methods:
 - estimating the rewards by empirical averages

Next lecture

- Wrap up RL lectures
 - Temporal difference methods and bootstrapping
 - Linear function approximation and Q-learning