CS540 Introduction to Artificial Intelligence
Lecture 13

Young Wu
Based on lecture slides by Jerry Zhu and Yingyu Liang

July 2, 2019

Secretary Problem

Motivation

- Interview 10 people, random order, either give an offer or reject immediately after each interview. The goal is to give an offer to the best candidate. Optimal strategy: interview first n people, give an offer to the first candidate who is better than all previous ones. What is n?
- A: 1, B: 2, C: 3, D: 4, E: 5

always reject n

find best so far after n

$n = \frac{10}{e} \approx 3$

$e \approx 2.7$

Prob{hiring best} = $\frac{1}{e}$

Secretary Problem Solution

Motivation

Schedule

Admin

posted

- Thursday, July 4: Post sample midterm and formula sheet.
- Tuesday, July 9: Tan review session: review + sample midterm.
- Thursday, July 11: Midterm Version A.
- Tuesday, July 16: Midterm Version B.

Midterm

Admin

- 2 hour midterm, $5:30$ to $7:30+\varepsilon, \varepsilon > 0$.
- Which midterm will you attend?
- A: Regular: Thursday, July 11.
- B: Alternative: Tuesday, July 16.
- C: Cannot make both.

Reinforcement Learning

Motivation

- Reinforcement learning is about learning from the outcome of actions.

1. Sense world.
2. Reason.
3. Choose an action to perform.
4. Get feedback.
5. Learn.

Applications

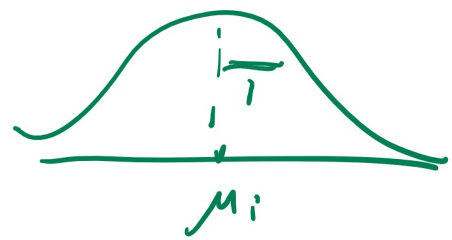
Motivation

- Actions can be performed in the physical world or artificial ones.
- Board games.
- Robotic control.
- Autonomous helicopter performance.
- Economics models.

Bandits

Motivation

- There are K arms, pulling each arm i results in reward r_i.
- The reward r_i is random and a follows Gaussian distribution with mean reward μ_i and variance $\sigma^2 = 1$.
- Suppose $\mu_1 \geqslant \mu_2 \geqslant \mu_3 \geqslant \ldots \geqslant \mu_K$.



Exploration then Exploitation Algorithm

Motivation

$\mu_1 > \mu_2 >> \mu_3$

1. Pull each arm t times to estimate the mean reward.

$$\hat{\mu}_{i,t} = \frac{1}{n} \sum_{t'=1}^{t} r_{i,t'}$$

$t \quad t \quad t$

$r_{i,t'}$ is the random reward from arm i and t' -th pull.

$\mu_i \approx \hat{\mu}_{i,t}$

2. Pull the arm $i^\star$ with the highest estimated mean reward.

after tK periods

$$\underline{i^\star = \arg \max_{i=1,2,\ldots,K} \hat{\mu}_{i,t}}$$

Upper Confidence Bound Algorithm

Motivation

1. Pull the arm $i^\star$ with the highest upper confidence bound.

$$i^\star = \arg\max_{i=1,2,\dots,K} \begin{cases} \text{UCB} = \hat{\mu}_{i,t} + \sqrt{\dfrac{2\log\left(\frac{1}{\delta}\right)}{t}} & t > 0 \\ \infty & t = 0 \end{cases}$$

always pulling best arm

with high confidence

δ is the confidence level parameter.

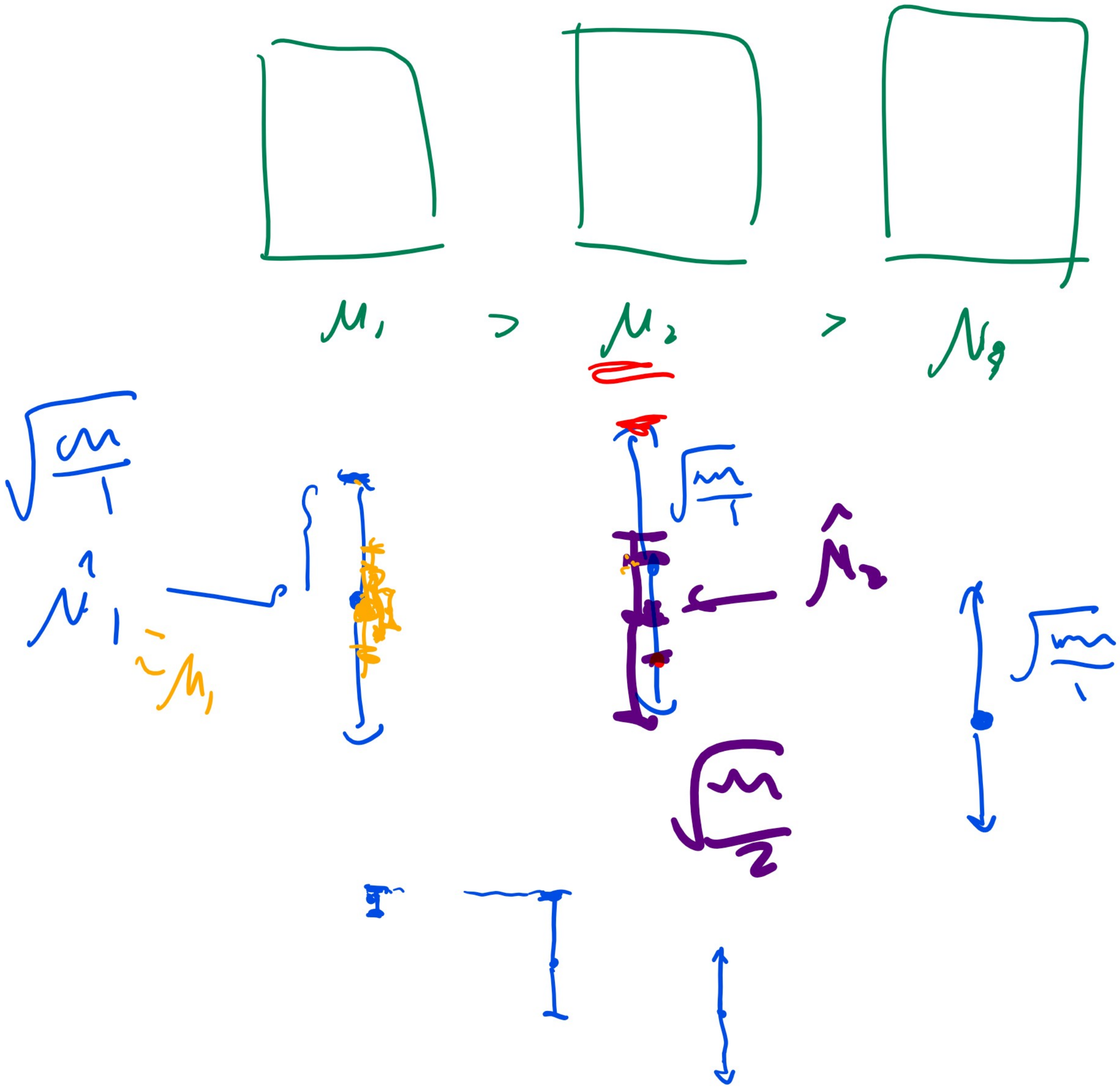
0.99

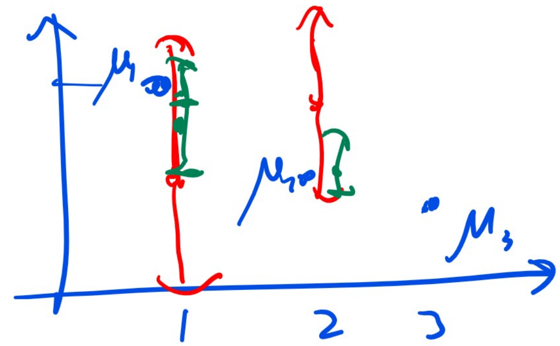
0.01

$$\mathbb{P}\left\{ \mu_i \leqslant \underbrace{\hat{\mu}_{i,t} + \sqrt{\dfrac{2\log\left(\frac{1}{\delta}\right)}{t}}} \right\} \leqslant 1 - \delta$$

UCB Algorithm Diagram

Motivation





pull everything once

K

Bandit Applications

Motivation

- Managing research projects.
- Treatment for patients.
- Search engine ranking.
- Wireless adaptive routing.
- Financial portfolio design.

Q Learning

Description

- Select an action.
- Receive reward.
- Observe new state.
- Update (learn) the value of the state-action pair.

State and Actions

Definition

- The set of possible states is $s_t \in S$.
- The set of possible actions is $a_t \in A$.
- The set of possible rewards is $r_t \in R$.
- At each time t :

1. Observe state s_t.
2. Chooses action a_t.
3. Receives reward r_t.
4. Changes to state s_{t+1}.

Markov Decision Process

Definition

- Markov property on states and actions is assumed.

$$\mathbb{P}\{s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...\} = \mathbb{P}\{s_{t+1}|s_t, a_t\}$$
$$\mathbb{P}\{r_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...\} = \mathbb{P}\{r_{t+1}|s_t, a_t\}$$

- The goal is to learn a policy function $\pi : S \to A$ for choosing actions that maximize the total expected discounted reward.

$$\mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...\right], \gamma \in [0, 1]$$

random

Expected Reward

Definition

- The expected reward at a given time t is the average reward weighted by probabilities.

$$\mathbb{E}\left[r_t\right] = \sum_{r_t \in R} r_t \mathbb{P}\left\{r_t | s_{t-1}, a_{t-1}\right\}$$

Discounted Reward

Definition

- The discounted reward at time 0 is the sum of reward weighted given the time preference, usually described by a constant discount factor.

$$\text{PV}\ (r_t) = \gamma^t r_t, \gamma \in [0, 1]$$

$$\text{PV}\ (r_1, r_2, \ldots) = \sum_{t=0}^{\infty} \gamma^t r_t$$

$\approx \frac{1}{1+\text{Interest}}$

$1 + \gamma + \gamma^2 + \gamma^3 + \ldots = \frac{1}{1-\gamma}$

- γ is the value of 1 unit of reward at time 1 perceived at time 0. If $\gamma = 1$, the sum over an infinite time period is usually infinity, therefore $\gamma < 1$ is usually used.

Value Function

Definition

- The value function is the expected discounted reward given a policy function π, assuming the action sequence is chosen according to π stating with state s.

$$\boxed{V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^{t} \mathbb{E}[r_t]}$$

- The optimal policy $\pi^{\star}$ is the one that maximizes the value function.

$$\pi^{\star} = \arg\max_{\pi} V^{\pi}(s) \text{ for all } s \in S$$

$$V^{\star}(s) = V^{\pi^{\star}}(s)$$

Goal Learning Example, Part I

Definition

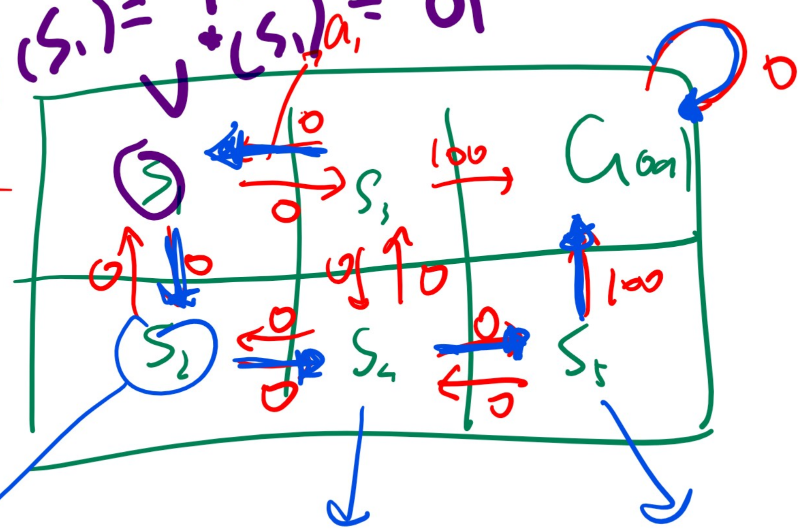
$\gamma = 0.9$

$V(s_1) = 72?$

$V^*(s_1, a_1) = 81$

state = where you are

policy your behavior(state)



$r(s_1, \downarrow) = 0$

$= \begin{cases} +1 & \text{w.p. } \frac{1}{2} \\ -1 & \text{w.p. } \frac{1}{2} \end{cases}$

$V(S_5) = 100$

$V(S_4) = 0 + 0.9 \cdot 100 = 90$

$V(S_2) = r(s_2, \rightarrow) + 0.9\, r(S_4, \rightarrow) + 0.9^2 r(S_5, \uparrow) + 0.9^3 \cdot r(G, ↺) + 0.9^4 r(G, ↺) + 0.9^5 r(G, ↺)$

81

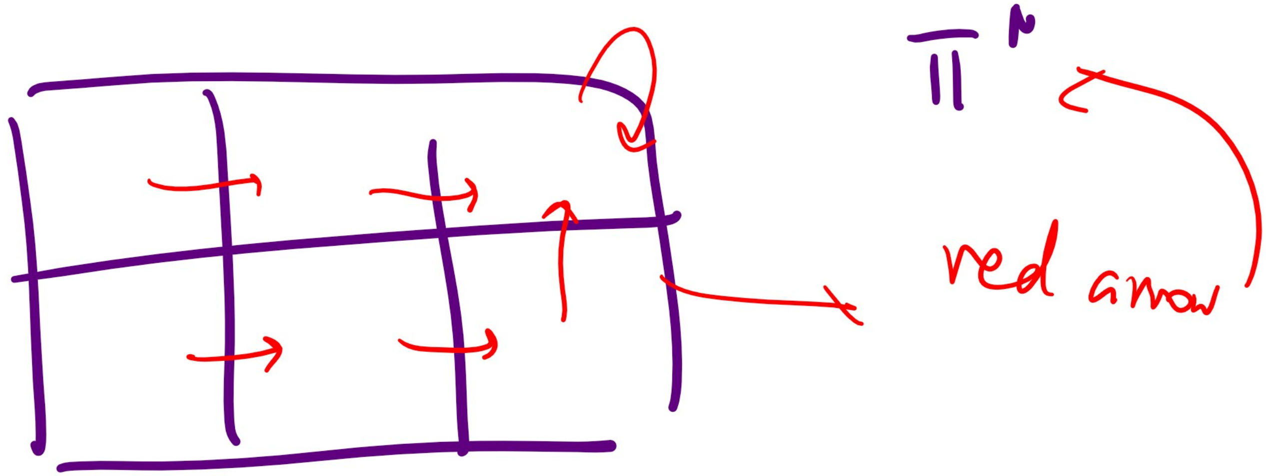
$0 + 0.9 \cdot 0 + 0.9^2 \cdot 100 + 0 + 0 \ldots$

$= 81$

0

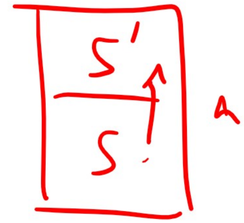
Goal Learning Example, Part II

Definition



Optimal Policy Given Value Function

Definition



$V(s) = r(s \to s', a) + V^*(s')$

- Given $V^\star(s)$, $r(s,a)$, $\mathbb{P}(s'|s,a)$, $\pi^\star$ can be computed directly.

current reward

$$\pi^\star(s) = \arg\max_{a\in A}\left(\mathbb{E}[r|s,a] + \gamma\mathbb{E}[V^\star(s')|s,a]\right) = V(s)$$

discounted all reward

$$= \arg\max_{a\in A}\underbrace{\left(\sum_{r\in R} r\mathbb{P}\{r|s,a\} + \gamma\sum_{s'\in S}\mathbb{P}\{s'|s,a\}V^\star(s')\right)}_{Q(s,a)}$$

in next period starting at state s'

- Define the function inside the arg max as the Q function.

Q Function

Definition

$$V^\star(s) = \mathbb{E}[r|s, \pi^\star(s)] + \gamma \mathbb{E}\left[V^\star(s')\,|s, \pi^\star(s)\right]$$
$$Q(s,a) = \mathbb{E}[r|s,a] + \gamma \mathbb{E}\left[V^\star(s')\,|s,a\right]$$

- If the agent knows Q, then the optimal action can be learned without $\mathbb{P}\{s'|s,a\}$.

$$\pi^\star(s) = \arg\max_a Q(s,a)\,, V^\star(s) = \max_a Q(s,a)$$

Deterministic Q Learning

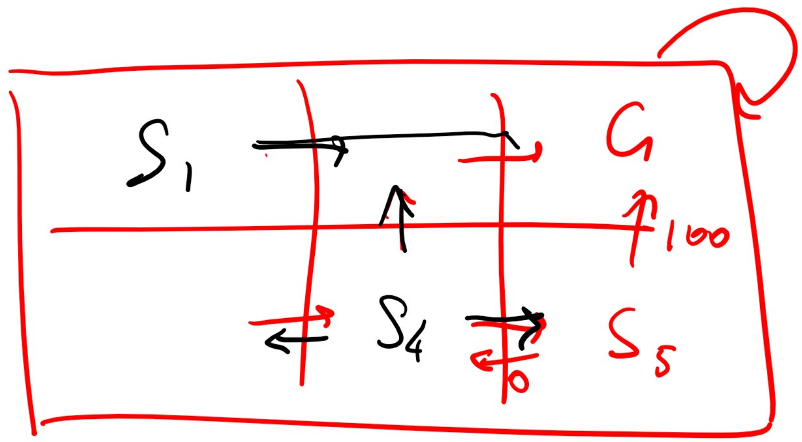
Definition

- In the deterministic case, $\mathbb{P}\{s'|s,a\}$ is either 0 or 1, the update formula for the Q function is the following.

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

Q Learning Example, Part I

Definition



$$Q(S_5, \uparrow) = 100 + 0.9 \cancel{\max_{a'}} \{ \cancel{Q(G, a')} \} = 0$$

$$Q(S_5, \leftarrow) = 0 + 0.9 \max_{a'} \{ \underbrace{Q(S_4, a')}_{\text{initialized at } 0} \} \; 0 = 0$$

$Q(S_4, \uparrow)$, $Q(S_4, \leftarrow)$

$$Q(S_4, \rightarrow) = 0 + 0.9 \max_{a'} \{ Q(S_5, a') \}$$

$$\max \{ \underbrace{\cancel{Q(S_5, \leftarrow)}}_{0}, \underbrace{Q(S_5, \uparrow)}_{100} \}$$

$$= 90$$

Q Learning Example, Part II

Definition

Non-Deterministic Q Learning

Definition

- In the nondeterministic case, the update formula for the Q function is the following.

$$\boxed{\hat{Q}(s,a) = (1-\alpha)\,\hat{Q}(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}(s',a')\right)}$$

$$\alpha = \frac{1}{1 + \text{visits}(s,a)}$$

- Q learning will converge to the correct Q function in both deterministic and non-deterministic cases. In practice, it takes a very large number of iterations.

Q Learning, Part I

Algorithm

- Input: the state and reward processes.
- Output: optimal policy function $\pi^{\star}(s)$
- Initialize the Q table.

$$\hat{Q}(s, a) = 0, \text{ for each } s \in S, a \in A$$

Q Learning, Part II

Algorithm

- Observe current state s.
- Select an action a and execute it.
- Receive immediate reward r.
- Observe the new state s'.
- Update the table entry.

$$\hat{Q}(s,a) = (1-\alpha)\,\hat{Q}(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}(s',a')\right)$$

$$\alpha = \frac{1}{1 + \text{ visits } (s,a)}$$

- Update the state and repeat forever.

$$s = s'$$

Exploration vs Exploitation

Discussion

- There is a trade-off between learning about possibly better alternatives and following the current policy. Sometimes, random actions should be selected.

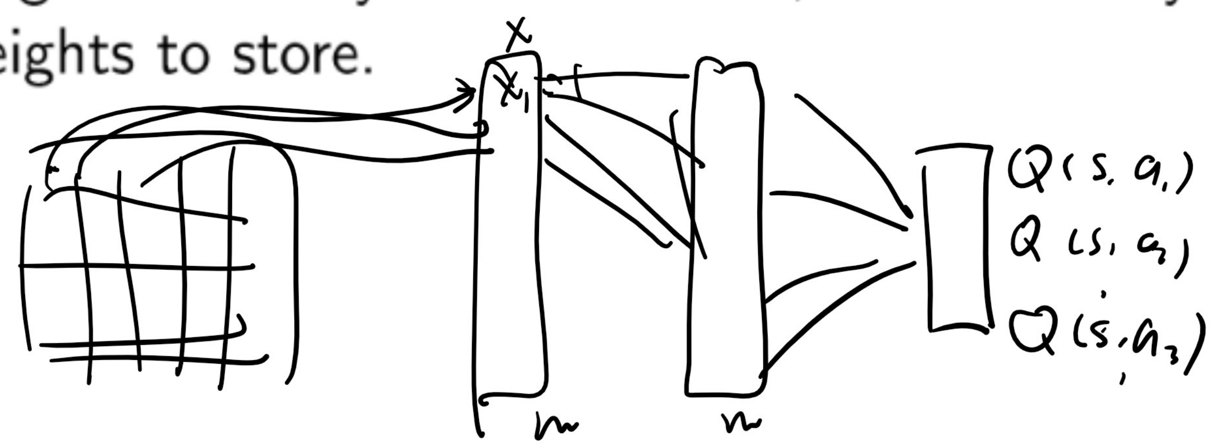
$$\mathbb{P}\{a|s\} = \frac{c^{\hat{Q}(s,a)}}{\sum_{a' \in A} c^{\hat{Q}(s,a')}}$$

- $c > 0$ is a constant that determines how strongly selection favors actions with higher Q values.

Q Table vs Q Net

Discussion

- In practice, Q table is too large to store since the number of possible states is very large.
- If there are m binary features that represent the state, the Q table contains $2^m |A|$.
- However, it can be stored in a neural network called Q net.
- If there is a single hidden layer with m units, there are only $m^2 + m|A|$ weights to store.



Q Net Diagram

Discussion

Q Net Training

Discussion

- Observe the features x given a state s.
- Apply action a and observe new state s' with features x' and reward r.
- Train the network with new instance (x, y)

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

$$y = \underbrace{(1-\alpha)\,\hat{y}(x,a) + \alpha\left(r + \gamma \max_{a'} \hat{y}(x', a')\right)}_{\hat{Q}}$$

activation on (x, a)

- $\hat{y}(x, a)$ is the activation of output unit a given the input x in the current neural network.
- $\hat{y}(x', a')$ is the activation output unit a' given the input x' in the current neural network.

Multi-Agent Learning

Discussion

- Value function and policy function iteration methods can be applied to solve dynamic games with multiple agents.
- It will be used again in game theory in Week 11.