SMU: Lecture 5

Monday, March 7, 2022

(Based heavily on the Stanford RL Course of Prof. Emma Brunskill, but all potential errors are mine.)

Part 0: Where are we?

MDP Control Problem

How to find $\pi^*(s) = \arg\max_{\pi} V^{\pi}(s)$???

State-Action Value Q

- **Definition:**

$$Q^\pi(s,a) = R(s,a) + \gamma \cdot \sum_{s' \in S} P(s' \mid s,a) \cdot V^\pi(s').$$

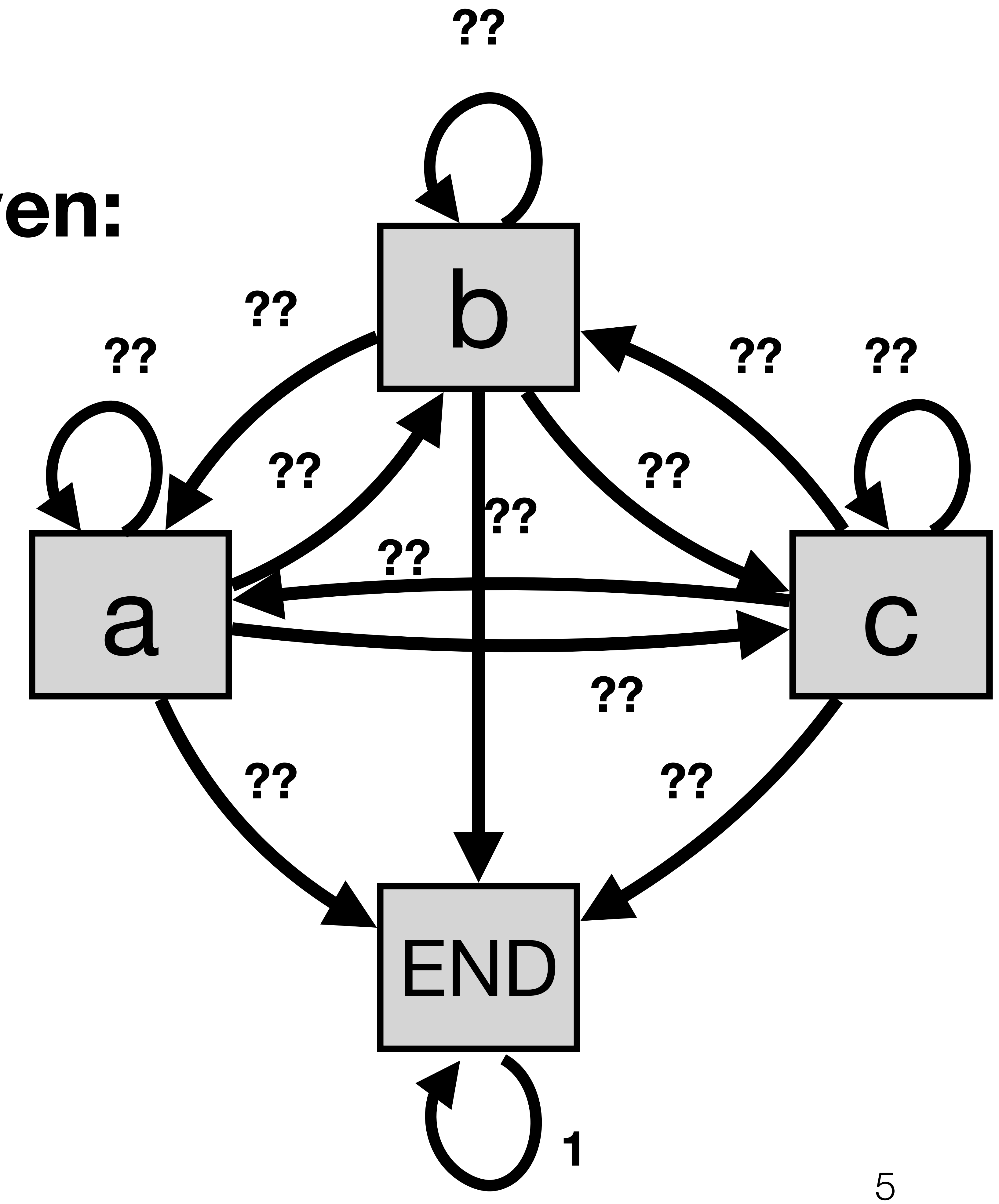
- **Intuition:**
  - The value of the return that we obtain if we first take the action a in the state s and then follow the policy π (including when we visit s again).
  - *Think of it as perturbing the policy π — we deviate from following the policy π only in the first step in s.*

Example

Agent: 

Rewards??

States are given:



Actions are given:

$$A = \{l, r\}$$

Policy is given, e.g.:

$$\pi(l\,|\,a) = 0.2,\ \pi(r\,|\,a) = 0.8,$$
$$\pi(l\,|\,b) = 0.3,\ \pi(r\,|\,b) = 0.7,$$
$$\ldots$$

Model-Free Control

- Given a policy and an MDP with unknown parameters (or generally an environment with which we can interact), **find the optimal policy π.**

Three Methods in Lecture 3

- Monte Carlo Control, SARSA and Q-Learning.
- All three using the concept of ε-greedy policy.

ε-Greedy Policy

We assume ties are decided consistently

$$\pi(a \mid s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|A|} & \text{when } a = \arg\max_{a \in A} Q(s, a) \\ \frac{\varepsilon}{|A|} & \text{when } a \neq \arg\max_{a \in A} Q(s, a) \end{cases}$$

TD-Target

Bellman for Q-function:

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + \gamma \cdot \boxed{\sum_{s_{t+1} \in S} P(s_{t+1} \,|\, s_t, a_t) \cdot \sum_{a_{t+1} \in A} \pi(a_{t+1} \,|\, s_{t+1}) \cdot Q^\pi(s_{t+1}, a_{t+1})}$$

$$\mathbb{E}[Q^\pi(X_{t+1}, A_{t+1}) \,|\, X_t = s_t, A_t = a_t]$$

Temporal difference update (SARSA)…

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha \left(\boxed{r_t + \gamma Q(s_{t+1}, a_{t+1})} - Q(s_t, a_t) \right)$$

Part 1: A Bit More About Convergence…

Convergence of MC, SARSA and Q-Learning

	Tabular	Linear	NN
MC	✅	Chattering (may oscilate at the end but not diverge)	❌
SARSA	✅	Chattering (may oscilate at the end but not diverge)	❌
Q-Learning	✅	❌	❌

Part 2: A Bit More About Deep RL

Last Time: Value-Function Approximation

- MC:

$$\mathbf{w} := \mathbf{w} + \alpha \cdot \left(g_t - \hat{Q}(s_t, a_t; \mathbf{w}) \right) \cdot \nabla \hat{Q}(s_t, a_t; \mathbf{w})$$

- SARSA:

$$\mathbf{w} := \mathbf{w} + \alpha \cdot \left(r + \gamma \hat{Q}(s_{t+1}, a_{t+1}; \mathbf{w}) - \hat{Q}(s_t, a_t; \mathbf{w}) \right) \cdot \nabla Q(s_t, a_t; \mathbf{w})$$

- Q-Learning:

$$\mathbf{w} := \mathbf{w} + \alpha \cdot \left(r + \gamma \max_{a \in A} \hat{Q}(s_{t+1}, a; \mathbf{w}) - \hat{Q}(s_t, a_t; \mathbf{w}) \right) \cdot \nabla Q(s_t, a_t; \mathbf{w})$$

The Same Idea Can Be Used with NNs, but…

Convergence is not guaranteed.

Two of the reasons why Q-learning with VFA may diverge: correlations between samples and non-stationary targets.

Remedies: experience replay and fixed Q-targets.

There are many variations proposed in the literature with many tricks to improve deep Q-learning and many are still appearing…

DQN Pseudocode

Input $\boldsymbol{C}$, $\boldsymbol{\alpha}$, $D = \{\}$, Initialize $\boldsymbol{w}$, $\boldsymbol{w}^- = \boldsymbol{w}$, $t = 0$
Get initial state s_0
loop
    Sample action a_t given ϵ-greedy policy for current $\hat{Q}(s_t, a; \boldsymbol{w})$
    Observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer D
    Sample random minibatch of tuples (s_i, a_i, r_i, s_{i+1}) from D
    for j in minibatch **do**
        if episode terminated at step $i+1$ **then**
            $y_i = r_i$
        else
            $y_i = r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \boldsymbol{w}^-)$
        end if
        Do gradient descent step on $(y_i - \hat{Q}(s_i, a_i; \boldsymbol{w}))^2$ for parameters $\boldsymbol{w}$: $\Delta \boldsymbol{w} = \alpha(y_i - \hat{Q}(s_i, a_i; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s_i, a_i; \boldsymbol{w})$
    end for
    $t = t + 1$
    if mod(t,C) == 0 **then**
        $\boldsymbol{w}^- \leftarrow \boldsymbol{w}$
    end if
end loop

Part 3: Bandits (Introduction)

Efficient Learning

So far we only cared about whether our RL algorithms converge, not that much how fast

We assumed that failed experiments (episodes) do not cost us anything (except, maybe, time). That is the case, e.g., when learning some strategy with a simulator or when playing computer games, but not, e.g., when optimizing an advertisement campaign…

We can generally study efficient learning for MDPs but in this course we will only look at efficient learning for multi-armed bandits (which are simpler but still interesting and used in practice).

Multi-Armed Bandits

1

2

3

4

$$P[R = r \mid A = i]$$

We can choose actions $\{1,2,3,4\}$ *and each of them leads to a different distribution of rewards.*

Setting

Multi-armed bandit is essentially a degenerate MDP that contains a single state.

Definition: A multi-armed bandit is given by:

A set A containing m actions $a_1, a_2, \ldots, a_m$ (each can be thought of as "pulling an arm").

Reward distributions $P[R_t = r \mid A_t = a]$, that is the distribution of rewards at time t given the action at time t.

At each step, the agent takes an action and receives a reward sampled from the above distribution.

The *informal* goal is to maximize the reward $\sum_{t=1}^{T} R_t$.… of course, this is a random variable.

Example

Your PR team created m different advertisements. You are now supposed to show these advertisements to people and maximize the number of times they click on them.

This can be modelled using multi-armed bandits:

The action a_i corresponds to displaying the i-th advertisement from our collection.

We get reward 1 when the person clicks on the advertisement and 0 otherwise.

Clearly, the probabilities $P[R_t = 1 \mid A_t = 1]$, $P[R_t = 1 \mid A_t = 2]$, … will be different (different advertisements will have different quality).

Regret (1/3)

Action-value: $Q(a) = \mathbb{E}[R_t | A_t = a]$.

Similar to MDPs where we had $Q^{\pi}(s, a)$*. However, we do not need* s *because we now have only one state. So we could rewrite it as* $Q^{\pi}(a)$*. But then, since the action only affects the immediate reward and not to which state we get, the whole notion of policy is not very important for* Q *in this setting, so we drop that as well and end up with* $Q(a) = \mathbb{E}[R_t | A_t = a]$.

Regret (2/3)

Optimal value: $V^* = \max_{a \in A} Q(a) = \max_{a \in A} \mathbb{E}[R_t \mid A_t = a].$

Optimal action: $a^* = \arg\max_{a \in A} Q(a).$

Regret: $L_t = V^* - Q(A_t).$

That is, regret is the "opportunity loss" at time t. Note that we use expected value in the definition of regret (recall how we defined $Q(a)$). That means we are not measuring regret directly in terms of what we observe. Since the parameters of bandits will generally be unknown, it also means we will not be able to compute regret directly.

Regret (3/3)

Total regret: $$L_T^{tot} = \sum_{t=1}^{T} L_t = \sum_{t=1}^{T} (V^* - Q(A_t)).$$

Minimizing total regret is the same as maximizing the expected sum of rewards (i.e. return).

Example

Consider again the example with advertisements, say we have 2 different advertisements that we can use, so $A = \{a_1, a_2\}$.

Suppose that:

$P[\text{Person t clicks on ad} \mid A_t = a_1] = 0.8$, $P[\text{Person t clicks on ad} \mid A_t = a_2] = 0.5$

So $\mathbb{E}[R_t \mid A_t = a_1] = 0.8$, $\mathbb{E}[R_t \mid A_t = a_2] = 0.5$.

Let us have the following deterministic sequence of actions:

$a_1, 1, a_1, 0, a_2, 1, a_1, 1, a_2, 0, a_1, 1, a_1, 0, a_1, 1, a_1, 1, a_1, 1$

What is the total regret of this episode?

We have $V^* = 0.8$, $V^* - Q(a_1) = 0$, $V^* - Q(a_2) = 0.8 - 0.5 = 0.3$.

So the total regret is:

$$0 + 0 + 0.3 + 0 + 0.3 + 0 + 0 + \ldots + 0 = 0.6/10 = 0.06.$$

Some More Terminology (Gaps and Counts)

Count: $N_t(a)$ is the number of times the action a was used within the first t time steps.

Gap: We will use the notation $\Delta_a = V^* - Q(a)$ and $\Delta_i = V^* - Q(a_i)$. *It will always be clear from the context which one we use.*

Expected Regret can be written also as:

$$\mathbb{E}[L_T^{tot}] = \sum_{t=1}^{T} \mathbb{E}[\,|V^* - Q(a_t)|\,] = \sum_{a \in A} \mathbb{E}[N_T(a)] \cdot (V^* - Q(a)) = \sum_{a \in A} \mathbb{E}[N_T(a)] \cdot \Delta_a$$

What We Want… (1/2)

We want to find algorithms where the regret will grow slowly with the number of time steps taken.

Note that:

When regret does not grow at all after some time, that means that we are already taking the optimal action.

Regret is the difference between best possible return and the return under our strategy. So when the regret grows slowly, it means we are already doing quite well.

What We Want… (2/2)

If we knew the expectations $\mathbb{E}[R_t | A_t = a]$ *then the problem would be trivial, but it would not be reinforcement learning.*

We could try to first estimate $\mathbb{E}[R_t | A_t = a]$ *by taking actions completely randomly. However, then in this first part we would incur high regret and it is also not clear how long we should be estimating (because that actually depends on the values of* $\mathbb{E}[R_t | A_t = a]$*)… So we will need something smarter.*

Greedy Methods (Why They Would Not Work)

Greedy Algorithm

Initialization: Do several passes over all actions and compute estimates $\hat{Q}(a)$ for all $a \in A$.
Maintain counter $N(a)$ with the number of times an action was used.

While (some stopping condition):

Select the action $a_t \in A$ which maximizes $\hat{Q}(a)$.

Use the selected action and observe r_t.

Set $N(a_t) := N(a_t) + 1$.

Set $\hat{Q}(a_t) := \hat{Q}(a_t) + \frac{1}{N(a_t)}(r_t - Q(a_t))$.**

$$**\left(\underbrace{Q(a_t)}_{=\frac{1}{N(a_t)-1}(r_{i_1}+\ldots+r_{i_{t-1}})} + \frac{1}{N(a_t)}r_{i_t} - \frac{1}{N(a_t)}Q(a_t) = \frac{N(a_t)(r_{i_1}+\ldots+r_{i_{t-1}}) + (N(a_t)-t)r_{i_t} - (N(a_t)-1)\frac{1}{N(a_t)-1}(r_{i_1}+\ldots+r_{i_{t-1}})}{(N(a_t)-1)N(a_t)} \right.$$

$$\left. = \frac{(N(a_t)-1)(r_{i_1}+\ldots+r_{i_{t-1}}) + (N(a_t)-t)r_{i_t}}{(N(a_t)-1)N(a_t)} = \frac{1}{N(a_t)}(r_{i_1}+r_{i_2}+\ldots+r_{i_t}) \right)$$

Why Greedy Will Not Work Well

This will be similar to why purely greedy methods do not work well for RL (as we saw before, where we solved the problem by using ε-greedy methods.

Example (Continue with our previous example):

$$\mathbb{E}[R_t | A_t = a_1] = 0.8, \mathbb{E}[R_t | A_t = a_2] = 0.5.$$

For greedy methods, we need some initialization (e.g. passing over all the actions a couple of times).

Suppose that our initial estimates for Q are $\hat{Q}(a_1) = 0$ and $\hat{Q}(a_2) = 0.5$ (which can happen if we are unlucky in the initialization).

Then we will never select a_1 even though it is the optimal action. So **regret will grow linearly with time** in this case.

ε-Greedy Methods (*Also not that great…*)

ε-Greedy (Basic Idea)

Similarly to what we did in the previous lectures…

Initialization: Do several passes over all actions and compute estimates $\hat{Q}(a)$ for all $a \in A$. Maintain counter $N(a)$ with the number of times an action was used.

While (some stopping condition):

With probability $1 - \varepsilon$:

Select the action $a_t \in A$ which maximizes $\hat{Q}(a)$.

Else:

Select an action $a_t \in A$ uniformly at random.

Use the selected action and observe r_t.

Set $N(a_t) := N(a_t) + 1$.

Set $\hat{Q}(a_t) := \hat{Q}(a_t) + \dfrac{1}{N(a_t)}(r_t - Q(a_t))$.

Regret of ε-Greedy Methods

If we keep ε constant during the run of the ε-greedy algorithm then we will incur regret growing linearly with the number of time steps—in every step we have probability $\varepsilon - \frac{\varepsilon}{|A|}$ of picking a suboptimal action (assuming no ties) which will incur a regret of at least $V^* - \max_{a \neq a^*} Q(a)$

So also not great…

We might try to set ε to be a function of t (as we did before) but it turns out to be tricky and need to know a lot about $Q(a)$'s in advance.

Optimism Under Uncertainty

UCB Algorithm: Basic Idea

Upper-Confidence Bound (UCB) Algorithm

For every action $a \in A$, maintain an upper bound $U_t(a)$ *(the upper bound will change with time, that is why it is indexed by t).*

In every time step t, take the action that has the maximum upper bound, i.e. take the action $\arg\max_{a \in A} U_t(a)$.

After observing the reward, update the estimates.

UCB Algorithm

Initialization:

Take every action $a \in A$ once and record the rewards in $\hat{Q}(a)$.

$t := 1$

Loop:

Compute upper confidence bounds for all actions $a_i \in A$:

$$U_t(a_i) = \hat{Q}(a_i) + \sqrt{\frac{1}{2N(a_i)} \log \frac{2t^2}{\delta}}$$

Use the action $a_t = \arg\max_{a \in A} U_t(a)$ and observe the reward r_t.

Update $N(a_t) := N(a_1) + 1$

Update $\hat{Q}(a_t) := \hat{Q}(a_t) + \frac{1}{N(a_t)}(r_t - Q(a_t))$.

$t := t + 1$

Proof (1/12)

Claim: If all upper bounds $U_t(a_1), U_t(a_2), \ldots, U_t(a_m)$ satisfy $U_t(a_i) \geq Q(a_i)$, i.e. if none of them underestimates the true value, then for the action a_t selected at time t, it must hold

$$U_t(a_t) \geq U(a^*) = V^*.$$

Easy to see why…

Proof (2/12)

First, we will state an auxiliary statement (*which you probably know from other courses).*

Theorem (Hoeffding's Inequality): Let $X_1, X_2, \dots, X_N$ be independent random variables bounded on the interval $[a; b]$. Let $\overline{X}_N = \frac{1}{N} \sum_{i=1}^{N} X_i$. Then it holds

$$P\left[\overline{X}_N - \mathbb{E}[\overline{X}_N] \geq \xi\right] \leq \exp\left(-\frac{2N\xi^2}{(b-a)^2}\right),$$

$$P\left[\mathbb{E}[\overline{X}_N] - \overline{X}_N \geq \xi\right] \leq \exp\left(-\frac{2N\xi^2}{(b-a)^2}\right),$$

$$P\left[|\overline{X}_N - \mathbb{E}[\overline{X}_N]| \geq \xi\right] \leq 2\exp\left(-\frac{2N\xi^2}{(b-a)^2}\right).$$

Proof (3/12)

Our $\overline{X}_N$ will be $\hat{Q}_t(a_i)$, i.e. the estimate of $\hat{Q}(a_i)$, and our N will therefore be $N_t(a_i)$, i.e. number of times a_i was used.

We have $\mathbb{E}[\hat{Q}_t(a_i)] = Q(a_i)$.

We will want to find ξ_t (one value for each t) such that

$$P\left[\,|\,Q(a_i) - \hat{Q}(a_i)\,| \geq \xi_t\right] \leq 2\exp\left(-\frac{2N_t(a_i)\xi_t^2}{(b-a)^2}\right) = \frac{\delta}{t^2},$$

where t is the current number of time steps.

Proof (4/12)

We have

$$P\left[\,|\,Q(a_i) - \hat{Q}(a_i)\,|\, \geq \xi_t\right] \leq 2\exp\left(-\frac{2N_t(a_i)\xi_t^2}{(b-a)^2}\right) = \frac{\delta}{t^2},$$

$$-\frac{2N(a_i)\xi_t^2}{(b-a)^2} = \log\frac{\delta}{2t^2},$$

$$\xi_t = (b-a)\sqrt{\frac{1}{2N_t(a_i)}\log\frac{2t^2}{\delta}}$$

For simplicity we will now assume that $a = 0,\, b = 1$.

Proof (5/12)

That is, the upper bounds $U_t(a_i)$ will be:

$$U_t(a_i) = \hat{Q}(a_i) + \sqrt{\frac{1}{2N_t(a_i)} \log \frac{2t^2}{\delta}}.$$

And we will also have lower bounds $L_t(a_i)$:

$$L_t(a_i) = \hat{Q}(a_i) - \sqrt{\frac{1}{2N_t(a_i)} \log \frac{2t^2}{\delta}}.$$

Proof (6/12)

Let A_t be the **action selected at time t.**

We will now bound the probability that at least some of the bounds are incorrect *(we will see in a moment why we want this)*.

$$P\left[\bigvee_{t=1}^{T}\bigvee_{i=1}^{m} U_t(A_i) \notin [L_t(A_i); U_t(A_i)]\right] \leq$$

$$\leq \sum_{t=1}^{T}\sum_{i=1}^{m} P[\,|Q(a_i) - \hat{Q}_t(a_i)| > \xi_t] \leq \sum_{t=1}^{T}\sum_{i=1}^{m}\frac{\delta}{t^2} = m\delta\sum_{t=1}^{T}\frac{1}{t^2}.$$

Proof (7/12)

We can now use the famous identity $\sum_{t=1}^{\infty} \frac{1}{t^2} = \frac{\pi^2}{6}$ *(which is smaller than 2)*.**

So we can bound:

$$P\left[\bigvee_{t=1}^{T}\bigvee_{i=1}^{m} U_t(A_i) \notin [L_t(A_i); U_t(A_i)]\right] \leq 2m\delta.$$

That means that the probability that all lower and upper bounds are valid at all time steps is at least $1 - 2m\delta$.

We will use this in a moment.

**** We actually do not need this fancy result to get the constant 2 (see the additional slide)**

Proof (8/12)

Let A_t be the **action selected at time t.**

We will now bound the probability that at least one of the upper bounds $U_1(A_1), U_2(A_2), \dots$ is lower than $U(a^*)$.

We can notice that the event that at least one action has wrong confidence bounds over the course of T time steps, formally written as

$$\bigvee_{t=1}^{T} \bigvee_{i=1}^{m} U_t(A_i) \notin [L_t(A_i); U_t(A_i)]$$

is a necessary condition for at least one of the upper bounds $U_1(A_1), U_2(A_2), \dots$ to be lower than $U(a^*)$.

Therefore we can bound this probability also by $1 - 2\delta m$.

Proof (9/12)

Let us now compute the regret of this algorithm:

$$\text{Regret}(T) = \sum_{t=1}^{T} \big(Q(a^*) - Q(A_t)\big) = \sum_{t=1}^{T} \big(U_t(A_t) - Q(A_t) + Q(a^*) - U_t(A_t)\big)$$

We have that $Q(a^*) < U_t(A_t)$ with probability at least $1 - 2m\delta$ (*from the previous slide!*) Hence we can bound the above as:

$$\text{Regret}(T) \leq \sum_{t=1}^{T} \big(U_t(A_t) - Q(A_t)\big).$$

Proof (10/12)

Now we will play with

$$\text{Regret}(T) \leq \sum_{t=1}^{T} \big(U_t(A_t) - Q(A_t)\big).$$

Recall that we defined $U_t(a_i) = \hat{Q}(a_i) + \sqrt{\frac{1}{2N_t(a_i)} \log \frac{2t^2}{\delta}}$ for all $a_i \in A$.

Hence we get

$$\text{Regret}(T) \leq \sum_{t=1}^{T} \left(\hat{Q}(A_t) + \sqrt{\frac{1}{2N_t(A_t)} \log \frac{2t^2}{\delta}} - Q(A_t) \right).$$

Proof (11/12)

Now we need to do something with

$$\text{Regret}(T) \leq \sum_{t=1}^{T} \left(\hat{Q}(A_t) + \sqrt{\frac{1}{2N_t(A_t)} \log \frac{t^2}{\delta}} - Q(A_t) \right).$$

Since we have that, with probability at least $1 - 2\delta m$, we have for all $a_t \in A$

$$\left| \hat{Q}(a_t) - Q(a_t) \right| \leq \sqrt{\frac{1}{2N_t(A_t)} \log \frac{t^2}{\delta}}.$$

We can bound the regret, with probability at least $1 - 2\delta m$, as

$$\text{Regret}(T) \leq \sum_{t=1}^{T} 2\sqrt{\frac{1}{2N_t(A_t)} \log \frac{t^2}{\delta}} = \sum_{t=1}^{T} \sqrt{\frac{2}{N_t(A_t)} \log \frac{t^2}{\delta}}.$$

Proof (12/12)

Finally we have, with probability at least $1 - 2\delta m$,

$$\text{Regret}(T) \leq \sum_{t=1}^{T} \sqrt{\frac{2}{N_t(A_t)} \log \frac{t^2}{\delta}} = \sqrt{\log \frac{t^2}{\delta}} \sum_{t=1}^{T} \sqrt{\frac{2}{N_t(A_t)}} =$$

$$= \sqrt{2 \log \frac{t^2}{\delta}} \sum_{i=1}^{m} \sum_{j=1}^{N_T(a_i)} \sqrt{\frac{1}{j}} \leq 2\sqrt{\frac{Tm}{2} \log \frac{T^2}{\delta}}.$$

Sublinear regret!!!!

Conclusions

- There is a lot more about bandits than we could cover here… and about sample-efficient reinforcement learning in general.

If you want to know more…

Lattimore, Tor, and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Available online: https://tor-lattimore.com/downloads/book/book.pdf

Additional Slide

(Why $\sum_{t=1}^{\infty} \frac{1}{t^2} = \frac{\pi^2}{6}$ is not needed)

Bounding

$$\sum_{t=1}^{\infty} \frac{1}{t^2} \leq 1 + \int_1^{\infty} \frac{1}{t^2} dt = 1 + \left[-\frac{1}{t} \right]_1^{\infty} = 2.$$