COL778: Principles of Autonomous Systems
Semester II, 2023-24

Learning with Multi-armed Bandits

Rohan Paul

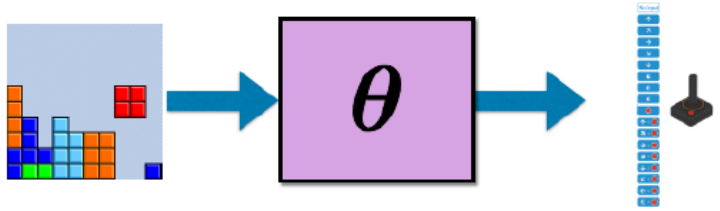
Outline

- Last Class
  - Policy Gradients
- This Class
  - Bandit problems
- Reference Material
  - Please follow the notes as the primary reference on this topic.

Acknowledgements

These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Nicholas Roy, Wolfram Burgard, Dieter Fox, Sebastian Thrun, Siddharth Srinivasa, Dan Klein, Pieter Abbeel, Max Likhachev, Alexander Amini (MIT Introduction to Deep Learning) and others. This lecture is primarily built on the slides from Katerina Fragkiadaki at CMU.

Recap: RL Problem



$\pi(a \mid s, \theta)$

Given an initial state distribution $\mu_0(s_0)$, estimate parameters θ of a policy π_θ so that, the trajectories τ sampled from this policy have maximum returns, i.e., sum of rewards $R(\tau)$.

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \mid \pi_\theta, \mu_0(s_0) \right]$$

τ : trajectory, a sequence of state, action, rewards, a game fragment or a full game:

$$\tau : s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots s_T, a_T, r_T$$

$R(\tau)$: reward of a trajectory: (discounted) sum of the rewards of the individual state/actions

$$R(\tau) = \sum_{t=1}^{T} r_t$$

This lecture - Motivation

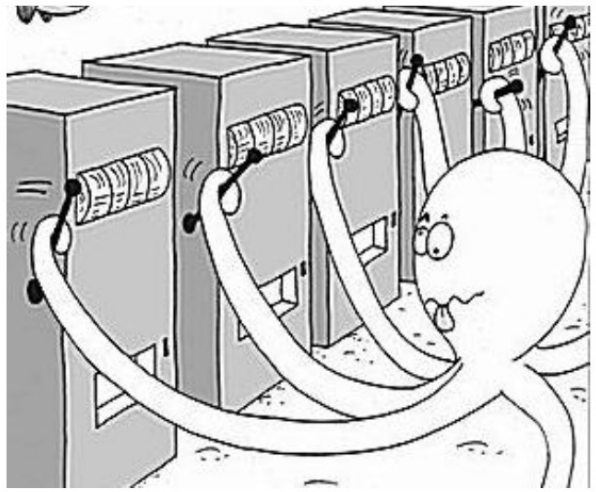
- Learning to act in a non-sequential (single action) setups:
- Each action results in an immediate reward.
- We want to choose actions that maximize our immediate reward in expectation.
- For example, choosing which restaurant to go to.
  - Actions: the restaurants to choose from for food
  - Rewards: satisfaction with the food (unknown, you may or may not be happy).
- What if you become friends with the restaurant employees and enjoy interacting with them?
  - Now, that the state changes, this is now a sequential problem (use standard RL).

Multi-Armed Bandits

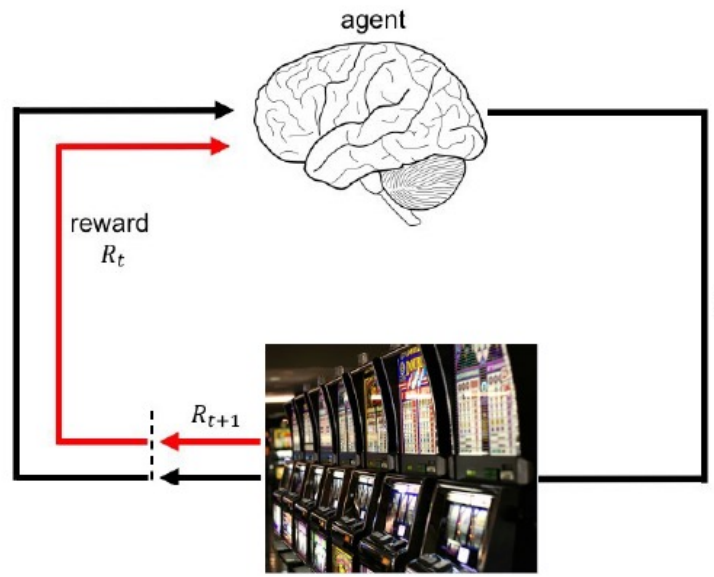
One-armed slot machine.

source: infoslotmachine.com

Multi-armed slot machine.



source: Microsoft Research



$$A_t, R_{t+1}, A_{t+1}, R_{t+2}, A_{t+2}, A_{t+3}, R_{t+3}, \ldots$$

The state does not change! (a.k.a. stateless)

Multi-Armed Bandit Problem

At each timestep t the agent chooses one of the K arms and plays it.

The k th arm produces reward $r_{k,t}$ when played at timestep t .

The rewards $r_{k,t}$ are drawn from a probability distribution $\mathcal{P}_k$ with mean μ_k .

The agent does not know neither the reward distributions neither their means.

source: Pandey et al.'s slide

Agent's Objective: Maximize cumulative rewards (over a finite or infinite horizon).

I can maximize cumulative rewards over a finite or infinite horizon if i just play the arm with the highest **mean** reward μ_k each time. (but i do not know which one it is.)

Multi-Armed Bandit Problem

At each timestep t the agent chooses one of the K arms and plays it.

The k th arm produces reward $r_{k,t}$ when played at timestep t .

The rewards $r_{k,t}$ are drawn from a probability distribution $\mathcal{P}_k$ with mean μ_k .

The agent does not know neither the reward distributions neither their means.

source: Pandey et al.'s slide

Definition: The action-value for action α is its mean reward:

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

The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a$$ *action-value estimates*

- Define the *greedy action* at time *t* as

$$A_t^* \doteq \arg\max_a Q_t(a)$$

- If $A_t = A_t{}^*$ then you are *exploiting*
  If $A_t \neq A_t{}^*$ then you are *exploring*

Exploration/Exploitation is core to reinforcement learning.

- You can't do both, but you need to do both
- You can never stop exploring, but maybe you should explore less with time

Exploration vs. Exploitation Dilemma

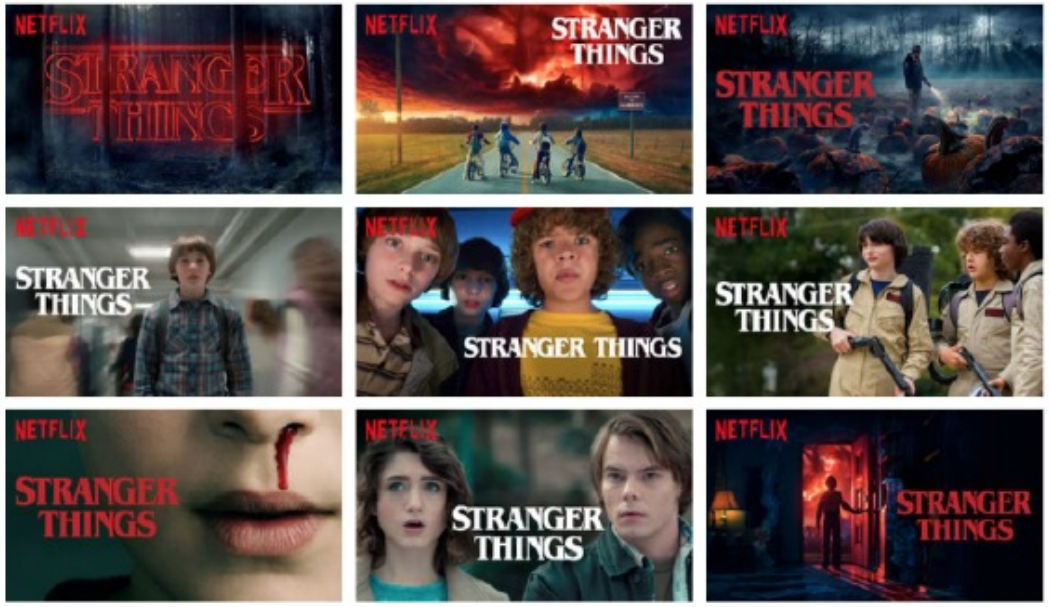
- Picking the opening batsman/batswomen
  - Exploitation: Go with the openers with the highest average.
  - Exploration: Ask some new player to open the innings.
- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- Oil Drilling
  - Exploitation: Drill at the best-known location
  - Exploration: Drill at a new location
- Game Playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Real world motivation: NETFLIX artwork

For a particular movie, we want to decide what image to show (to all the NEFLIX users)

- Actions: uploading one of the K images to a user's home screen
- Reward: 1 if the user clicks and watches, 0 otherwise.
- Mean reward (success probability) for each image: the percentage of users that clicked and watched

Problem of selecting images to show on your log in screen.



Variations based on Unknown Reward Model: Bernoulli Bandits

- Each action results in success or failure, rewards are binary.
- Mean reward for each arm represents the probability of success.
- Action $k \in \{1 \ldots K\}$ produces a success with probability $\theta_k \in [0,1]$.

θ_1

win 0.6
of time

θ_2

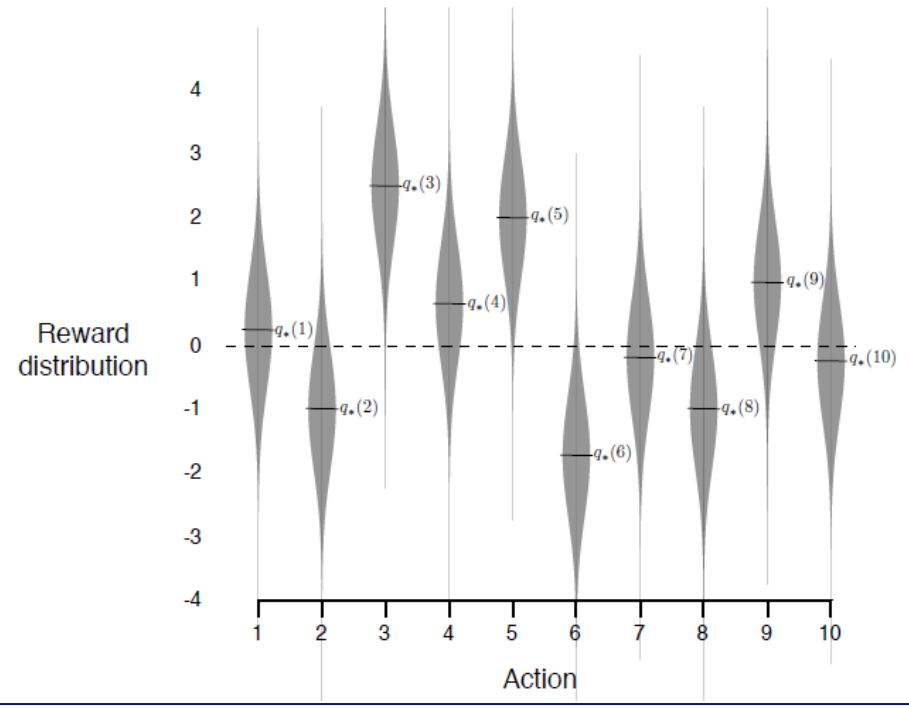
win 0.4
of time

θ_3

win 0.45
of time

Variations based on Unknown Reward Model: Gaussian Bandits

- Each action results in a real number.
- Action $k \in \{1 \ldots K\}$ produces on average reward equal to the mean of its Gaussian distribution.



Notion of Regret

- The action-value is the mean reward for action a,

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a], \quad \forall a \in \{1, \ldots, k\}$$

- The optimal value is

$$v_* = q(a^*) = \max_{a \in \mathcal{A}} q_*(a)$$

- The regret is the expected opportunity loss for one step. For an algorithm that selects action a_t at timestep t it reads:

$$I_t = \mathbb{E}[v_* - q_*(a_t)] \qquad \text{reward} = -\text{regret}$$

- The total regret is the total opportunity loss

$$L_T = \mathbb{E}\left[\sum_{t=1}^{T} v_* - q_*(a_t)\right]$$

- Maximize cumulative expected reward = minimize total regret

Forming Action-Value Estimates

- To simplify notation, let us focus on one action
  - Its action value estimate after n-1 rewards:

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Essentially, averaging the rewards acquired.

- How can we do this incrementally (without storing all the rewards)?
- Could store a running sum, or equivalently:

$$Q_{n+1} = Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]$$

- This is a standard form for learning/update rules:

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \; \Big[Target - OldEstimate\Big]$$

How to learn the reward model/solve the MAB problem?

Option I: A fixed exploration period + Greedy after that

1. Allocate a fixed time period to exploration when you try bandits uniformly at random

2. Estimate mean rewards for all actions: $Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{t-1} r_i \mathbf{1}(A_i = a)$

3. Select the action that is optimal for the estimated mean rewards, breaking ties at random: $a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q_t(a)$

4. GOTO 2

Problem with random + greedy

- After the fixed exploration period we have formed the following reward estimates

$Q_t(a_1) = 0.3$

$Q_t(a_2) = 0.5$

$Q_t(a_3) = 0.1$

Q1: Will the greedy method always pick the second action?

Q2: Can greedy lock onto a suboptimal action forever?

⇒ Greedy has linear total regret

Linear in time. Why? if t more steps are taken then we simply get (t*regret).

Regret

- The count $N_t(a)$: the number of times that action a has been selected prior to time t
- The gap Δa is the difference in value between action a and optimal action a_*: $\Delta_a = v_* - q_*(a)$
- Regret is a function of gaps and the counts

$$L_T = \mathbb{E}\left[\sum_{t=1}^{T} v_* - q_*(a_t)\right]$$

$$= \sum_{a \in \mathcal{A}} \mathbb{E}[N_T(a)](v_* - q_*(a))$$

$$= \sum_{a \in \mathcal{A}} \mathbb{E}[N_T(a)]\Delta_a$$

ε-Greedy Action Selection

Option II: ε-Greedy Action Selection. Exploring all the time but adjusting. Reduce exploration over time.

- In greedy action selection, you always exploit
- In ε-greedy, you are usually greedy, but with probability ε you instead pick an action at random (possibly the greedy action again)
- This is perhaps the simplest way to balance exploration and exploitation

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

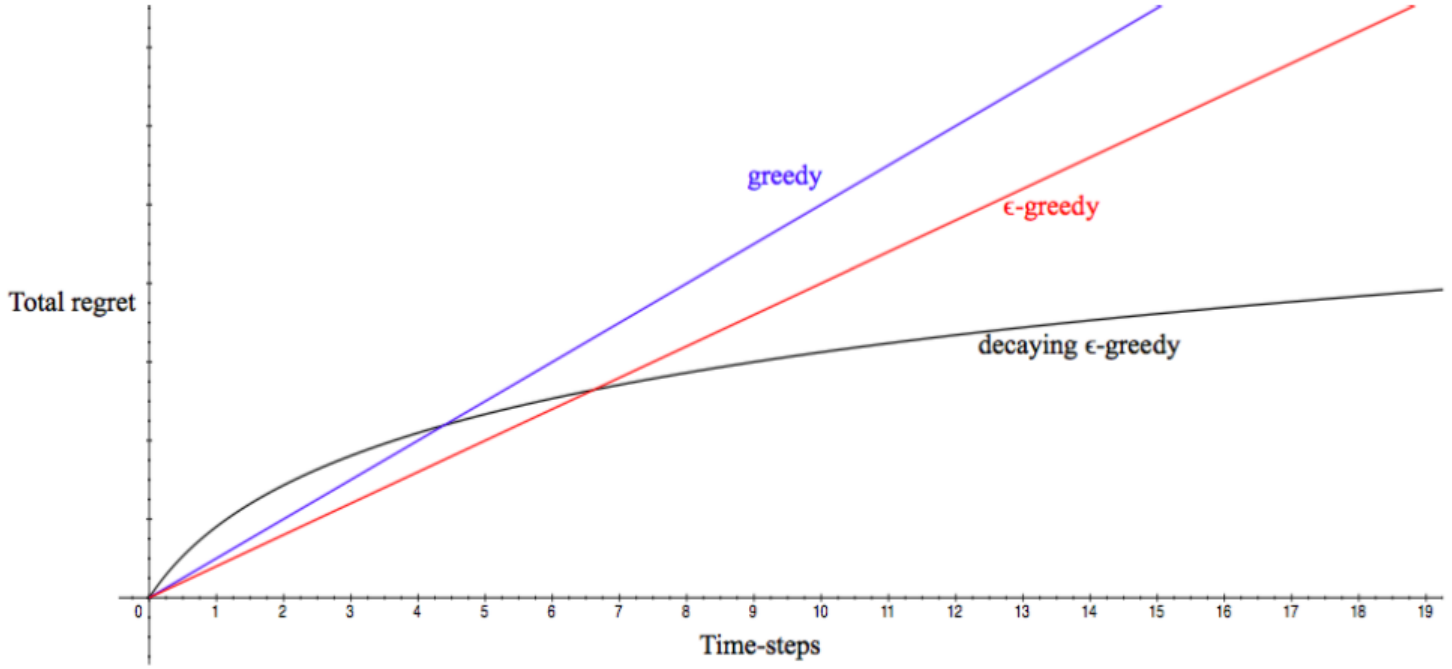
ε-Greedy Algorithm

- The ε-greedy algorithm continues to explore forever
  - With probability 1 – ε select $a_t = \text{argmax}_{a \in \mathcal{A}} Q_t(a)$
  - With probability ε select a random action (independent of its Q estimate)
- Constant ε results in the per time step regret to be:

$$I_t \geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

- ⇒ ε-greedy has linear total regret

Counting Regret



- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret

Incorporating Uncertainty in Exploration

1000 pulls,
600 wins
Q_t(a_1)=0.6

1000 pulls,
400 wins
Q_t(a_2)=0.4

10 pulls,
4 wins
Q_t(a_1)=0.4

Epsilon-greedy

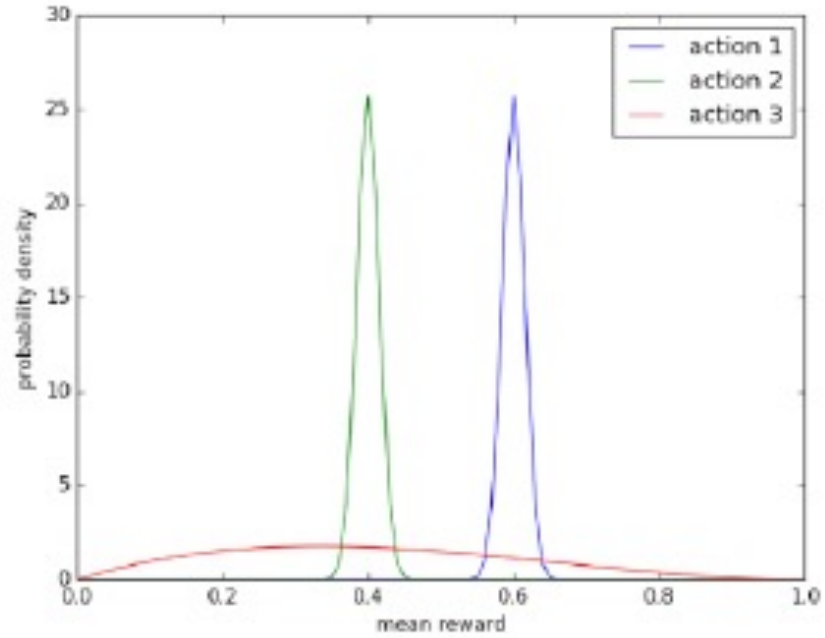
Repeat forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$



We may have played the arm on a smaller number of occasions and hence may have more uncertainty due to that.

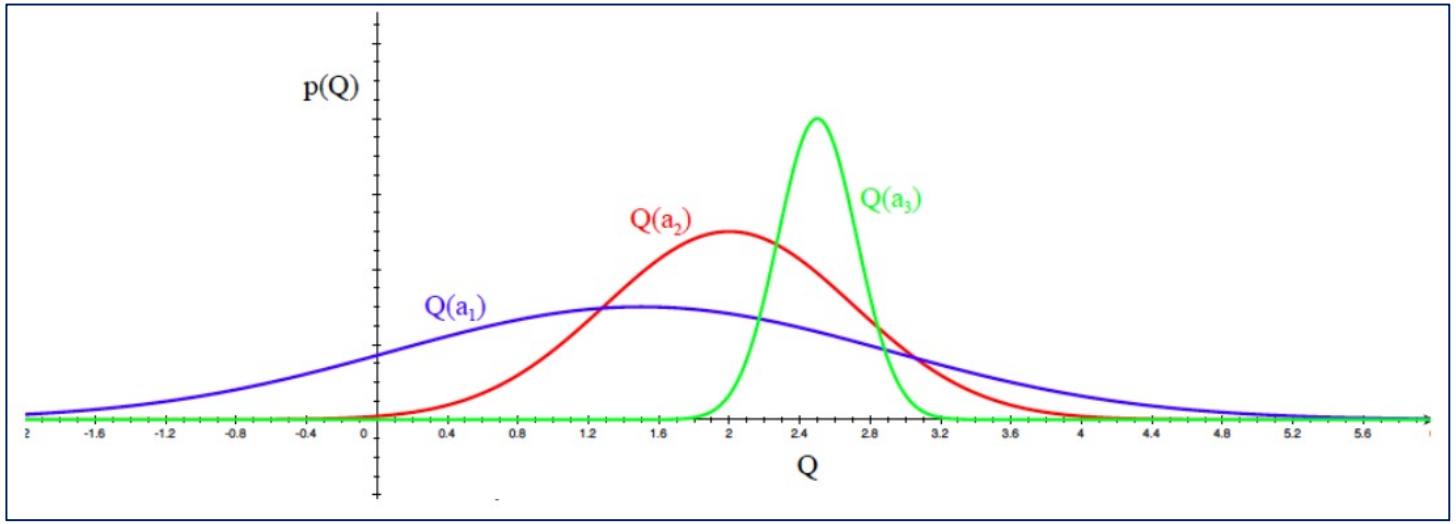
The problem with using mean estimates is that we do not reason about uncertainty of those estimates

Incorporating Uncertainty in Exploration

Need to reason about uncertainty in our estimates of the action values.

Need to reason about how uncertain is the reward estimate for an arm.

If there is “high” uncertainty in the reward distribution for an arm, the agent should try that action “more” times to get better estimates.



How to incorporate uncertainty into MAB framework action selection?

Upper Confidence Bounds

Core Idea:

- Estimate an upper bound on the "action value". Confidence that with high likelihood the reward for an arm will not exceed a given upper bound.
- If the variance in the estimate is high then it should be "preferred" for exploration.
- Action selection should take into account the uncertainty and not simply be greedy.

- Estimate an upper confidence $U_t(a)$ for each action value such that with high probability:

$$q_*(a) \leq Q_t(a) + U_t(a)$$

Estimated mean

Estimated Upper Confidence

- This upper confidence depends on the number of times action a has been selected
  - Small $N_t(a) \Rightarrow$ large $U_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $U_t(a)$ (estimated value is accurate)
- Select action maximizing Upper Confidence Bound (UCB)

$$a_t = \text{argmax}_{a \in \mathcal{A}} Q_t(a) + U_t(a)$$

Hoeffding's Inequality

Core Idea:

- Provides an upper bound on a probability.
- The probability is that the sum of bounded independent random variables deviates (positive or negative side) from its expected value.
- By a certain amount.

Note: think of the blue as the sample mean (the estimate we are doing) from samples we are getting from the random variable (reward in our case).

Let $X_1, \ldots X_t$ be independent random variables in the range [0,1] with $\mathbb{E}(X_i) = \mu$. Then for $u > 0$,

sample mean

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq \mu + u\right) \leq e^{-2u^2 n}$$

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i \leq \mu - u\right) \leq e^{-2u^2 n}$$

Hoeffding's Inequality

Core Idea:

- Apply the Hoeffding's inequality to the rewards observed from each action (bandit) a.
- U_{t}(a) is the margin and can depend on t, the amount of interactions t that have been done with this arm.

Let $X_1, \ldots X_n$ be independent random variables in the range [0,1] with $\mathbb{E}(X_i) = \mu$. Then for $u > 0$,

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq \mu + u\right) \leq e^{-2u^2 n}$$

I made the margin to depend on the amount of interactions t

- We will apply Hoeffding's Inequality to the rewards obtained from each action (bandit) a:

$$\mathbb{P}\left(\hat{Q}_t(a) \geq q(a) + U_t(a)\right) \leq e^{-2U_t(a)^2 N_t(a)}$$

  - t: how many times I have played any action,
  - $N_t(a)$: how many times I have played action a in t interactions

Calculating Upper Confidence Bounds

Core Idea:

- Introduce a probability p.
- The probability is the likelihood value estimate to lie between a deviation of U_{t}(a) of the mean.
- Given p, solve for what the deviation U_{t}(a) should be.

- Pick a probability p that the value estimate deviates from its mean
- Now solve for $U_t(a)$

$$e^{-2U_t(a)^2 N_t(a)} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Reduce p as we play more, e.g. $p = t^{-c}, c = 4$
- Ensures we select optimal action as t → ∞

$$U_t(a) = \sqrt{\frac{2 \log t}{2N_t(a)}}$$

Upper Confidence Bound (UCB)

Core Idea:

- Formulation allows the exploration term to reduce over time.
- Action selection is inherently doing a trade off between exploitation (the action estimates) and exploration (the second term).

- Estimate an upper bound on the true action values
- Select the action with the largest (estimated) upper bound

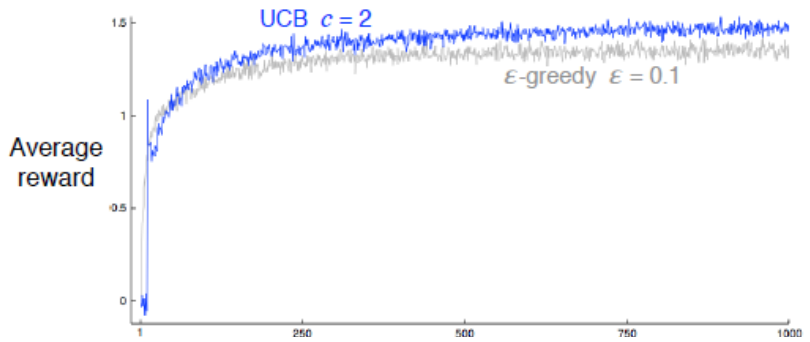
$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]$$

- c is a hyper-parameter that trades-off explore/exploit
- the confidence bound grows with the total number of actions we have taken t but shrinks with the number of times we have tried this particular action $N_t(a)$. This ensures each action is tried infinitely often but still balances exploration and exploitation.

UCB1 Algorithm

- This leads to the UCB1 algorithm

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$



Theorem

The UCB algorithm achieves logarithmic asymptotic total regret

$$\lim_{t \to \infty} L_t \leq 8 \log t \sum_{a | \Delta_a > 0} \Delta_a$$

This is a good result! The UCB regret grows at a lower rate (than linear).

Bayesian Bandits: Updating Priors over the Latent Reward Distribution

- So far we have made no assumptions about the reward distributions.
  - In UCB we just considered some bounds on rewards
- Bayesian bandits exploit prior knowledge of rewards, $p[\mathcal{R}]$
- They compute posterior distribution of rewards $p[\mathcal{R} \mid h_t]$

$$h_t = a_1, r_1, \ldots, a_{t-1}, r_{t-1}$$

- Use posterior to guide exploration: we simply sample from the posterior!

Core Idea: represent the full reward distribution rather than the mean reward estimate alone.

Bayesian Learning of Model Parameters

Step 1: Given n data, $D = x_{1...n} = \{x_1, x_2, \ldots, x_n\}$ write down the expression for likelihood:

$$p(D \mid \theta)$$

Step 2: Specify a prior: $p(\theta)$

Step 3: Compute the posterior:

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{p(D)}$$

This is more robust that point estimate. This is a Bayesian way to model problems.

Assume a distribution that characterizes the reward probability. The rewards observed are samples from this ”latent” distribution that we estimate. Let the reward distribution be parameterized by theta. Once we see samples from the distribution we update what the parameters of the distribution should be.

Thompson Sampling

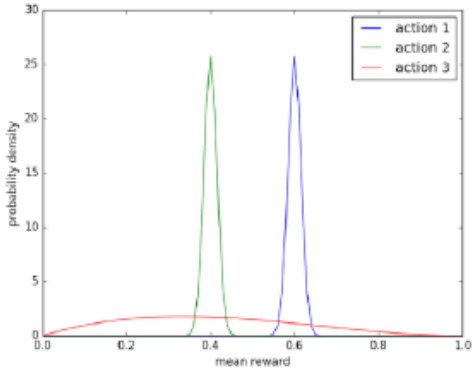
Represent a distribution for the mean reward of each bandit as opposed to the mean reward estimate alone. At each timestep:

1. Sample from the mean reward distributions:
   $\bar{\theta}_1 \sim \hat{p}(\theta_1), \bar{\theta}_2 \sim \hat{p}(\theta_2), \cdots, \bar{\theta}_k \sim \hat{p}(\theta_k)$

2. Choose action $a = \arg\max_a \mathbb{E}_{\bar{\theta}}[r(a)]$

3. Observe the reward.

4. Update the mean reward posterior distributions: $\hat{p}(\theta_1), \hat{p}(\theta_2) \cdots \hat{p}(\theta_k)$



Introducing Priors

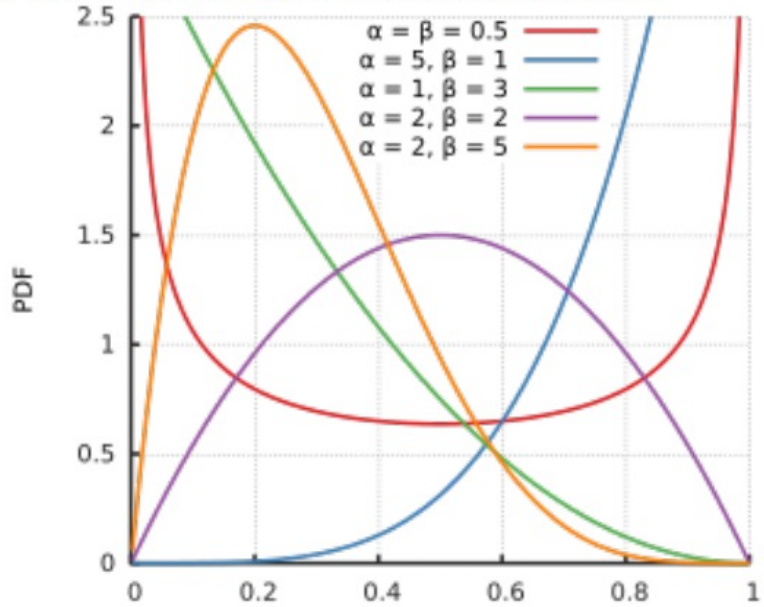
Let's consider a Beta distribution prior over the mean rewards of the Bernoulli bandits:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1}(1 - \theta_k)^{\beta_k - 1} \qquad \Gamma(n) = (n-1)!$$

The mean is $\frac{\alpha}{\alpha + \beta}$

The larger the $\alpha + \beta$ the more concentrated the distribution

$\text{Beta}(\alpha, \beta)$



Update the Prior Distribution with Data

Let's consider a Beta distribution prior over the mean rewards of the Bernoulli bandits:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1}(1 - \theta_k)^{\beta_k - 1} \qquad \Gamma(n) = (n-1)!$$

The posterior is also a Beta because Beta is conjugate distribution for the Bernoulli distribution.

A closed form solution for the bayesian update, possible only for conjugate distributions.

$$(\alpha_k, \beta_k) \leftarrow \begin{cases} (\alpha_k, \beta_k) & \text{if } x_t \neq k \\ (\alpha_k, \beta_k) + (r_t, 1 - r_t) & \text{if } x_t = k. \end{cases}$$

Example: MAB with Context

Recommendation system for an OTT platform:

- Users are interested in a particular genre of movies. You are to display movie thumbnails for a particular user.
- Mean rewards (unknown): the % of users that will click on the title and watch the movie. This is unknown and you want to maximize the reward.

Context:

- User attributes, e.g., language preferences, genre of films, history of the user, time of the day etc.
- Multi-armed bandit but there is a context.

Learning:

- Your algorithm will estimate the mean rewards: the average click rate (+quality engagement

You want to display thumb nails for a particular user. Note: there is still one state, and it does not change. Here, assuming that preferences are stationary.

Contextual Bandit Problems

- A contextual bandit is a tuple (A, S, R)
- A is a known set of k actions (or "arms")
- $\mathcal{S} = \mathbb{P}[s]$ is an unknown distribution over states (or "contexts")
- $\mathcal{R}_s^a(r) = \mathbb{P}[r \mid s, a]$ is an unknown probability distribution over rewards
- At each time t
  - Environment generates state $s_t \sim \mathcal{S}$
  - Agent selects action $a_t \in \mathcal{A}$
  - Environment generates reward $r_t \sim \mathcal{R}_{s_t}^{a_t}$
- The goal is to maximize cumulative reward $\sum_{\tau=1}^{t} r_\tau$

In essence, we need to condition or take into account the context. Need to encode it.

Note: the state is still not changing.