Exploration-exploitation, Regret, Bandits

How can we derive an **optimal** exploration strategy? In online learning, this is called "regret"
How to make exploration problem tractable?

- multi-armed bandits: formalized as POMDP identification. Policy learning is easy
- small&finite MDP: framed as a Bayesian model identification
- large&infinite MDP: optimal methods do not work here.
  Multi-armed bandits
  Multiple slot machines, each configured with an unknown reward probability.
  What is the best strategy to achieve highest long-term rewards given an infinite number of trials?

Let A be the number of arms. Denote each arm by a, where $a = 1, 2, .. N$. Each arm a would yield a random reward in $\{0, 1\}$ with an **unknown reward distribution** and **unknown mean** μ_a.
Let R_a, t be the reward received from playing arm a at time t, which is a Bernoulli random variable. The goal is to maximize the cumulative reward over a given number of plays:

$$maxR[\sum_{t=1}^{T} R_{a_t,t}]$$

where a_t is the arm played at time t.
Expected reward of arm a is denoted as $\mu_a = E[R_a]$
Total expected reward after T trials is $\sum_{t=1}^{T} \mu_{a_t}$
The optimal arm $a*$ has the highest expected reward $\mu* = max_a \mu_a$.
Regret
Regret is a measure of the performance of a strategy and is defined as the difference between the reward that would have been obtained by always playing the best arm and the reward actually obtained.
The simple regret measures the optimality gap at a time step:

$$SimpleRegret(t) = \mu^* - \mu_{A_t}$$

It's always non-negative. The cumulative regret after T trials is:

$$Regret(T) = T * \mu^* - \sum_{t=1}^{T} \mu_{A_t}$$

Non-negative and monotonically increasing.
There are 3 main choices for exploration-exploitation:

1. Greedy
2. $\epsilon-$ greedy

3. Upper Confidence Bound

Greedy

A greedy policy always chooses the arm with the highest estimated payoff:

$$a_t = argmax_a \tilde{R}_a(t)$$

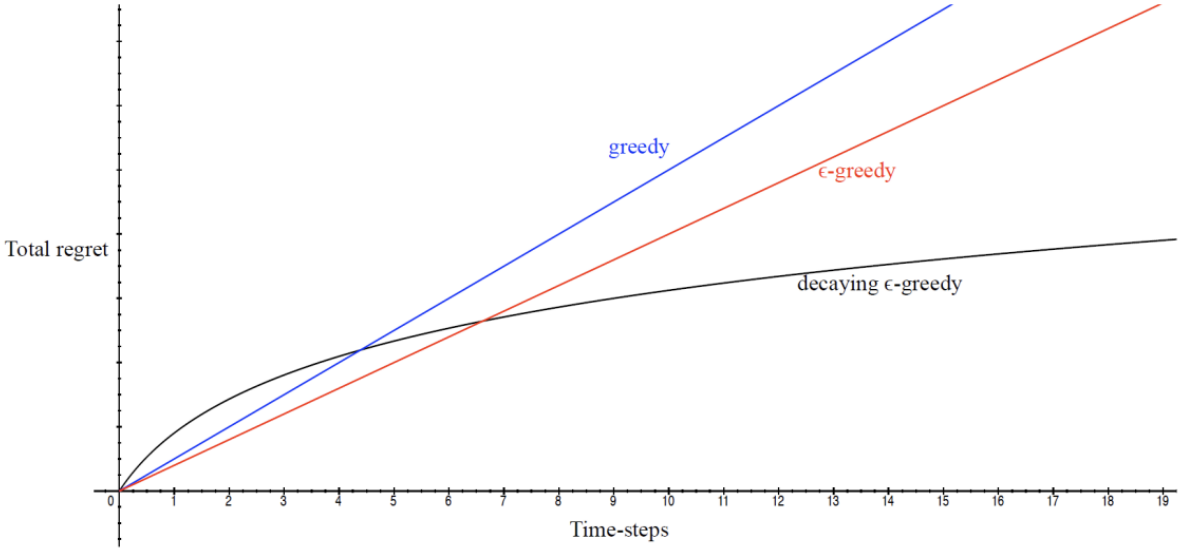
where a_t is the action at time t, and $\tilde{R}_a(t)$ is the estimated reward of arm a at time T:

$$\tilde{R}_a(t) = \frac{\#\text{ times algo pulled action and won up to time t}}{\#\text{ times algo pulled action up to time t)}}$$

This would fail if the initial unlucky trials which can happen due to some noise or bad initialization.

ϵ -greedy

With probability ϵ, choose an arm at random, and with probability $1-\epsilon$, choose the arm with the highest estimated reward
While it is better than greedy approach, the random exploration can be suboptimal.
One way to remedy this is to decay ϵ over time, but this also requires parameter tuning and hard to choose.



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

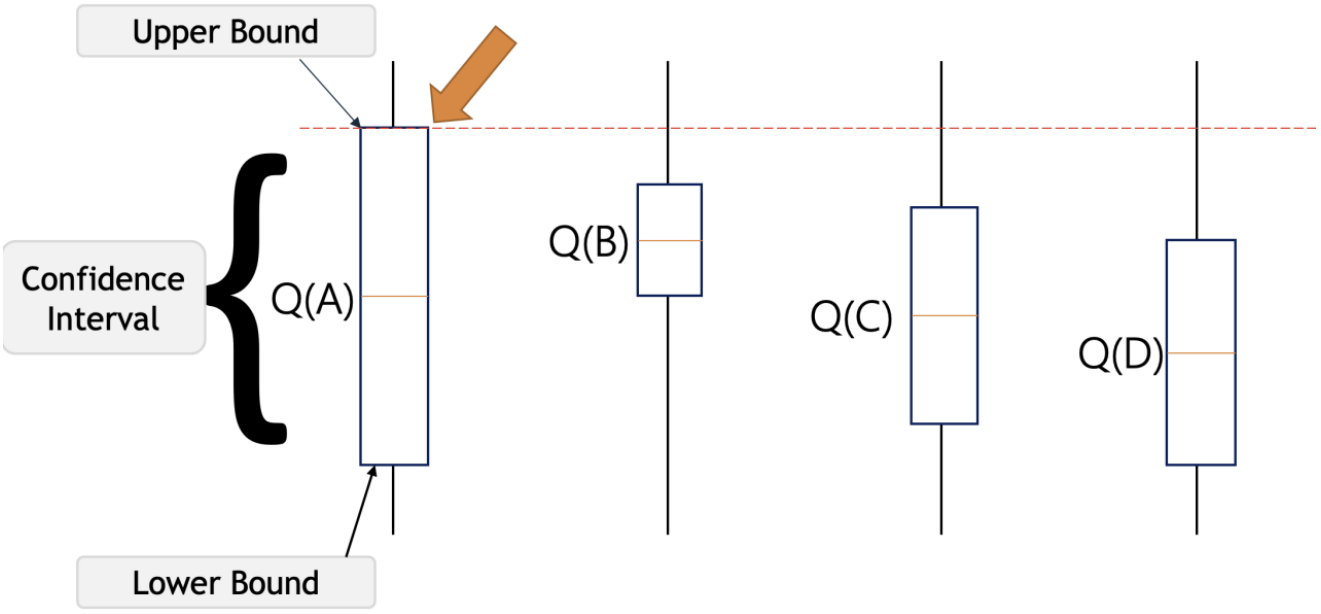
UCB

The more **uncertain** we are about an action-value, the more important it is to explore that action. It could turn out to be the best action.
The UCB method maintains an **optimistic estimate** for each arm's expected reward.
Suppose $\tilde{R}_a(t)$ is the estimated reward for arm a at time t. The UCB selects arms based on

a balance of estimated rewards and the uncertainty or variance in these estimates. The action selection is given by:

$$A_t = argmax_a(\tilde{R}_a(t) + \sqrt{\frac{2\ln t}{N_a(t)}})$$

where $N_a(t)$ is the number of times arm a has been selected up to time t.



Small $N_a(t) \rightarrow$ large bound (estimated value is uncertain)
Large $N_a(t) \rightarrow$ small bound (estimated value is certain/accurate)

Optimism

Consider the simple case where the random reward R_a for each arm a is a Bernoulli with parameter μ_a. Then $\tilde{R}_a(t)$ is the mean of $N_a(t)$ i.i.d Bernoulli random variable.
Let the upper confidence bound be:

$$\bar{R}_a(t) = \tilde{R}_a(t) + \sqrt{\frac{2\ln t}{N_a(t)}}$$

With high probability, we get an **upper confidence bound** for the true mean reward:

> With close to 1 probability, the UCB algorithm has regret $Regret(T) \leq C\sqrt{ATlog(AT)}$ where C is some constant

Bayesian UCB

In UCB, we did not consider any prior on the reward distribution. Prior knowledge on the distribution allows for a better bound. Example:
Bernoulli distribution on rewards: 0 or 1
Prior: uniform on $[0,1]$ $\forall a \in A$
Posterior: Beta distribution $Beta(\alpha_a, \beta_a)$ with initial parameters $\alpha_a = 1$ and $\beta_a = 1$ for each

action a.

Bayesian learning for 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:

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

Update the posterior:

$$\alpha_{a_t} \leftarrow \alpha_{a_t} + 1, r_t = 1$$

$$\beta_{a_t} \leftarrow \beta_{a_t} + 1, r_t = 0$$

Thompson Sampling

Select action a according to probability that a is the optimal action(given history of everything we observed so far):

Probability matching/posterior sampling

assume $r(a_i) \sim p_{\theta_i}(r_i)$

this defines a POMDP with $\mathbf{s} = [\theta_1, \ldots, \theta_n]$

belief state is $\hat{p}(\theta_1, \ldots, \theta_n)$

this is a *model* of our bandit

idea: sample $\theta_1, \ldots, \theta_n \sim \hat{p}(\theta_1, \ldots, \theta_n)$

pretend the model $\theta_1, \ldots, \theta_n$ is correct

take the optimal action

update the model

- This is called posterior sampling or Thompson sampling
- Harder to analyze theoretically
- Can work very well empirically

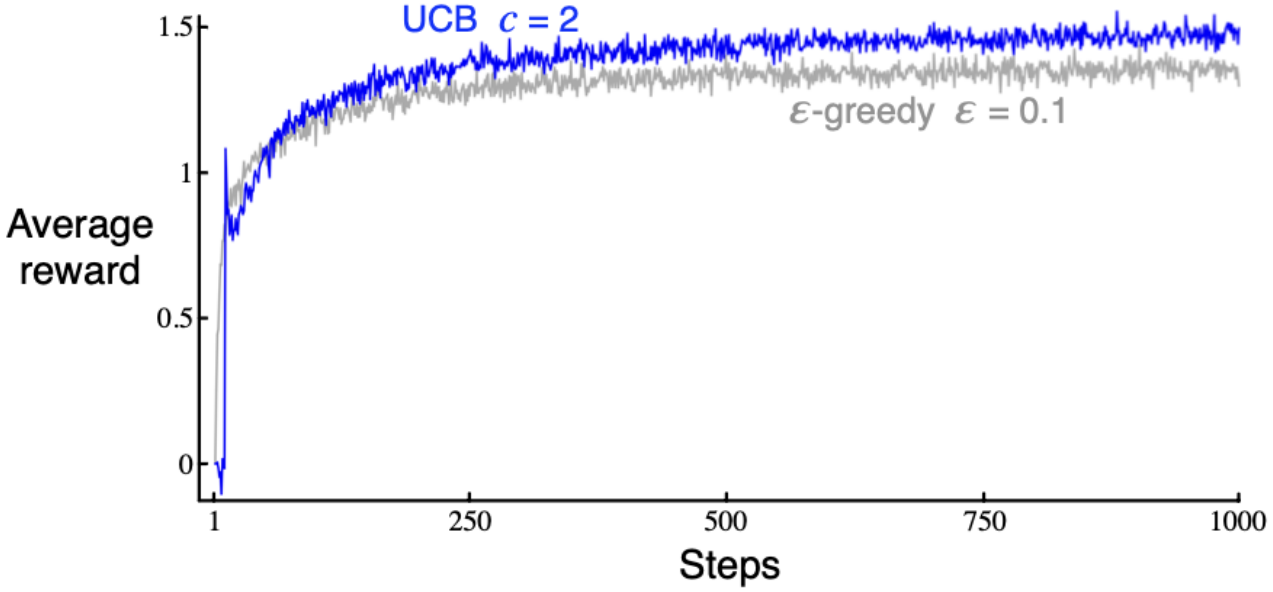
See: Chapelle & Li, "An Empirical Evaluation of Thompson Sampling."

Assume $R_a(t)$ follows a Beta distribution for the Bernoulli bandit.

Questions

Q1. Why does RMSE between utility/value estimate and true utility(win-probabilities) not correlate with the regret?
A1: RMSE and regret measure different aspects of performance; RMSE focuses on prediction accuracy, while the regret focuses on decision quality and cumulative performance.
Q2. Why the performance of epsilon-greedy is lower compared to UCB1 and probability matching? What do you need to change?
A2: Due to inefficient exploration. Adjusting ϵ dynamically or decaying it over time can improve its performance
Q3. What is the benefit of using probability matching compared to UCB1?
A3: Depending on the constant parameter selection UCB1, we might have a worse performance in overall regret minimization. Besides, Thompson Sampling can accommodate delayed feedback and is a probabilistic algorithm.
Q4 How easy is it to extend UCB into general RL problems?
A4 There are 2 difficulties that arise when applying UCB to RL problems:

- Dealing with non-stationary problems
- Dealing with large state spaces, particularly when using function approximation
  Q5 In this figure, results with UCB on the **10-armed** testbed are shown. The UCB algorithm shows a distinct spike in performance on the 11th step. Why is this? Note that for your answer to be fully satisfactory it must explain both why the reward increases on the 11th step and why it decreases on the subsequent steps. Hint: if c = 1, then the spike is less prominent.



A5: This is due to the fact that actions are essentially selected randomly in early steps. Large c suppresses the action to select the action greedily. This makes the rewards low in early steps.
The spike: once a high-reward is reached by some almost randomly selected actions, the average reward for that step increases significantly
Decrease on subsequent move: In the subsequent moves, C suppresses the agent keep selecting the high reward action, and therefore the reward is lowered. But because the Q for

high-reward action has already been updated, there is still a reasonable chance to select that action. Hence the reward is not as low as before the spike.