



A presentation for

CS 578

By

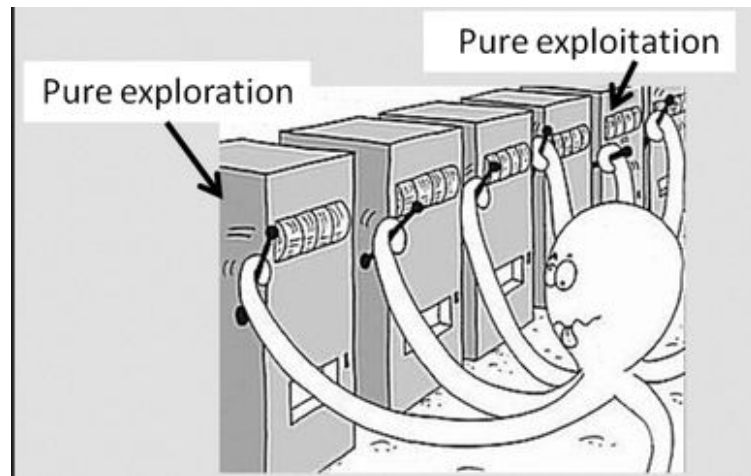
Shivam Bhat

Shyamvanshikumar Singh

Raghav Ram

MAB : Multi-armed bandit Problem

- A gambler at a row of slot machines has to decide which machines to play, how many times to play each machine and in which order to play them.
- When played, each machine provides a reward from a distribution specific to that machine.
- The objective is to maximize the sum of rewards earned through a sequence of lever pulls.



Should I keep pulling the best lever so far or should I explore a new lever?

MAB : Multi-armed bandit Problem

Formal Definition

- A set of k possible actions to choose from, a.k.a arms.
- Each action has an associated reward drawn from a probability distribution.
- Goal is to maximize the reward over T rounds.

“Bandit problems embody in essential form a conflict evident in all human action: choosing actions which yield immediate reward vs. choosing actions (e.g. acquiring information or preparing the ground) whose benefit will come only later” - P. Whittle (1980).

MAB : Use Case

Some real world exploration/exploitation examples:

Restaurant Selection

- Exploitation : Go to your favorite restaurant
- Exploration: Try a new restaurant

Oil Drilling

- Exploitation : Continue using existing well
- Exploration: Drill at a new location

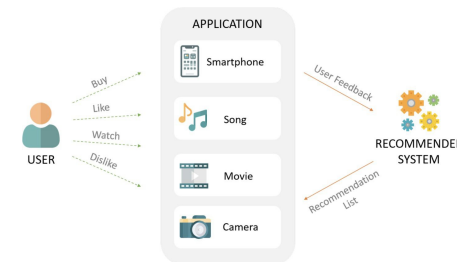
Online Banner Advertisements

- Exploitation: Show the most successful advert
- Exploration :Show a different advert



Action: Prescribing one of several possible treatments

Reward: Treatments effectiveness on health



Action: Recommending one of the several possible products available

Reward: User follows the recommendation

Stochastic Bandits

- We consider the basic model with IID rewards, called stochastic bandits.
- An algorithm has K possible actions to choose from, a.k.a. arms, and there are T rounds, for some known K and T .
- In each round, the algorithm chooses an arm and collects a reward for this arm.
- The algorithm's goal is to maximize its total reward over the T rounds

Problem protocol: Stochastic bandits

Parameters: K arms, T rounds (both known); reward distribution D_a for each arm a (unknown).

In each round $t \in [T]$:

- Algorithm picks some arm a_t
- Reward $r_t \in [0,1]$ is sampled independently from distribution D_{a_t}
- Algorithm collects reward r_t , and observes nothing else.

Remark

- We are primarily interested in the mean reward vector $\mu \in [0,1]^K$, where $\mu(a) = E[D(a)]$ is the mean reward of arm a .
- The set of all arms is A . The best mean reward is denoted $\mu^* := \max_{a \in A} \mu(a)$.
- An optimal arm, denoted by a^* , is an arm with $\mu(a) = \mu^*$; note that it is not necessarily unique.
- The difference $\Delta(a) := \mu^* - \mu(a)$ describes how bad arm a is compared to μ^* ; we call it the gap of arm a .

Regret

- We need a measurable quantity to differentiate between the performance of different algorithms.
- As our goal is to maximize our reward, we can compare the expected reward attained by an algorithm with the best possible reward of ; expected reward of always playing an optimal arm

Formally, we define the following quantity, called regret at round T :

$$R(T) = \mu^*T - \sum_{t=1}^T \mu(a_t)$$

As a_t is a random quantity, due to randomness in reward and the algorithm, $R(T)$ is also a r.v.

Therefore, we will mostly analyse $\mathbb{E}[R(T)]$.

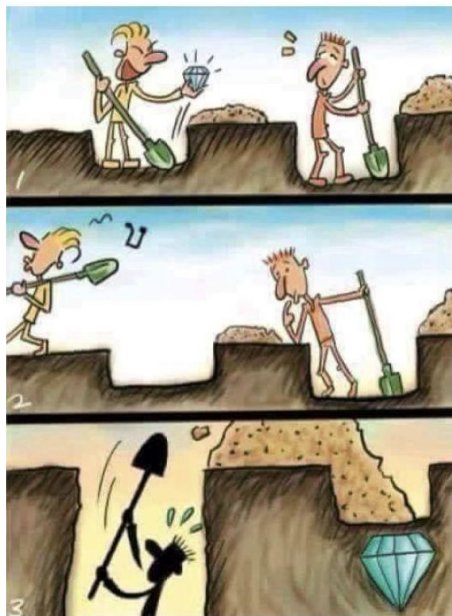
Theorem: Lower Bound of Regret

For a fixed time horizon T , number of arms K and any bandit algorithm, there exists a problem instance such that,

$$\mathbb{E}[R(T)] \geq \Omega(\sqrt{KT}).$$

This lower bound is “worst-case”, leaving open the possibility that a particular bandit algorithm has low regret for many/most other problem instances.

Exploitation vs Exploration

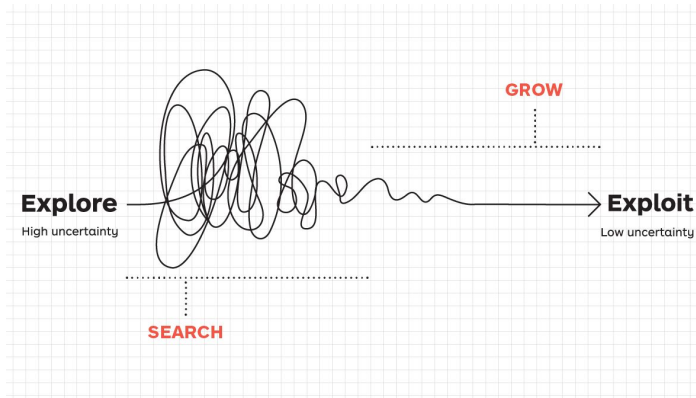
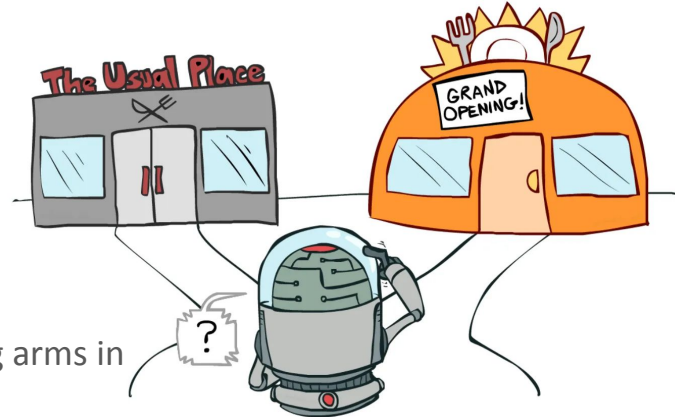


Should we exploit
the currently best
known arm?

Or should we look
for even better
arm?

Exploitation vs Exploration

- Exploitation and exploration are two competing strategies for selecting arms in the multi-armed bandit problem.
- Balancing these strategies is essential to achieve the best overall performance.
- Choosing the right mix of **exploration vs. exploitation** is a difficult balance to achieve. Exploit too much, and you might miss out on the real best machine. Explore too much, and you'll waste turns on subpar machines.



ϵ greedy

A pure greedy strategy poses a high risk of sticking with a sub-optimal arm, preventing the discovery of the best arm.

To address this issue, Epsilon-Greedy introduces exploration:

1. Actions are usually chosen based on their estimated rewards, favoring the one with the highest estimate.
2. Yet, at each time step, there's a chance to choose a random action from all possibilities, determined by a probability ' ϵ ' (Epsilon).
3. In this way exploration is added to the standard Greedy algorithm. Over time every action will be sampled repeatedly to give an increasingly accurate estimate of its true reward value.

ϵ greedy

Formal Definition

- Choose a random machine to pull with probability = ϵ
- Choose the best machine to pull with probability = $1-\epsilon$

Here, the algorithm defines the **“best” machine** very simply -- it is just the one with the highest experimental mean, where the experimental mean is calculated as the sum of the rewards from that machine divided by the number of times that machine has been pulled.

Definition: *Experimental Mean*

The experimental mean of machine i after T turns is

$$\frac{\sum_{t=1}^T r_{i,t}}{p_{i,T}}$$

where $r_{i,t}$ is the reward given by machine i at timestep t and $p_{i,t}$ is the number of times that machine i has been pulled across T total turns.

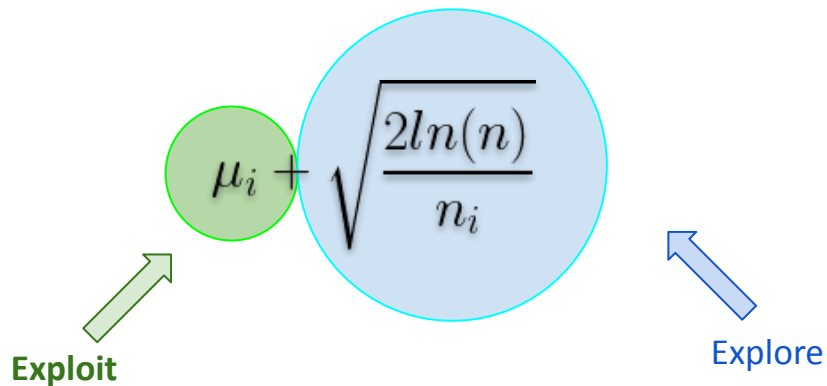
ϵ greedy

Pseudo Code

```
for t = 1...numTurns:
    flip a coin with probability epsilon of heads and 1 - epsilon of tails
    if heads:
        pull a random arm
    else:
        calculate the arm with the best experimental mean
        pull the arm with the best experimental mean
```

UCB

- UCB algorithm incorporates exploration and exploitation
- UCB1 algorithm is described as “**optimism in the face of uncertainty.**”
- The UCB algorithm always picks the arm with the highest reward UCB as represented by the equation below.



The diagram illustrates the UCB1 equation, which is used to select the arm with the highest reward UCB. The equation is represented by a large blue circle containing the formula:

$$\mu_i + \sqrt{\frac{2\ln(n)}{n_i}}$$

The term μ_i is highlighted in a green circle, and a green arrow labeled "Exploit" points to it. The term $\sqrt{\frac{2\ln(n)}{n_i}}$ is highlighted in a blue circle, and a blue arrow labeled "Explore" points to it.

Confidence Bounds

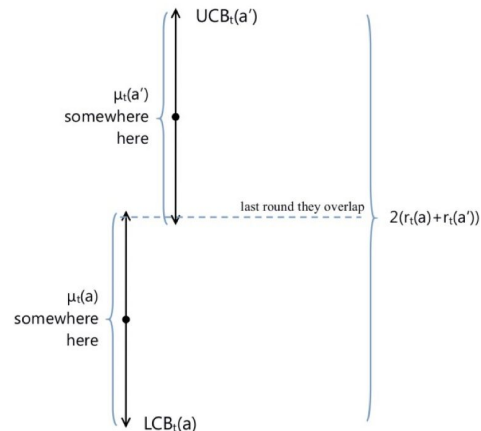
Let $n_t(a)$ be number of times arm a is chosen in first t rounds, and $\bar{\mu}_t(a)$ be average reward collected in these rounds. By Hoeffding Inequality we have:

$$Pr[|\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)] \geq 1 - \frac{2}{T^4}, \text{ where } r_t(a) = \sqrt{2\log(T)/n_t(a)} \quad (6)$$

The event ε will be clean event and we define the upper and lower confidence bounds as follows:

$$UCB_t(a) = \bar{\mu}_t(a) + r_t(a)$$

$$LCB_t(a) = \bar{\mu}_t(a) - r_t(a)$$



UCB1

Formal Definition

$$\mu_i + \sqrt{\frac{2\ln(n)}{n_i}}$$

- μ_i represents the current reward return average of arm i at the current round
- n represents the number of trials passed,
- n_i represents the number of pulls given to arm i in the playthrough history.
- The more times the specific arm has been engaged before in the past, the greater the confidence boundary reduces towards the point estimate.

UCB1

Pseudo Code

```
first k turns: initialize experimental means by pulling each arm once
for t = k+1...numTurns:
    for i = 1...k:
        calculate  $a(i, t) = 2 * \log(t / p(i, t))$ 
    pull arm i that maximizes experimental mean +  $a(i, t)$ 
```

UCB1: Optimism under uncertainty

- Here optimism is based in the assumption that each arm can be as best as it can.
- The expected reward of each arm will be compared on the basis of its upper confidence bound (UCB) and in each round we will select an arm with maximum UCB.

```
Try each arm once  
for each round  $t = 1, \dots, T$  do  
    pick arm some  $a$  which maximizes  $UCB_t(a)$ .  
end for
```

Why UCB based selection works?

$$UCB_t(a) = \bar{\mu}_t(a) + r_t(a) \quad \text{where } r_t(a) = \sqrt{2\log(T)/n_t(a)}$$

- UCB value of an arm can be large if average reward confidence radius $r_t(a)$ is large
- Larger $\bar{\mu}_t(a)$ is likely indication of arm having high reward, whereas a larger $r_t(a)$ is because of smaller $n_t(a)$ i.e. the arm has not been explored much.
- Both reasons are appealing to select a particular arm.
- $\bar{\mu}_t(a)$ provides a reason for exploitation and $r_t(a)$ provides a reason for exploration. Both of them are summed up in UCB expression.

UCB regret analysis

For analysis of regret lets again focus only on clean event. Let a^* be the optimal arm and a_t be the arm chosen by the algorithm in round t . According to the algorithm, $UCB_t(a_t) \geq UCB_t(a^*)$

For a clean event $\mu(a_t) \geq LCB_t(a_t)$ i.e. $\mu(a_t) \geq \bar{\mu}_t(a_t) - r_t(a)$. By rearranging we get $\mu(a_t) + r_t(a) \geq \bar{\mu}_t(a_t)$. Also $UCB_t(a^*) \geq \mu(a^*)$

$$\begin{aligned}\mu(a_t) + 2r_t(a_t) &\geq \bar{\mu}_t(a_t) + r_t(a) \\ &\geq UCB_t(a_t) \\ &\geq UCB_t(a^*) \\ &\geq \mu(a^*)\end{aligned}$$

Therefore

$$\Delta(a_t) := \mu(a^*) - \mu(a_t) \leq 2r_t(a_t) = 2\sqrt{2\log(T)/n_t(a)}$$

UCB regret analysis

Therefore each arms contribution to regret is

$$R(t; a) = n_t(a) \cdot \Delta(a) \leq n_t(a) \cdot O(\sqrt{\log(T)/n_t(a)}) = O(\sqrt{n_t(a)\log(T)})$$

Adding regret of all arms, we get

$$R(t) = \sum_{a \in \mathcal{A}} R(t; a) \leq O(\sqrt{\log T}) \sum_{a \in \mathcal{A}} \sqrt{n_t(a)} \quad (9)$$

As, $\sum_{a \in \mathcal{A}} n_t(a) = t$ and $f(x) = \sqrt{x}$ is a concave function, we use Jensen's inequality to get:

$$\frac{1}{K} \sum_{a \in \mathcal{A}} \sqrt{n_t(a)} \leq \sqrt{\frac{1}{K} \sum_{a \in \mathcal{A}} n_t(a)} = \sqrt{\frac{t}{K}}$$

Using this eq.9 we get

$$R(t) \leq O(\sqrt{Kt\log T}) \quad (10)$$

$t^{1/2}$ is a better bound compared to $T^{2/3}$

Bayesian Bandits and Thompson sampling

Bayesian Bandits

A MAB problem instance \mathcal{I} is defined by the parameter vector μ , which in turn define the reward distribution \mathcal{D}_a .

Previously, we assumed μ to be fixed, but now we assume a distribution \mathbb{P} called a prior distribution and $\mathcal{I} \sim \mathbb{P}$

Here the goal is optimize the Bayesian regret given as

$$\text{BR}(T) := \mathbb{E}_{\mathcal{I} \sim \mathbb{P}} [\mathbb{E}[R(T) | \mathcal{I}]] \quad (11)$$

Problem simplifications and assumptions for analysis:

- We will focus on Bernoulli bandits with mean vector
- The realized rewards can only take finitely many different values, and the prior \mathbb{P} has a finite support, denoted by \mathcal{F}
- the best arm a^* is unique for each mean reward vector in the support of \mathbb{P} .

Bayesian update

We start with defining t -history, H_t as a sequence of action-reward pair for first t -rounds

$$H_t = ((a_1, r_1), (a_2, r_2), \dots, (a_t, r_t)) \in (\mathcal{A} \times \mathbb{R}) \quad (12)$$

For a fixed sequence $H \in (\mathcal{A} \times \mathbb{R})$ is said to be feasible if $Pr[H_t = H] > 0$ for some bandit algorithm and algorithm is called H -consistent.

Let \mathcal{H}_t be set of all feasible t -histories. For Bernoulli rewards $\mathcal{H}_t \in (\mathcal{A} \times \mathbb{R})^t$. Using this we define our posterior distribution as

$$\mathbb{P}_H(\mathcal{M}) := Pr[\mu \in \mathcal{M} | H_t = H], \quad \forall \mathcal{M} \subset [0, 1]^K \quad (13)$$

The process of deriving \mathbb{P}_H is called Bayesian update of \mathbb{P} given H

Two important lemmas:

- The posterior does not depend on the algorithm. It only depends on the t -history.
- The posterior can be used as a prior for subsequent Bayesian updates

Thompson sampling

```
for each round  $t = 1, 2, \dots$  do  
  Observe  $H_{t-1} = H$ , for some feasible  $(t - 1)$ -history  $H$ ;  
  Sample mean reward vector  $\mu_t$  from the posterior distribution  $\mathbb{P}_H$ ;  
  Choose the best arm  $\tilde{a}_t$  according to  $\mu_t$ .  
end
```

Algorithm	Regret
ϵ -greedy	$O(T^{2/3}(K \log(T))^{1/3})$
UCB	$O(\sqrt{KT \log T})$
Thompson sampling	$O(\sqrt{KT \log T})$

Computational Cost

$$\mathbb{P}_H(\tilde{\mu}) = \frac{\Pr[\mu = \tilde{\mu} \text{ and } H_t = H]}{\Pr(H_t = H)} = \frac{\mathbb{P}(\tilde{\mu}) \cdot \Pr[H_t = H \mid \mu = \tilde{\mu}]}{\sum_{\tilde{\mu} \in \mathcal{F}} \mathbb{P}(\tilde{\mu}) \cdot \Pr[H_t = H \mid \mu = \tilde{\mu}]}, \quad \forall \tilde{\mu} \in \mathcal{F}.$$

Using sequential updates, where after each round t we treat posterior as our new prior, we can compute $\Pr[H_t = H \mid \mu = \tilde{\mu}] = O(1)$. Thus $\mathbb{P}(H_t = H)$ can be computed in $O(|\mathcal{F}|)$.

The support \mathcal{F} can be exponentially large for a MAB problem with many hands. One solution is to consider independent priors i.e. each arm has a prior independent of each other.

$$\mathbb{P}_{H'}^a(x) = \Pr_{\mu(a) \sim \mathbb{P}_H^a} [\mu(a) = x \mid (a_t, r_t) = (a, r)] = \frac{\mathbb{P}_H^a(x) \cdot \mathcal{D}_x(r)}{\sum_{x \in \mathcal{F}_a} \mathbb{P}_H^a(x) \cdot \mathcal{D}_x(r)}, \quad \forall x \in \mathcal{F}_a,$$

where \mathcal{F}_a is the support of $\mu(a)$. Thus, the new posterior $\mathbb{P}_{H'}^a$ can be computed in time $O(|\mathcal{F}_a|)$. This is an exponential speed-up compared $|\mathcal{F}|$ (in a typical case when $|\mathcal{F}| \approx \prod_a |\mathcal{F}_a|$).

Special Case

Some special cases admit much faster computation of the posterior \mathbb{P}_H^a and much faster sampling thereafter.

Beta-Bernoulli: A Beta-Bernoulli conjugate pair is a combination of Bernoulli rewards and a prior $\mathbb{P} = \text{Beta}(\alpha_o, \beta_o)$ for some parameter (α_o, β_o) . The posterior \mathbb{P}_H is simply $\text{Beta}(\alpha_o + \text{REW}_H, \beta_o + t)$, where REW_H is the total reward in H and t is the time period.

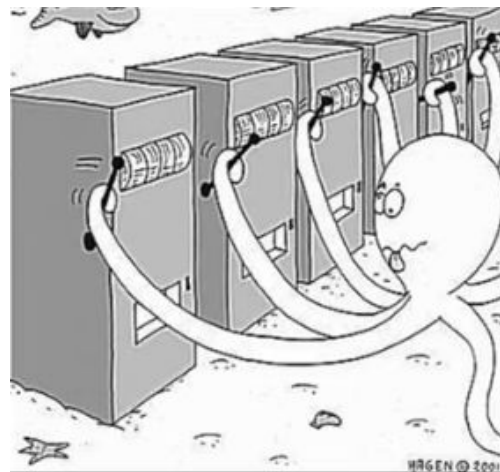
Gaussian: A Gaussian conjugate pair is a combination of a Gaussian reward distribution and a Gaussian prior \mathbb{P} . Assuming the prior is $N(\mu_o, \sigma_o)$ and the reward distribution is of unit variance σ and mean parameter μ , the posterior $\mathbb{P}_H = N\left(\frac{1}{\frac{1}{\sigma_o^2} + t} \left(\frac{\mu_o}{\sigma_o^2} + \text{REW}_H\right), \left(\frac{1}{\sigma_o^2} + t\right)^{-1}\right)$

Experiments

ϵ greedy

Simulating Multi Armed Bandit Problem using ϵ greedy

- Code modeling the MAB problem:
 - Create 5 arms, four of which have average reward of 0.1, and the best has average reward of 0.9
 - We implement a Bernoulli distribution for each arm
 - Create 5000 independent simulations for each epsilon value for a total of 5 epsilon values ranging from 0.1 to 0.5



Simulation Results

We choose means: $[0.1, 0.1, 0.1, 0.1, 0.9]$

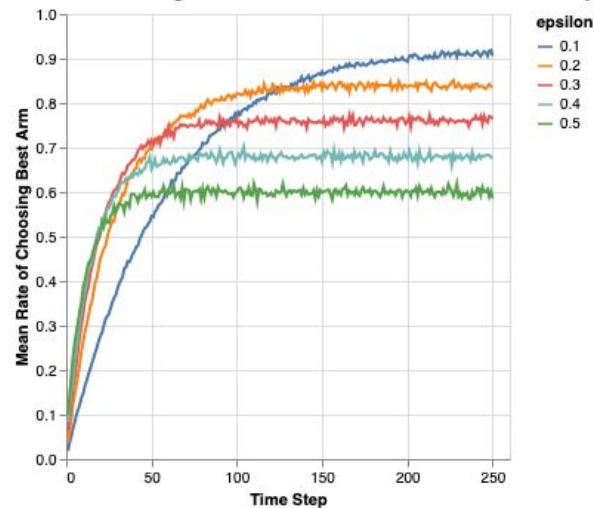
Best-arm-discovery rate is not linear

Asymptotic performance

Higher the value of epsilon, the lower its asymptotic performance

Note the rate of convergence. The higher the rate of exploration, the earlier the algorithm discovers the best arm. Green is higher initially before blue takes over.

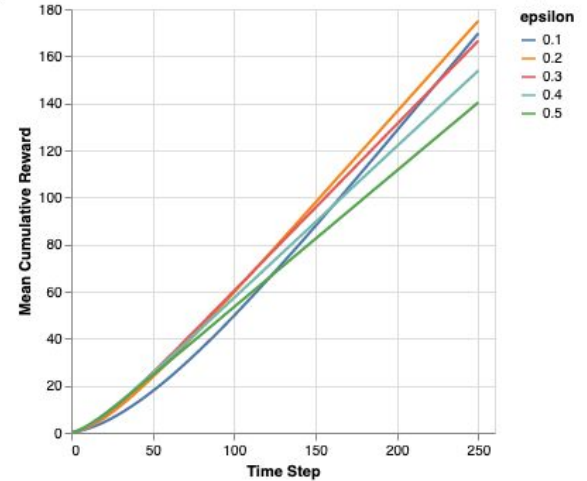
Eps-Greedy: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = $[4 \times 0.1, 1 \times 0.9]$



Simulation Results

- Looking at cumulative reward ,neither 0.1 nor 0.5 epsilon values are the best performers
- The best performer is instead 0.2

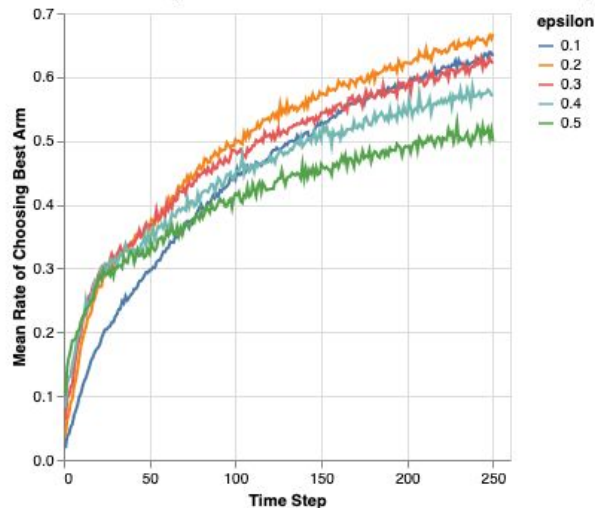
Eps-Greedy: Mean Cumulative Reward from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



Simulation of Arms with relatively smaller differences in Means

- We choose means: $[0.8, 0.8, 0.8, 0.8, 0.9]$
- Arms are relatively closer
- closer difference in reward means results in algorithm taking much longer to approach the asymptotic limit
- Time = 250 was insufficient for the algorithm to discover the best arm as compared to the experiment done in the previous section
- $\text{arm}(0.1)$ takes much longer to discover the best arm as shown by slow start by blue curve

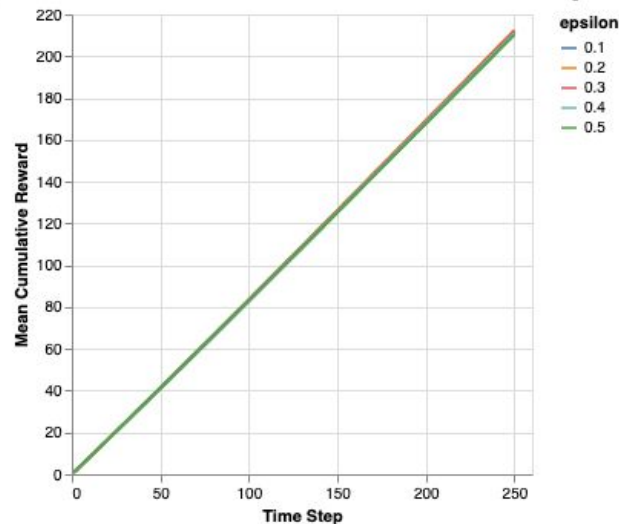
Eps-Greedy: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = $[4 \times 0.1, 1 \times 0.9]$



Simulation of Arms with relatively smaller differences in Means

- Cumulative rewards for all epsilon values are much closer
- This is because :
 - The reward means of all arms are quite close
 - Within the time=250, the algorithms have not discovered the best arm yet
- Since cumulative rewards are close we look at regret

Eps-Greedy: Mean Cumulative Reward from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]

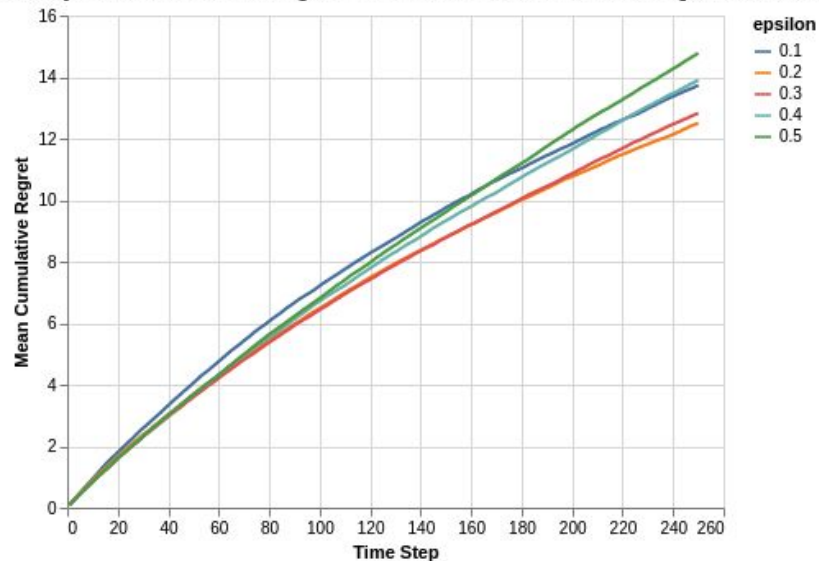


Simulation of Arms with relatively smaller differences in Means

Regret

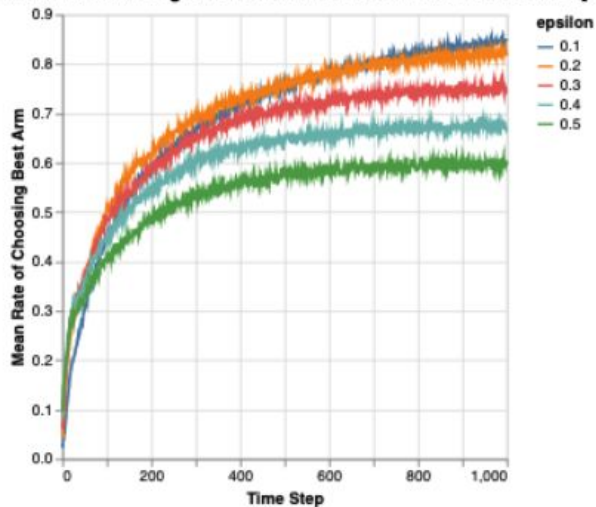
- epsilon value of 0.2 is the best with least regret
- reduction in progressive regret accumulation

Eps-Greedy: Mean Cumulative Regret from 5000 Simulations. 5 Arms = [4 x 0.8, 1 x 0.9]

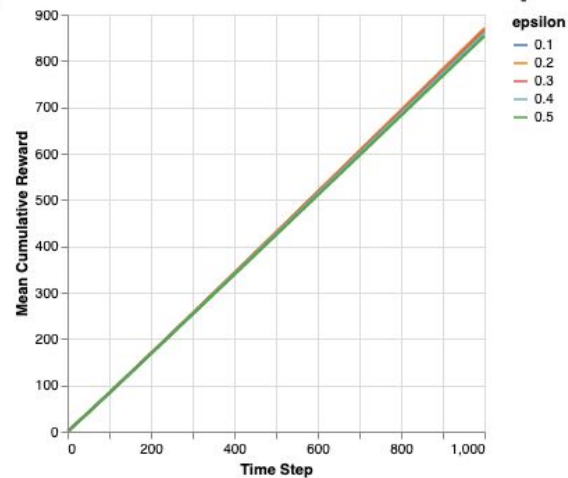


At $t=1000$

Eps-Greedy: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



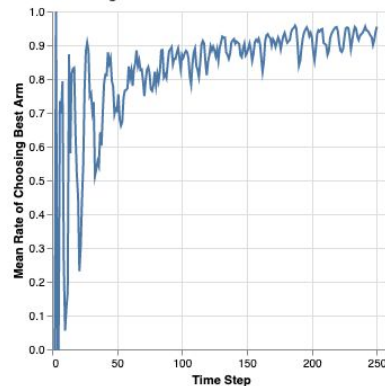
Eps-Greedy: Mean Cumulative Reward from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



UCB

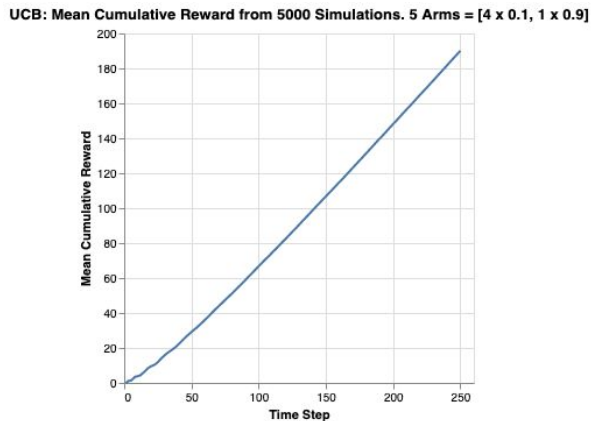
- The UCB algorithm has extreme fluctuations in its rate of choosing the best arm in the early phases of the experiment
- This can be explained by the emphasis of exploration amongst all arms since the UCB components for all arms are much bigger at the start.
- As the trial progresses, the UCB components becomes much smaller for all arms, and the UCB function representation of each arm converges towards the average reward mean of each arm

UCB: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



UCB

- We also observe some form of curvature in the early phases of the trial, which can be corroborated by the extreme fluctuations we saw in the rate of choosing best arms.
- Likewise, when the experiment progresses, the algorithm can distinguish the best arm, and picks it with higher frequency, and the cumulative reward plot has a straight line gradient

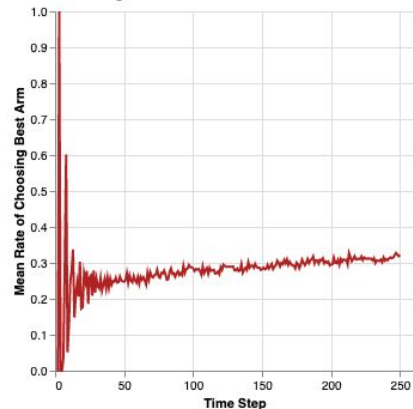


UCB - with close means

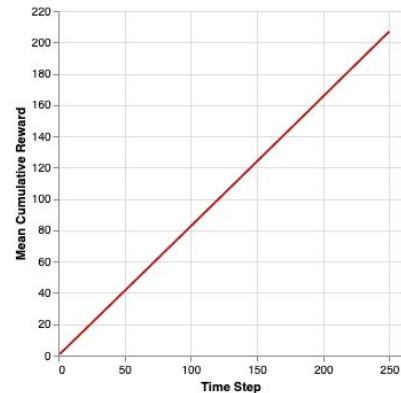
Analysis of the situation where the arms are relatively closer.

- Based on a reduced difference between the reward returns of all arms, we observe a big deterioration in the performance of the UCB algorithm. The rate of choosing the best arm now approaches 0.33
- Reduced difference in reward function makes it harder to determine which is the best arm.
- In such case for Epsilon Greedy algorithm, the rate of choosing the best arm is actually higher as represented by the ranges of 0.5 to 0.7
- Epsilon-greedy seems to be better suited for multi-armed based situations where the difference in means are much smaller as compared to UCB

UCB: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



UCB: Mean Cumulative Reward from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



UCB

Regret

UCB cumulative reward obtain = 210

Reward on choosing the best arm = $0.9 * 250 = 225$

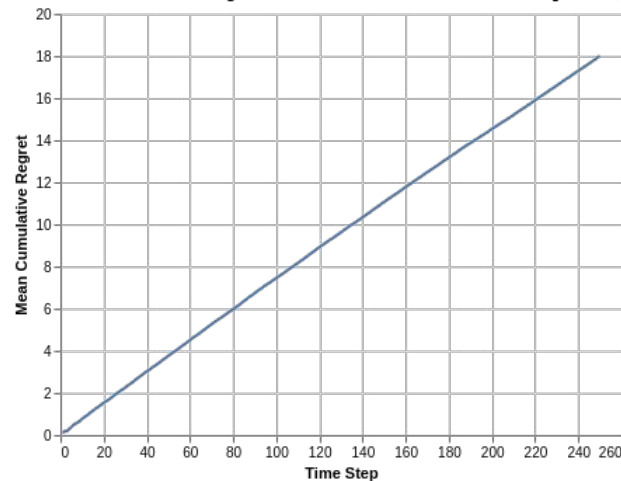
UCB regret = $225 - 210 = 15$ (~ 6.6%)

It's a still a significant amount of regret

Epsilon-greedy regret = 12.5

Straight line implying it will accumulate more regret with a longer time horizon.

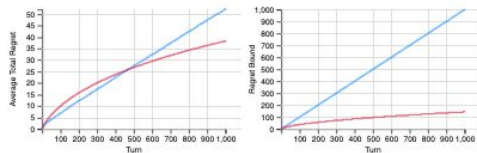
UCB: Mean Cumulative Regret from 5000 Simulations. 5 Arms = [4 x 0.8, 1 x 0.9]



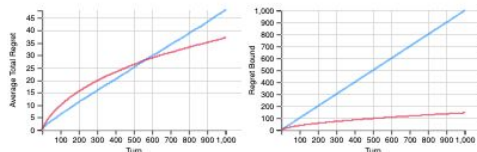
UCB vs ϵ -Greedy

In practice, we see that UCB1 tends to outperform epsilon greedy when the number of arms is low and the standard deviation is relatively high, but its performance worsens as the number of arms increases.

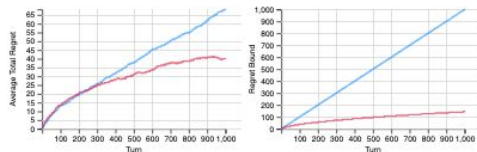
$k = 3, \sigma = 0.01$:



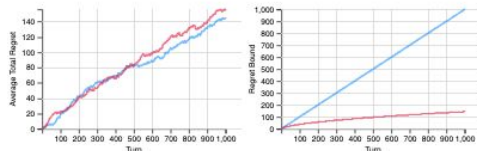
$k = 3, \sigma = 0.1$:



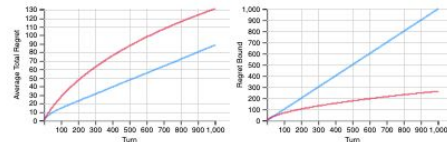
$k = 3, \sigma = 1$:



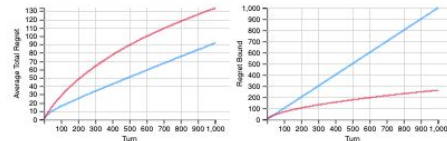
$k = 3, \sigma = 5$:



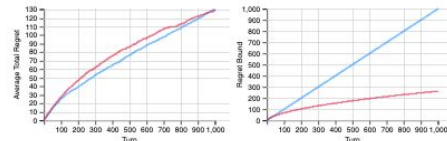
$k = 10, \sigma = 0.01$:



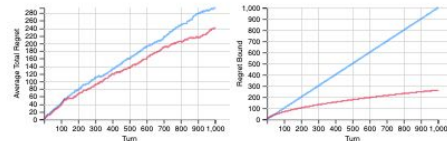
$k = 10, \sigma = 0.1$:



$k = 10, \sigma = 1$:



$k = 10, \sigma = 5$:



K (# arms)=3
 $\sigma=0.01$

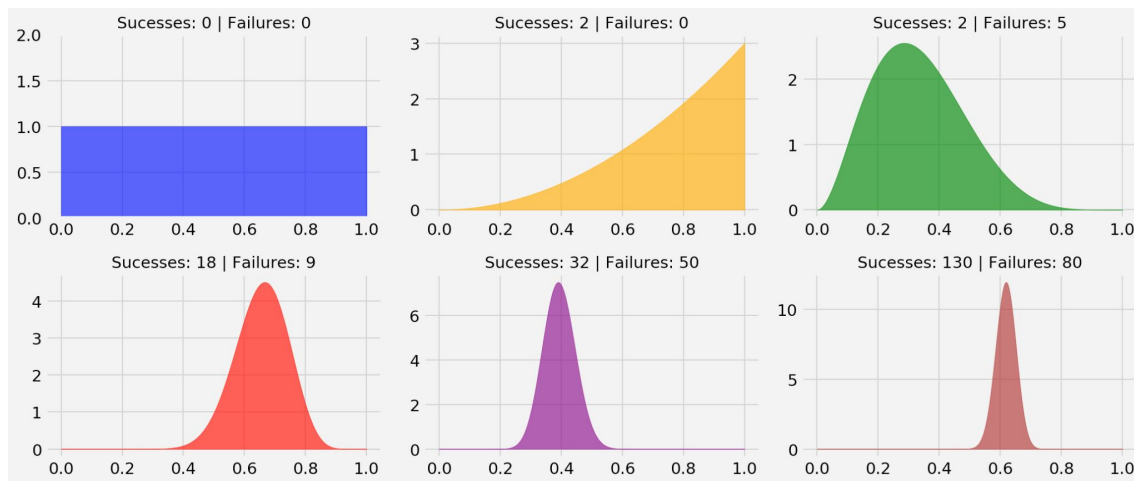
K (# arms)=10
 $\sigma=0.01$

■ Epsilon Greedy ■ UCB1

■ Epsilon Greedy ■ UCB1

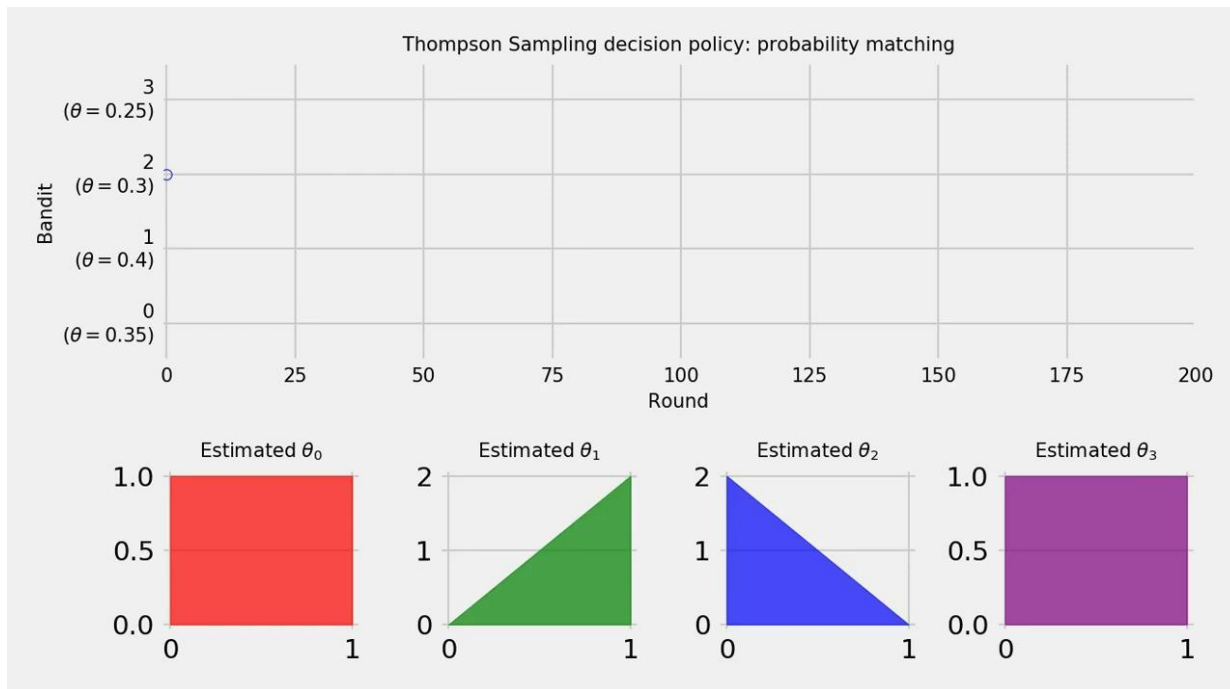
Thompson Sampling

- Up until now, all of the methods we've seen for tackling the Bandit Problem have selected their actions based on the current averages of the rewards received from those actions
- Thompson Sampling (Bayesian Bandits algorithm) takes a slightly different approach
- Instead of just refining an estimate of the mean reward it extends this, to instead build up a probability model from the obtained rewards, and then samples from this to choose an action

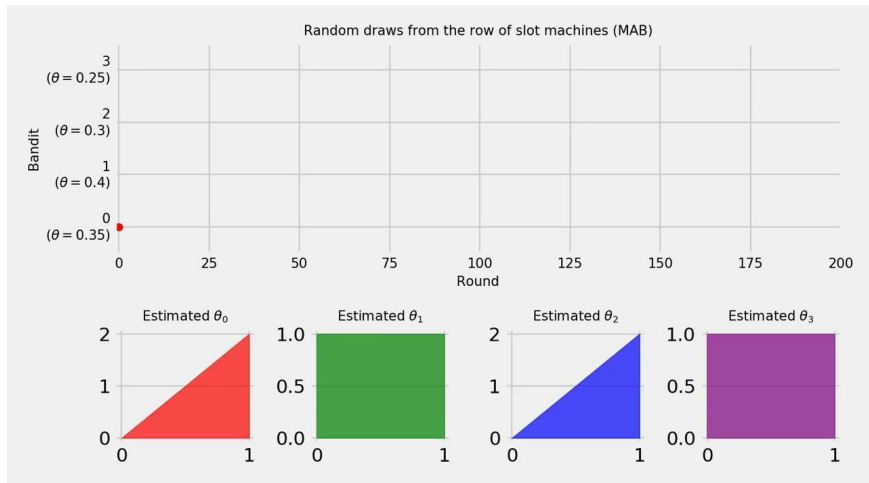


Thompson Sampling

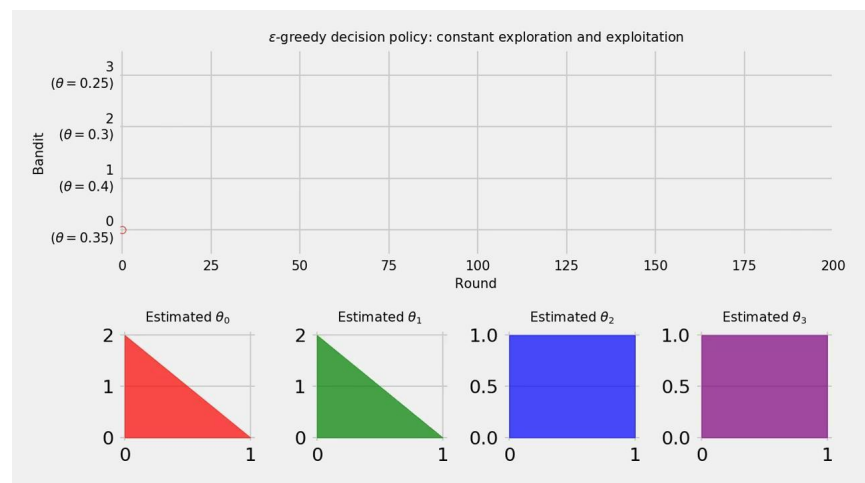
- Estimated probabilities change as we play.
- We can use this information to our benefit, in order to balance the uncertainty around our beliefs (exploration) with our objective of maximizing the cumulative reward (exploitation)



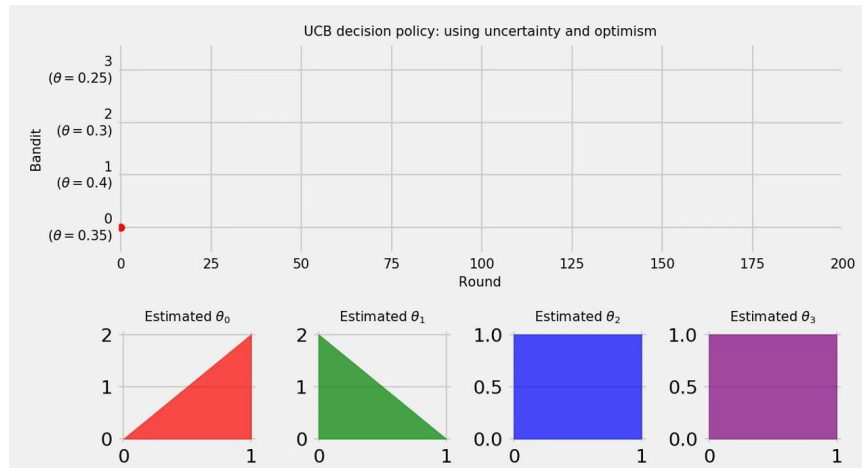
Random



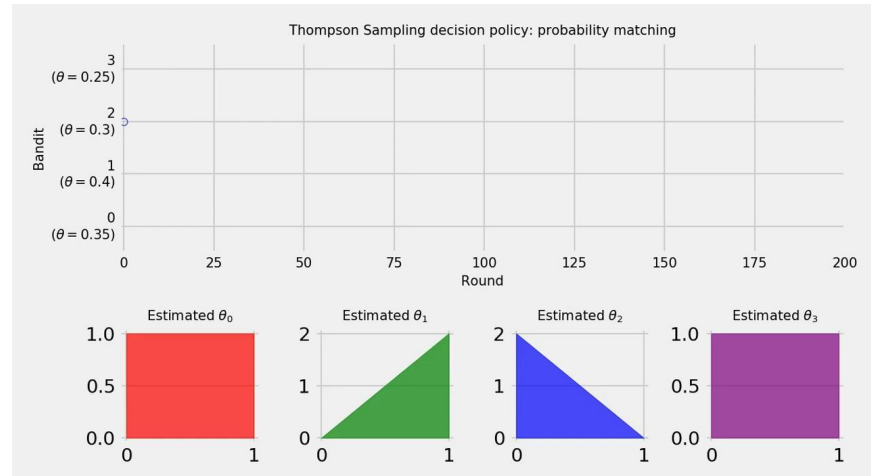
Epi-greedy



UCB



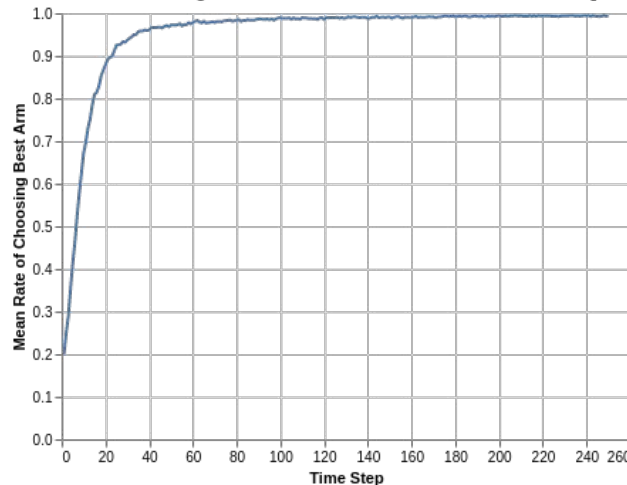
TS



Thompson Sampling

- Relatively quick convergence to the choice of best arm.
- The curve is also smoother than UCB
- Within 40 trials, the average rate of choosing the best arm is around 95%.
- This is extremely impressive compared to the other algorithms we have seen so far
- At the start, all arms are perceived equally since they all have the same priors.
- Hence, the rate of choosing the best arm always starts from 20%, which is a random chance of choosing the best arm out of 5 arms

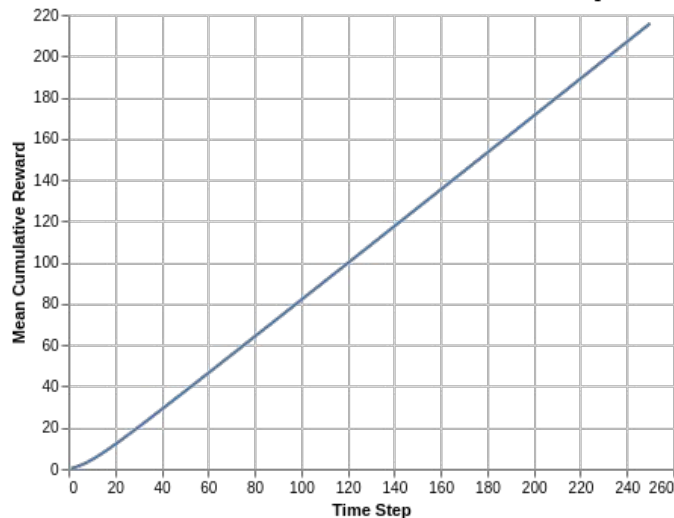
TS: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = [4 x 0.1, 1 x 0.9]



Thompson Sampling

- Since TS manages to identify best arm early on, it starts accumulating rewards quickly.
- TS reaches about 215 cumulative points on average $t=250$
- TS outperforms almost all the other algorithms such as Epsilon Greedy, and UCB1
- This is extremely impressive compared to the other algorithms we have seen so far
- At the start, all arms are perceived equally since they all have the same priors.
- Hence, the rate of choosing the best arm always starts from 20%, which is a random chance of choosing the best arm out of 5 arms

TS: Mean Cumulative Reward from 5000 Simulations. 5 Arms = $[4 \times 0.1, 1 \times 0.9]$

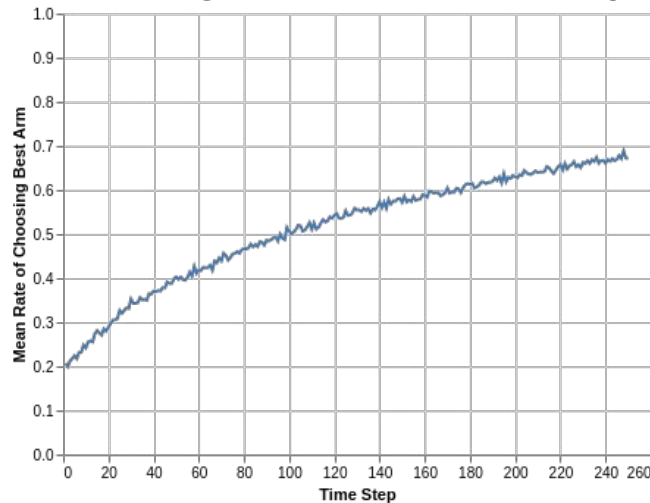


Thompson Sampling - With closer mean

Extend the analysis to a situation where the arms are relatively closer

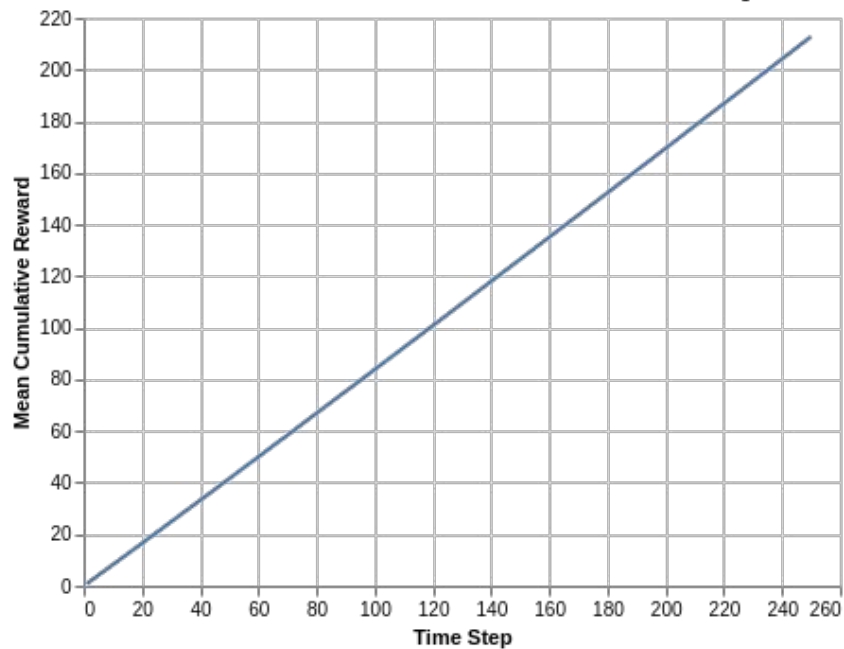
With TS, the rate of choosing the best arm now approaches 0.68, which is the best result so far (even compared to Eps-Greedy $\epsilon = 0.2$)

TS: Mean Rate of Choosing Best Arm from 5000 Simulations. 5 Arms = $[4 \times 0.8, 1 \times 0.9]$



Thompson Sampling - With closer mean

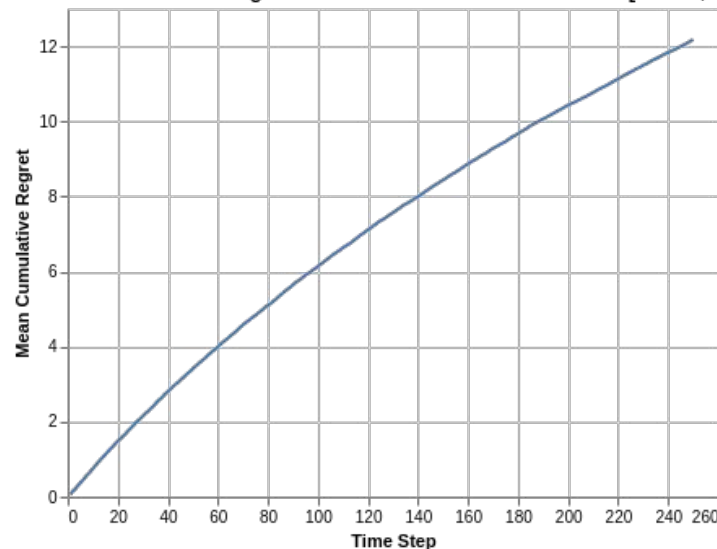
TS: Mean Cumulative Reward from 5000 Simulations. 5 Arms = [4 x 0.8, 1 x 0.9]



Thompson Sampling - With closer mean

Thompson Sampling is the best performer with a cumulative regret of 12.1. This is much more superior compared to UCB and epi-greedy

TS: Mean Cumulative Regret from 5000 Simulations. 5 Arms = [4 x 0.8, 1 x 0.9]



End!

Future Work

Besides the two algorithms that we discussed, other strategies for solving the MAB problem exist:

- [Boltzmann exploration \(SoftMax\)](#): similar to the algorithms discussed above, but each arm is pulled with a probability that is proportional to the average reward given by that arm.
- [Adaptive epsilon-greedy based on value differences \(VDBE\)](#): an extension of the epsilon greedy algorithm that uses a state dependent exploration probability, therefore taking the responsibility of balancing exploration vs. exploitation out of the hands of the implementer.
- [LinUCB](#): an algorithm aimed at solving a variant of the MAB problem called the *contextual* multi-armed bandit problem. In the contextual version of the problem, each iteration is associated with a feature vector that can help the algorithm decide which arm to pull.