Tutorial on Bandit

Subhojyoti Mukherjee

IIT Madras

February 3, 2017

Overview

Introduction

- The bandit problem is a sequential decision making process where at each timestep we have to choose one action or arm from a set of arms.

Introduction

- The bandit problem is a sequential decision making process where at each timestep we have to choose one action or arm from a set of arms.
- After say pulling each arm once we are presented with an *exploration-exploitation* problem, that is whether to continue to pull the arm for which we have observed the highest estimated reward till now(exploitation) or to explore a new arm(exploration).

Introduction

- The bandit problem is a sequential decision making process where at each timestep we have to choose one action or arm from a set of arms.
- After say pulling each arm once we are presented with an *exploration-exploitation* problem, that is whether to continue to pull the arm for which we have observed the highest estimated reward till now(exploitation) or to explore a new arm(exploration).
- If we become too greedy and always exploit we may miss the chance of actually finding the optimal arm and get stuck with a sub-optimal arm.

Why study bandits at all?

- We all know of ϵ-greedy algorithm, we can simply stick to it.

Why study bandits at all?

- We all know of ϵ-greedy algorithm, we can simply stick to it.
- But ϵ-greedy only gives us an asymptotic guarantee. There is no guarantee that in a highly regressive environment how ϵ-greedy will behave. Can we be better in our search?

Why study bandits at all?

- We all know of ϵ-greedy algorithm, we can simply stick to it.
- But ϵ-greedy only gives us an asymptotic guarantee. There is no guarantee that in a highly regressive environment how ϵ-greedy will behave. Can we be better in our search?
- Bandits allows us to study this behavior in a more formal way giving us strict guarantees regarding the performance of our algorithm.

Why study bandits at all?

- We all know of ϵ-greedy algorithm, we can simply stick to it.
- But ϵ-greedy only gives us an asymptotic guarantee. There is no guarantee that in a highly regressive environment how ϵ-greedy will behave. Can we be better in our search?
- Bandits allows us to study this behavior in a more formal way giving us strict guarantees regarding the performance of our algorithm.
- They form the linking pieces of a larger problem.

Why study bandits at all?

- We all know of ϵ-greedy algorithm, we can simply stick to it.
- But ϵ-greedy only gives us an asymptotic guarantee. There is no guarantee that in a highly regressive environment how ϵ-greedy will behave. Can we be better in our search?
- Bandits allows us to study this behavior in a more formal way giving us strict guarantees regarding the performance of our algorithm.
- They form the linking pieces of a larger problem.
- They are easy to implement.

Some practical applications

- Selecting the best channel (out of several existing channels) for mobile communications in a very short duration.

Some practical applications

- Selecting the best channel (out of several existing channels) for mobile communications in a very short duration.
- Selecting a small set of best workers (out of a very large pool of workers) whose productivity is above a threshold.

Some practical applications

- Selecting the best channel (out of several existing channels) for mobile communications in a very short duration.
- Selecting a small set of best workers (out of a very large pool of workers) whose productivity is above a threshold.
- Selecting the best possible route for a message to pass through in a peer-to-peer network connection.

Stochastic Multi-Armed Bandit Problem

- In stochastic multi-armed bandit problem we are presented with a finite set of actions or arms.

Stochastic Multi-Armed Bandit Problem

- In stochastic multi-armed bandit problem we are presented with a finite set of actions or arms.
- The rewards for each of the arms is drawn from identical and independent distributions.

Stochastic Multi-Armed Bandit Problem

- In stochastic multi-armed bandit problem we are presented with a finite set of actions or arms.
- The rewards for each of the arms is drawn from identical and independent distributions.
- The learner does not know the mean of the distributions, denoted by μ_i.

Stochastic Multi-Armed Bandit Problem

- In stochastic multi-armed bandit problem we are presented with a finite set of actions or arms.
- The rewards for each of the arms is drawn from identical and independent distributions.
- The learner does not know the mean of the distributions, denoted by μ_i.
- The learner has to find the optimal arm the mean of whose distribution is denoted by μ^* such that $\mu^* > \mu_i, \forall i \in A$.

Stochastic Multi-Armed Bandit Problem

- In stochastic multi-armed bandit problem we are presented with a finite set of actions or arms.
- The rewards for each of the arms is drawn from identical and independent distributions.
- The learner does not know the mean of the distributions, denoted by μ_i.
- The learner has to find the optimal arm the mean of whose distribution is denoted by μ^* such that $\mu^* > \mu_i, \forall i \in A$.
- The distributions for each of the arms are fixed throughout the time horizon.

Basic Notations

- Goal: To minimize Regret

Basic Notations

- Goal: To minimize Regret
- Average reward of best action is μ^* and any other action i as μ_i. There are K total actions. $T_i(n)$ is number of times tried action i is executed till n-timesteps.

Basic Notations

- Goal: To minimize Regret
- Average reward of best action is μ^* and any other action i as μ_i. There are K total actions. $T_i(n)$ is number of times tried action i is executed till n-timesteps.
- Cumulative Regret: The loss we suffer because of not pulling the optimal arm till the total number of timesteps n.

$$R_n = r^* n - \sum_{i \in A} r_i T_i(n),$$

Basic Notations

- Goal: To minimize Regret
- Average reward of best action is μ^* and any other action i as μ_i. There are K total actions. $T_i(n)$ is number of times tried action i is executed till n-timesteps.
- Cumulative Regret: The loss we suffer because of not pulling the optimal arm till the total number of timesteps n.

$$R_n = r^* n - \sum_{i \in A} r_i T_i(n),$$

- The expected regret of an algorithm after n rounds can be written as

$$\mathbb{E}[R_n] = \sum_{i=1}^{K} \mathbb{E}[T_i(n)] \Delta_i,$$

Basic Notations

- Goal: To minimize Regret
- Average reward of best action is μ^* and any other action i as μ_i. There are K total actions. $T_i(n)$ is number of times tried action i is executed till n-timesteps.
- Cumulative Regret: The loss we suffer because of not pulling the optimal arm till the total number of timesteps n.

$$R_n = r^* n - \sum_{i \in A} r_i T_i(n),$$

- The expected regret of an algorithm after n rounds can be written as

$$\mathbb{E}[R_n] = \sum_{i=1}^{K} \mathbb{E}[T_i(n)]\Delta_i,$$

- $\Delta_i = r^* - r_i$ denotes the gap between the means of the optimal arm and of the i-th arm.

UCB 1 Algorithm

Algorithm 1 UCB1

1: Pull each arm once
2: **for** $t = K + 1, ..., T$ **do**
3: Pull the arm such that $\max_{i \in A} \left\{ \hat{\mu} + \sqrt{\frac{2 \log t}{n_i}} \right\}$
4: **end for**

[Auer et al.(2002a)Auer, Cesa-Bianchi, and Fischer]

Concentration Bounds

- The issue of coin tossing.

Concentration Bounds

- The issue of coin tossing.
- Chernoff-Hoeffding Bounds and its applications.

Concentration Bounds

- The issue of coin tossing.
- Chernoff-Hoeffding Bounds and its applications.
- Let $X_1, ..., X_n$ be random variables with common range [0, 1] and such that $E[X_t | X_1, ..., X_{t-1}] = \mu$. Let $\bar{S}_n = \frac{X_1 +, ..., +X_n}{n}$. Then for all $a \geq 0$,

$$\mathbb{P}\{\bar{S}_n \geq \mu + a\} \leq e^{-2a^2 n}$$
$$\mathbb{P}\{\bar{S}_n \leq \mu - a\} \leq e^{-2a^2 n}$$

UCB1 Theorem on Regret Bound

Theorem

For all $K > 1$, if policy UCB1 is run on K arms having arbitrary reward distributions $P_1, ..., P_K$ with support in $[0, 1]$, then its expected regret after any number n plays is at most,

$$\mathbb{E}[R_n] \leq \sum_{i \in A} \frac{8 \log n}{\Delta_i} + \sum_{i \in A} \Delta_i \left(1 + \frac{\pi^2}{3}\right)$$

where $\mu_1, ..., \mu_K$ are the expected values of $P_1, ..., P_K$.

UCB1 Proof

- The main goal is to bound the number of pulls ($T_i(n)$) of the sub-optimal arm i till the n-th timestep.

UCB1 Proof

- The main goal is to bound the number of pulls ($T_i(n)$) of the sub-optimal arm i till the n-th timestep.
- So, we will assume that the i-th arm has been pulled atleast ℓ times and bound the probability of how many times it can be pulled after that.

$$T_i(n) \leq \ell + \sum_{t=K+1}^{n} \{I_t = i, T_i(t-1) \geq \ell\}$$

UCB1 Proof

- The main goal is to bound the number of pulls ($T_i(n)$) of the sub-optimal arm i till the n-th timestep.
- So, we will assume that the i-th arm has been pulled atleast ℓ times and bound the probability of how many times it can be pulled after that.

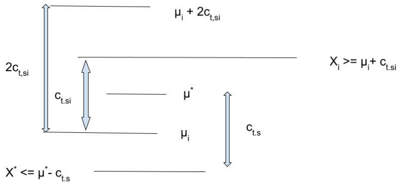
$$T_i(n) \le \ell + \sum_{t=K+1}^{n} \{I_t = i, T_i(t-1) \ge \ell\}$$

- But, this is nothing but the probability that how many times after the ℓ pulls the UCB of $*$ is less than the UCB of i which will have the highest UCB among all arms in A to be selected,

$$T_i(n) \le \ell + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_i=\ell}^{t-1} \{\bar{X}_s^* + c_{t,s} \le \bar{X}_{i,s_i} + c_{t,s_i}\}$$

UCB1 Proof

- The main argument lies in this,



-

$$\bar{X}_s^* \leq \mu^* - c_{t,s}$$
$$\bar{X}_{i,s_i} \geq \mu + c_{t,s_i}$$
$$\mu^* < \mu_i + 2c_{t,s_i}$$

UCB1 Proof

- Now, we get the value of confidence interval $c_{t,s_i} = \sqrt{\frac{2\ln t}{s_i}}$ by plugging its value in the below equations,
- $\mathbb{P}\{\bar{X}_s^* \leq \mu^* - c_{t,s}\} \leq \exp\left(-2(\sqrt{\frac{2\ln t}{s}})^2 s\right) \leq e^{-4\log t} \leq t^{-4}$

UCB1 Proof

- Now, we get the value of confidence interval $c_{t,s_i} = \sqrt{\frac{2\ln t}{s_i}}$ by plugging its value in the below equations,
- $\mathbb{P}\{\bar{X}_s^* \le \mu^* - c_{t,s}\} \le \exp\left(-2(\sqrt{\frac{2\ln t}{s}})^2 s\right) \le e^{-4\log t} \le t^{-4}$
- $\mathbb{P}\{\bar{X}_{i,s_i} \ge \mu + c_{t,s_i}\} \le \exp\left(-2(\sqrt{\frac{2\ln t}{s_i}})^2 s_i\right) \le e^{-4\log t} \le t^{-4}$

UCB1 Proof

- Now, we get the value of confidence interval $c_{t,s_i} = \sqrt{\frac{2\ln t}{s_i}}$ by plugging its value in the below equations,
- $\mathbb{P}\{\bar{X}^*_s \le \mu^* - c_{t,s}\} \le \exp\left(-2(\sqrt{\frac{2\ln t}{s}})^2 s\right) \le e^{-4\log t} \le t^{-4}$
- $\mathbb{P}\{\bar{X}_{i,s_i} \ge \mu + c_{t,s_i}\} \le \exp\left(-2(\sqrt{\frac{2\ln t}{s_i}})^2 s_i\right) \le e^{-4\log t} \le t^{-4}$
- And by plugging $\ell = \left\lceil \frac{8\log n}{\Delta_i^2} \right\rceil$ in,

$$\mu^* - \mu_i - 2c_{t,s_i} = \mu^* - \mu_i - 2\sqrt{\frac{2\log t}{s_i}} \ge \mu^* - \mu_i - \Delta_i = 0$$

we get $\mu^* - \mu_i - 2c_{t,s_i} \ge 0$. So for any pulls greater than ℓ, μ^* will surely be atleast $2c_{t,s_i}$ more than μ_i and one of the rest two events will occur with high probability.

UCB1 Proof

- Summing everything up, any sub-optimal arm i will get pulled atleast ℓ times and then the two events $\bar{X}^*_s \leq \mu^* - c_{t,s}$ and $\bar{X}_{i,s_i} \geq \mu + c_{t,s_i}$ will occur with atmost t^{-4} probability.

UCB1 Proof

- Summing everything up, any sub-optimal arm i will get pulled atleast ℓ times and then the two events $\bar{X}^*_s \leq \mu^* - c_{t,s}$ and $\bar{X}_{i,s_i} \geq \mu + c_{t,s_i}$ will occur with atmost t^{-4} probability.
- $$\begin{aligned}\mathbb{E}[T_i(n)] &\leq \left\lceil \frac{8 \log n}{\Delta_i^2} \right\rceil + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_i=\ell}^{t-1} 2t^{-4} \\ &\leq \frac{8 \log n}{\Delta_i^2} + 1 + \frac{\pi^2}{3}, \text{by Bazel's equation}\end{aligned}$$

UCB1 Proof

- Summing everything up, any sub-optimal arm i will get pulled atleast ℓ times and then the two events $\bar{X}_s^* \leq \mu^* - c_{t,s}$ and $\bar{X}_{i,s_i} \geq \mu + c_{t,s_i}$ will occur with atmost t^{-4} probability.
- $$\begin{aligned}\mathbb{E}[T_i(n)] &\leq \left\lceil \frac{8 \log n}{\Delta_i^2} \right\rceil + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_i=\ell}^{t-1} 2t^{-4} \\ &\leq \frac{8 \log n}{\Delta_i^2} + 1 + \frac{\pi^2}{3}, \text{by Bazel's equation}\end{aligned}$$
- So finally the cumulative regret is,

$$\mathbb{E}[R_n] \leq \sum_{i \in A} \mathbb{E}[T_i(n)]\Delta_i \leq \frac{8 \log n}{\Delta_i} + \Delta_i \left(1 + \frac{\pi^2}{3}\right)$$

Looking beyond Cumulative regret

- Sometimes rather than worrying about cumulative regret we just want to be satisfied with a *nearly good* arm with *high probability*.

Looking beyond Cumulative regret

- Sometimes rather than worrying about cumulative regret we just want to be satisfied with a *nearly good* arm with *high probability*.
- This is called the (ϵ, δ)-guarantee or PAC-guarantee.

Looking beyond Cumulative regret

- Sometimes rather than worrying about cumulative regret we just want to be satisfied with a *nearly good* arm with *high probability*.
- This is called the (ϵ, δ)-guarantee or PAC-guarantee.
- We are interested in finding an arm such that it's ϵ close to the optimal arm and we can guarantee this with $1 - \delta$ probability.

Looking beyond Cumulative regret

- Sometimes rather than worrying about cumulative regret we just want to be satisfied with a *nearly good* arm with *high probability*.
- This is called the (ϵ, δ)-guarantee or PAC-guarantee.
- We are interested in finding an arm such that it's ϵ close to the optimal arm and we can guarantee this with $1 - \delta$ probability.
- Note, that ϵ and δ are given as input and the main aim is to *minimize the number of pulls of an arm i so that it is ϵ close to the optimal arm with $1 - \delta$ probability*. This is called Sample complexity.

A Naive Algorithm

Algorithm 2 Naive Algorithm

1: Input: $\epsilon > 0$, $\delta > 0$
2: Output: An arm
3: **for** each arm $i \in A$ **do**
4: Sample it for $\ell = \dfrac{4}{\epsilon^2} \log \dfrac{2K}{\delta}$
5: Let $\bar{X}_i$ be the average reward of arm i
6: **end for**
7: Output $argmax_{i \in A}\{\bar{X}_i\}$

[Even-Dar et al.(2006)Even-Dar, Mannor, and Mansour]

Sample Complexity of Naive Algorithm

Theorem

The sample complexity of Naive Algorithm for a set of arms K is given by,

$$O\left(\frac{K}{\epsilon^2}\log\left(\frac{K}{\delta}\right)\right)$$

[Even-Dar et al.(2006)Even-Dar, Mannor, and Mansour]

Naive Algorithm Sample Complexity Proof

- We want to bound the probability of the event $\bar{X}_i > \bar{X}^*$.

Naive Algorithm Sample Complexity Proof

- We want to bound the probability of the event $\bar{X}_i > \bar{X}^*$.
- But by (ϵ, δ) definition we need *i* only to be epsilon close to $*$. This means $\mu_i < \mu^* - \epsilon \rightarrow \mu_i + \frac{\epsilon}{2} < \mu^* - \frac{\epsilon}{2}$.

Naive Algorithm Sample Complexity Proof

- We want to bound the probability of the event $\bar{X}_i > \bar{X}^*$.
- But by (ϵ, δ) definition we need i only to be epsilon close to $*$. This means $\mu_i < \mu^* - \epsilon \to \mu_i + \frac{\epsilon}{2} < \mu^* - \frac{\epsilon}{2}$.
- So, we only need to bound the probability of,

$$\mathbb{P}\{\bar{X}_i > \bar{X}^*\} \leq \mathbb{P}\left\{\bar{X}_i > \mu_i + \frac{\epsilon}{2}\right\} + \mathbb{P}\left\{\bar{X}^* < \mu^* - \frac{\epsilon}{2}\right\}$$

$$\leq 2\exp\left(-2\left(\frac{\epsilon}{2}\right)^2 \ell\right) \leq 2\exp\left(-2\frac{\epsilon^2}{4}.\frac{4}{\epsilon^2}\log\frac{2K}{\delta}\right) \leq \frac{\delta}{K}$$

Naive Algorithm Sample Complexity Proof

- We want to bound the probability of the event $\bar{X}_i > \bar{X}^*$.
- But by (ϵ, δ) definition we need i only to be epsilon close to $*$. This means $\mu_i < \mu^* - \epsilon \rightarrow \mu_i + \frac{\epsilon}{2} < \mu^* - \frac{\epsilon}{2}$.
- So, we only need to bound the probability of,

$$\mathbb{P}\{\bar{X}_i > \bar{X}^*\} \leq \mathbb{P}\left\{\bar{X}_i > \mu_i + \frac{\epsilon}{2}\right\} + \mathbb{P}\left\{\bar{X}^* < \mu^* - \frac{\epsilon}{2}\right\}$$
$$\leq 2\exp\left(-2\left(\frac{\epsilon}{2}\right)^2 \ell\right) \leq 2\exp\left(-2\frac{\epsilon^2}{4}.\frac{4}{\epsilon^2}\log\frac{2K}{\delta}\right) \leq \frac{\delta}{K}$$

- Summing over all the $K-1$ arms (arms excluding $*$) we get, $\frac{(K-1)\delta}{K} < \delta$

Intuition about Median Elimination

- Can we be more powerful than using Naive Algorithm?

Intuition about Median Elimination

- Can we be more powerful than using Naive Algorithm?
- One simple way to modify Naive Algorithm is to divide the time horizon into phases.

Intuition about Median Elimination

- Can we be more powerful than using Naive Algorithm?
- One simple way to modify Naive Algorithm is to divide the time horizon into phases.
- In each phase pull all the arms equal number of times.

Intuition about Median Elimination

- Can we be more powerful than using Naive Algorithm?
- One simple way to modify Naive Algorithm is to divide the time horizon into phases.
- In each phase pull all the arms equal number of times.
- After that eliminate half the arms with a high guarantee that they are surely not ϵ-optimal arms.

Median Elimination

Algorithm 3 Median Elimination

Input: $\epsilon > 0$, $\delta > 0$
Output: An arm
Set $S_1 = A$, $\epsilon_1 = \epsilon/4$, $\delta_1 = \delta/2$ and $\ell = 1$
for Repeat till $|S_\ell| = 1$ **do**
    Sample every arm in S_ℓ for $\frac{4}{\epsilon_\ell^2}\log(\frac{3}{\delta_\ell})$ times and let $\bar{X}_i$ denote the average estimated payoff of i.
    Find median m_ℓ of all the surviving arms based on their $\bar{X}_i, \forall i \in A$
    Eliminate all arms from S_ℓ such that $\bar{X}_i < m_\ell$ and create $S_{\ell+1}$.
    Reset Parameters: $\epsilon_{\ell+1} = \frac{3}{4}\epsilon$; $\delta_{\ell+1} = \frac{1}{2}\delta$; $\ell = \ell + 1$
end for

[Even-Dar et al.(2006)Even-Dar, Mannor, and Mansour]

Comparison of Median Elimination, Naive Algorithm

Table: Sample Complexity of Median Elimination, Naive Algorithm

Algorithm	Upper bound on Sample Complexity
Naive	$O\left(\frac{K}{\epsilon^2} \log\left(\frac{K}{\delta}\right)\right)$
ME	$O\left(\frac{K}{\epsilon^2} \log big(\frac{1}{\delta})\right)$

- So clearly Naive algorithm uses more samples than Median Elimination to give us the same (ϵ, δ) guarantee

Arm Elimination Algorithm

- Arm Elimination (AE) algorithms proceeds in a phase-wise manner whereby they pull all the surviving arms equal number of times in each phase and then at the end of the phase proceeds to eliminate one or more arms from its active set.

Arm Elimination Algorithm

- Arm Elimination (AE) algorithms proceeds in a phase-wise manner whereby they pull all the surviving arms equal number of times in each phase and then at the end of the phase proceeds to eliminate one or more arms from its active set.
- We have seen that Median Elimination algorithm which is an AE algorithm, is more powerful than Naive Algorithm.

Arm Elimination Algorithm

- Arm Elimination (AE) algorithms proceeds in a phase-wise manner whereby they pull all the surviving arms equal number of times in each phase and then at the end of the phase proceeds to eliminate one or more arms from its active set.
- We have seen that Median Elimination algorithm which is an AE algorithm, is more powerful than Naive Algorithm.
- Can we have such algorithm for minimizing Cumulative Regret?

Enter UCB-Improved

- The basic idea of UCB-Improved ([Auer and Ortner(2010)]) is nearly same as ME.

Enter UCB-Improved

- The basic idea of UCB-Improved ([Auer and Ortner(2010)]) is nearly same as ME.
- Divide the horizon into phases and initialize parameters.

Enter UCB-Improved

- The basic idea of UCB-Improved ([Auer and Ortner(2010)]) is nearly same as ME.
- Divide the horizon into phases and initialize parameters.
- Pull all surviving arms equal number of times during a phase.

Enter UCB-Improved

- The basic idea of UCB-Improved ([Auer and Ortner(2010)]) is nearly same as ME.
- Divide the horizon into phases and initialize parameters.
- Pull all surviving arms equal number of times during a phase.
- At the end of the phase eliminate some arms based on some criteria.

Enter UCB-Improved

- The basic idea of UCB-Improved ([Auer and Ortner(2010)]) is nearly same as ME.
- Divide the horizon into phases and initialize parameters.
- Pull all surviving arms equal number of times during a phase.
- At the end of the phase eliminate some arms based on some criteria.
- Reset parameters and proceed to next phase.

Enter UCB-Improved

Algorithm 4 UCB-Improved

Input: Time horizon n

Initialization: Set $B_0 := A$ and $\epsilon_0 := 1$.

for $m = 0, 1, .. \lfloor \frac{1}{2} \log_2 \frac{n}{e} \rfloor$ **do**

Pull each arm in B_m so that the total number of times it has been pulled is $n_m = \left\lceil \frac{2 \log (n\epsilon_m^2)}{\epsilon_m} \right\rceil$.

Arm Elimination

For each $i \in B_m$, delete arm i from B_m if,

$$\bar{X}_i + \sqrt{\frac{\log (n\epsilon_m^2)}{2n_m}} < \max_{j \in B_m} \left\{ \bar{X}_j - \sqrt{\frac{\log (n\epsilon_m^2)}{2n_m}} \right\}$$

Reset Parameters

Set $\epsilon_{m+1} := \frac{\epsilon_m}{2}$, Set $B_{m+1} := B_m$

Comparison of UCB-Improved,ME and UCB1

Table: Cumulative Regret of UCB-Improved, UCB1

Algorithm	Upper bound on Cumulative Regret
UCB1	$O\left(\frac{K \log n}{\Delta}\right)$
UCB-Improved	$O\left(\frac{K \log(n\Delta^2)}{\Delta}\right)$

- So, UCB-Improved is more powerful than UCB1 theoretically.

Comparison of UCB-Improved,ME and UCB1

Table: Cumulative Regret of UCB-Improved, UCB1

Algorithm	Upper bound on Cumulative Regret
UCB1	$O\left(\frac{K \log n}{\Delta}\right)$
UCB-Improved	$O\left(\frac{K \log(n\Delta^2)}{\Delta}\right)$

- So, UCB-Improved is more powerful than UCB1 theoretically.
- UCB-Improved is more powerful than ME, because ME will always take $\log_2 K$ number of phases to come up with the best arm (since it eliminates half the number of arms after every phase) whereas UCB-Improved eliminates arbitrary number of arms in any phase.

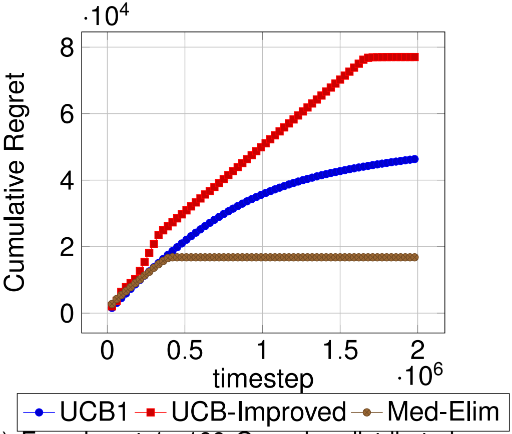
Comparison of UCB-Improved,ME and UCB1

Table: Cumulative Regret of UCB-Improved, UCB1

Algorithm	Upper bound on Cumulative Regret
UCB1	$O\left(\frac{K \log n}{\Delta}\right)$
UCB-Improved	$O\left(\frac{K \log(n\Delta^2)}{\Delta}\right)$

- So, UCB-Improved is more powerful than UCB1 theoretically.
- UCB-Improved is more powerful than ME, because ME will always take $\log_2 K$ number of phases to come up with the best arm (since it eliminates half the number of arms after every phase) whereas UCB-Improved eliminates arbitrary number of arms in any phase.
- Empirically, UCB-Improved beats UCB1 when K is very large and gaps ($\Delta_i, \forall i \in A$) are very small.

Finally, an experiment!!!



(a) Experiment 1: 100 Gaussian-distributed arms with $\mu_{i_{i \neq *:1-33}} = 0.01$, $\mu_{i_{i \neq *:34-99}} = 0.06$, $r^*_{i=100} = 0.1$ and $\sigma^2_{i:1-100} = 0.3$

Figure: Experiment with bandit

Some Other Bandits and Applications

- Adversarial Bandits : Used in Investment in Stock Markets
- Contextual Bandits : Used in online Advertisement/news article selection
- Budgeted Bandits : Used in Clinical trials
- Distributed Bandits : Used in packet routing through a network

References I

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári.
Improved algorithms for linear stochastic bandits.
In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

Shipra Agrawal and Navin Goyal.
Analysis of thompson sampling for the multi-armed bandit problem.
arXiv preprint arXiv:1111.1797, 2011.

Jean-Yves Audibert and Sébastien Bubeck.
Minimax policies for adversarial and stochastic bandits.
In *COLT*, pages 217–226, 2009.

References II

Jean-Yves Audibert and Sébastien Bubeck.
Best arm identification in multi-armed bandits.
In *COLT-23th Conference on Learning Theory-2010*, pages 13–p, 2010.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári.
Exploration–exploitation tradeoff using variance estimates in multi-armed bandits.
Theoretical Computer Science, 410(19):1876–1902, 2009.

Peter Auer and Ronald Ortner.
Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem.
Periodica Mathematica Hungarica, 61(1-2):55–65, 2010.

References III

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer.
Finite-time analysis of the multiarmed bandit problem.
Machine learning, 47(2-3):235–256, 2002a.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire.
The nonstochastic multiarmed bandit problem.
SIAM Journal on Computing, 32(1):48–77, 2002b.

Sébastien Bubeck and Nicolo Cesa-Bianchi.
Regret analysis of stochastic and nonstochastic multi-armed bandit problems.
arXiv preprint arXiv:1204.5721, 2012.

Sébastien Bubeck, Nicolo Cesa-Bianchi, and Gábor Lugosi.
Bandits with heavy tail.
arXiv preprint arXiv:1209.1727, 2012.

References IV

Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet.
Bounded regret in stochastic multi-armed bandits.
arXiv preprint arXiv:1302.1611, 2013.

Olivier Cappe, Aurelien Garivier, and Emilie Kaufmann.
pymabandits, 2012.
http://mloss.org/software/view/415/.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour.
Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems.
The Journal of Machine Learning Research, 7:1079–1105, 2006.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani.
The elements of statistical learning, volume 1.
Springer series in statistics Springer, Berlin, 2001.

References V

Aurélien Garivier and Olivier Cappé.
The kl-ucb algorithm for bounded stochastic bandits and beyond.
arXiv preprint arXiv:1102.2490, 2011.

Junya Honda and Akimichi Takemura.
An asymptotically optimal bandit algorithm for bounded support models.
In *COLT*, pages 67–79. Citeseer, 2010.

Tze Leung Lai and Herbert Robbins.
Asymptotically efficient adaptive allocation rules.
Advances in applied mathematics, 6(1):4–22, 1985.

Tor Lattimore.
Optimally confident ucb: Improved regret for finite-armed bandits.
arXiv preprint arXiv:1507.07880, 2015.

References VI

Yun-Ching Liu and Yoshimasa Tsuruoka.
Modification of improved upper confidence bounds for regulating exploration in monte-carlo tree search.
Theoretical Computer Science, 2016.

Shie Mannor and John N Tsitsiklis.
The sample complexity of exploration in the multi-armed bandit problem.
Journal of Machine Learning Research, 5(Jun):623–648, 2004.

Vianney Perchet, Philippe Rigollet, Sylvain Chassang, and Erik Snowberg.
Batched bandit problems.
arXiv preprint arXiv:1505.00369, 2015.

References VII

Herbert Robbins.
Some aspects of the sequential design of experiments.
In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1952.

Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction.
MIT press, 1998.

William R Thompson.
On the likelihood that one unknown probability exceeds another in view of the evidence of two samples.
Biometrika, pages 285–294, 1933.

David Tolpin and Solomon Eyal Shimony.
Mcts based on simple regret.
In *AAAI*, 2012.

Thank You