Online sequential Learning using Bandits

Subhojyoti Mukherjee
Dr. Balaraman Ravindran (Advisor, CSE Dept.)
Dr. Nandan Sudarsanam (Co-Advisor, DoMS Dept.)

IIT Madras

July 8, 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.
- There is a specific reward distribution attached to each arm. After pulling an arm we receive a reward from the reward distribution specific to the arm.

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.
- There is a specific reward distribution attached to each arm. After pulling an arm we receive a reward from the reward distribution specific to the arm.
- After say pulling each arm once, we are presented with an *exploration-exploitation* trade-off, 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.
- There is a specific reward distribution attached to each arm. After pulling an arm we receive a reward from the reward distribution specific to the arm.
- After say pulling each arm once, we are presented with an *exploration-exploitation* trade-off, 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.

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.

Why study bandits at all?

- We know of ϵ-greedy [**?**] algorithm, we can simply stick to it.

Why study bandits at all?

- We 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 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 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 are easy to implement.

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 are identical and independent random variables drawn from distribution specific to the arm.

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 are identical and independent random variables drawn from distribution specific to the arm.
- The learner does not know the mean of the distributions, denoted by $r_i, \forall i \in A$.

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 are identical and independent random variables drawn from distribution specific to the arm.
- The learner does not know the mean of the distributions, denoted by $r_i, \forall i \in A$.
- The learner has to find the optimal arm the mean of whose distribution is denoted by r^* such that $r^* > r_i, \forall i \in A$.

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 are identical and independent random variables drawn from distribution specific to the arm.
- The learner does not know the mean of the distributions, denoted by $r_i, \forall i \in A$.
- The learner has to find the optimal arm the mean of whose distribution is denoted by r^* such that $r^* > r_i, \forall i \in A$.
- 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 r^* and any other action i as r_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 r^* and any other action i as r_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 T.

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

Basic Notations

- Goal: To minimize Regret
- Average reward of best action is r^* and any other action i as r_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 T.

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

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

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

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

Another Notion of Regret

- Goal: To minimize Regret

Another Notion of Regret

- Goal: To minimize Regret
- Can we have a policy which achieves the minimum regret among all the possible environments available?

Another Notion of Regret

- Goal: To minimize Regret
- Can we have a policy which achieves the minimum regret among all the possible environments available?
- This is called the worst case gap-independent regret.

Another Notion of Regret

- Goal: To minimize Regret
- Can we have a policy which achieves the minimum regret among all the possible environments available?
- This is called the worst case gap-independent regret.
- It is generally found by setting all the gaps to equal values of $\min_{i \in A} \Delta_i = \Delta = \sqrt{\frac{f(K)}{T}}$.

Another Notion of Regret

- Goal: To minimize Regret
- Can we have a policy which achieves the minimum regret among all the possible environments available?
- This is called the worst case gap-independent regret.
- It is generally found by setting all the gaps to equal values of $\min_{i \in A} \Delta_i = \Delta = \sqrt{\dfrac{f(K)}{T}}$.
- Also we will define the hardness parameter H as $H = \sum_{i=1}^{K} \dfrac{1}{\Delta_i^2}$

Literature Survey

- A considerable amount of research has been conducted on SMAB. We can divide the gamut of literature on SMAB into broadly two categories:

Literature Survey

- A considerable amount of research has been conducted on SMAB. We can divide the gamut of literature on SMAB into broadly two categories:
  - **Frequentist approach:** In this approach for each of the arms compute a dynamic allocation index that depends only on the number of draws on the arm and exploration parameters and choose the arm with the maximal index. Eg: *UCB* variants

Literature Survey

- A considerable amount of research has been conducted on SMAB. We can divide the gamut of literature on SMAB into broadly two categories:
  - **Frequentist approach:** In this approach for each of the arms compute a dynamic allocation index that depends only on the number of draws on the arm and exploration parameters and choose the arm with the maximal index. Eg: *UCB* variants
  - **Bayesian Approach:** In this approach we start with a prior guess over the performance of each of the arms. Then we pull an arm by behaving greedily based on our guess, receive the reward and then we update our prior for the arm. Eg: TS

Literature Survey

- A considerable amount of research has been conducted on SMAB. We can divide the gamut of literature on SMAB into broadly two categories:
  - **Frequentist approach:** In this approach for each of the arms compute a dynamic allocation index that depends only on the number of draws on the arm and exploration parameters and choose the arm with the maximal index. Eg: *UCB* variants
  - **Bayesian Approach:** In this approach we start with a prior guess over the performance of each of the arms. Then we pull an arm by behaving greedily based on our guess, receive the reward and then we update our prior for the arm. Eg: TS
- We will be focusing on UCB based approaches in our work.

Literature Survey

- The literature under UCB can be broadly classified into two categories:

Literature Survey

- The literature under UCB can be broadly classified into two categories:
  - **Mean-based estimation:** In this approach at every timestep we choose an arm based on $\hat{r}_i$ and its confidence interval c_i. Eg: UCB1 [**?**], MOSS [**?**], UCB-Improved [**?**]

Literature Survey

- The literature under UCB can be broadly classified into two categories:
  - **Mean-based estimation:** In this approach at every timestep we choose an arm based on $\hat{r}_i$ and its confidence interval c_i. Eg: UCB1 [**?**], MOSS [**?**], UCB-Improved [**?**]
  - **Mean and Variance based Estimation:** Here, at every timestep we choose an arm based on $\hat{r}_i$, $\hat{V}_i$ and its confidence interval c_i. Eg: UCB-Normal [**?**]l, UCB-V [**?**].

Literature Survey

- The literature under UCB can be broadly classified into two categories:
  - **Mean-based estimation:** In this approach at every timestep we choose an arm based on $\hat{r}_i$ and its confidence interval c_i. Eg: UCB1 [**?**], MOSS [**?**], UCB-Improved [**?**]
  - **Mean and Variance based Estimation:** Here, at every timestep we choose an arm based on $\hat{r}_i$, $\hat{V}_i$ and its confidence interval c_i. Eg: UCB-Normal [**?**]l, UCB-V [**?**].
- We will be focusing on Mean based estimation.

UCB1 Algorithm ([?])

Algorithm 1 UCB1

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

- Maintain an upper confidence bound (c_i) for each of the arms
- This c_i will help in sufficiently exploring sub-optimal arms and then exploiting the optimal arm.
- The gap-independent regret bound of $O\left(\sqrt{KT \log T}\right)$ and gap-dependent bound of $O\left(\frac{K \log(T)}{\Delta}\right)$.

Algorithm 2 MOSS

1: Pull each arm once
2: **for** $t = K+1, ..., T$ **do**
3: Pull the arm such that $\max_{i \in A} \left\{ \hat{r}_i + \sqrt{\frac{\max\{0, \log(\frac{T}{Ks_i})\}}{s_i}} \right\}$
4: $t := t+1$
5: **end for**

- UCB1 suffers from a worst case regret of $O\left(\sqrt{KT \log T}\right)$.
- MOSS corrects this and gives us a gap-independent regret bound of $O\left(\sqrt{KT}\right)$ and gap-dependent bound of $O\left(\frac{K^2 \log(\frac{T\Delta^2}{K})}{\Delta}\right)$.

Approach of UCB-Improved

- The basic idea of UCB-Improved is to divide the horizon into phases or rounds and initialize parameters.

Approach of UCB-Improved

- The basic idea of UCB-Improved is to divide the horizon into phases or rounds and initialize parameters.
- Pull all surviving arms equal number of times during a round.

Approach of UCB-Improved

- The basic idea of UCB-Improved is to divide the horizon into phases or rounds and initialize parameters.
- Pull all surviving arms equal number of times during a round.
- At the end of the round eliminate some arms based on some criteria.

Approach of UCB-Improved

- The basic idea of UCB-Improved is to divide the horizon into phases or rounds and initialize parameters.
- Pull all surviving arms equal number of times during a round.
- At the end of the round eliminate some arms based on some criteria.
- Reset parameters and proceed to next round.

Approach of UCB-Improved

- The basic idea of UCB-Improved is to divide the horizon into phases or rounds and initialize parameters.
- Pull all surviving arms equal number of times during a round.
- At the end of the round eliminate some arms based on some criteria.
- Reset parameters and proceed to next round.
- UCB-Imp achieves a gap-independent regret bound of $O\left(\sqrt{KT\log K}\right)$ and gap-dependent bound of $O\left(\frac{K\log(T\Delta^2)}{\Delta}\right)$.

UCB-Improved ([?])

Algorithm 3 UCB-Improved

Input: Time horizon T

Initialization: Set $B_0 := A$ and $\tilde{\Delta}_0 := 1$.

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

Pull each arm in B_m, $n_m = \left\lceil \frac{2 \log(T\tilde{\Delta}_m^2)}{\tilde{\Delta}_m} \right\rceil$ number of times.

Arm Elimination

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

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

Set $\tilde{\Delta}_{m+1} := \frac{\tilde{\Delta}_m}{2}$, Set $B_{m+1} := B_m$

Stop if $|B_m| = 1$ and pull $i \in B_m$ till n is reached.

end for

Some technical details of UCB-Improved

- We do not know the true means $r_i, \forall i \in A$ of the distributions so we estimate it by the $\tilde{\Delta}$ by initializing it from 1.

Some technical details of UCB-Improved

- We do not know the true means $r_i, \forall i \in A$ of the distributions so we estimate it by the $\tilde{\Delta}$ by initializing it from 1.
- All rewards are assume to be bounded between $[0, 1]$ and so $\Delta_i \in [0, 1], \forall i \in A$ as well.

Some technical details of UCB-Improved

- We do not know the true means $r_i, \forall i \in A$ of the distributions so we estimate it by the $\tilde{\Delta}$ by initializing it from 1.
- All rewards are assume to be bounded between $[0, 1]$ and so $\Delta_i \in [0, 1], \forall i \in A$ as well.
- As opposed to UCB1, MOSS and OCUCB, UCB-Improved has fixed confidence interval $c_m = \sqrt{\dfrac{\log(T\tilde{\Delta}_m^2)}{2n_m}}$ for all arms in a particular phase.

Some technical details of UCB-Improved

- We do not know the true means $r_i, \forall i \in A$ of the distributions so we estimate it by the $\tilde{\Delta}$ by initializing it from 1.
- All rewards are assume to be bounded between $[0, 1]$ and so $\Delta_i \in [0, 1], \forall i \in A$ as well.
- As opposed to UCB1, MOSS and OCUCB, UCB-Improved has fixed confidence interval $c_m = \sqrt{\dfrac{\log(T\tilde{\Delta}_m^2)}{2n_m}}$ for all arms in a particular phase.
- c_m ensures that whenever $\tilde{\Delta}_m < \dfrac{\Delta_i}{2}$ in the m-th round, the arm i gets eliminated.

Optimally confident UCB ([?])

Algorithm 4 MOSS

1: **Input:** K,T, α, ψ
2: Pull each arm once
3: **for** $t = K + 1, ..., T$ **do**
4: Pull the arm such that $\max_{i \in A} \left\{ \hat{r}_i + \sqrt{\alpha \frac{\max\{0, \log(\frac{\psi T}{s_i})\}}{s_i}} \right\}$
5: $t := t + 1$
6: **end for**

- UCB1 is too conservative in exploiting, MOSS is not conservative enough and tends to explore more often than required.
- OCUCB correctly balances this and achieves a gap-independent regret bound $O\left(\sqrt{KT}\right)$ and gap-dependent bound $O\left(\frac{K \log(T/H)}{\Delta}\right)$.

Comparison of UCB1, MOSS, OCUCB, UCB-Improved

Table: Cumulative Regret of Algorithms

Algorithm	Upper bound on Cumulative Regret
UCB1	$\min\left\{O\left(\frac{K\log T}{\Delta}\right), O\left(\sqrt{KT\log T}\right)\right\}$
MOSS	$\min\left\{O\left(\frac{K^2\log(T\Delta^2/K)}{\Delta}\right), O\left(\sqrt{KT}\right)\right\}$
UCB-Improved	$\min\left\{O\left(\frac{K\log(T\Delta^2)}{\Delta}\right), O\left(\sqrt{KT\log K}\right)\right\}$
OCUCB	$\min\left\{O\left(\frac{K\log(T/H)}{\Delta}\right), O\left(\sqrt{KT}\right)\right\}$

Problems of UCB-Improved

- UCB-Improved conducts too much early exploration.

Problems of UCB-Improved

- UCB-Improved conducts too much early exploration.
- The arm elimination condition of UCB-Imp is very conservative.

Problems of UCB-Improved

- UCB-Improved conducts too much early exploration.
- The arm elimination condition of UCB-Imp is very conservative.
- When the gaps are small and uniform UCB-Imp performs very badly.

Problems of UCB-Improved

- UCB-Improved conducts too much early exploration.
- The arm elimination condition of UCB-Imp is very conservative.
- When the gaps are small and uniform UCB-Imp performs very badly.
- There is a gap in theoretical guarantee and empirical performance.

Problem Definition

- In our work, we ask the following set of questions:

Problem Definition

- In our work, we ask the following set of questions:
  - Can we achieve a better gap-dependent/gap-independent regret bound than UCB-Improved?

Problem Definition

- In our work, we ask the following set of questions:
  - Can we achieve a better gap-dependent/gap-independent regret bound than UCB-Improved?
  - Can we bridge the theoretical versus empirical performance of UCB-Improved?

Problem Definition

- In our work, we ask the following set of questions:
  - Can we achieve a better gap-dependent/gap-independent regret bound than UCB-Improved?
  - Can we bridge the theoretical versus empirical performance of UCB-Improved?
  - Can we use ideas from Clustering to achieve this?

Problem Definition

- In our work, we ask the following set of questions:
  - Can we achieve a better gap-dependent/gap-independent regret bound than UCB-Improved?
  - Can we bridge the theoretical versus empirical performance of UCB-Improved?
  - Can we use ideas from Clustering to achieve this?
  - Can we study the effect of Clustering in SMAB?

Problem Definition

- In our work, we ask the following set of questions:
  - Can we achieve a better gap-dependent/gap-independent regret bound than UCB-Improved?
  - Can we bridge the theoretical versus empirical performance of UCB-Improved?
  - Can we use ideas from Clustering to achieve this?
  - Can we study the effect of Clustering in SMAB?
- The answer to all of this is ClusUCB.

Approach of ClusUCB

- The basic idea of ClusUCB directly follows from UCB-Imp. It starts by dividing the horizon into rounds and initializing parameters.

Approach of ClusUCB

- The basic idea of ClusUCB directly follows from UCB-Imp. It starts by dividing the horizon into rounds and initializing parameters.
- It then creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.

Approach of ClusUCB

- The basic idea of ClusUCB directly follows from UCB-Imp. It starts by dividing the horizon into rounds and initializing parameters.
- It then creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Pull all surviving arms equal number of times during a round.

Approach of ClusUCB

- The basic idea of ClusUCB directly follows from UCB-Imp. It starts by dividing the horizon into rounds and initializing parameters.
- It then creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Pull all surviving arms equal number of times during a round.
- At the end of the round eliminate arms inside each cluster by comparing its performance against the best arm in the cluster.

Approach of ClusUCB

- The basic idea of ClusUCB directly follows from UCB-Imp. It starts by dividing the horizon into rounds and initializing parameters.
- It then creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Pull all surviving arms equal number of times during a round.
- At the end of the round eliminate arms inside each cluster by comparing its performance against the best arm in the cluster.
- Also eliminate clusters with all of its arms by comparing its performance against the globally best arm.
- Reset parameters and move to the next round.

Approach of ClusUCB

- The basic idea of ClusUCB directly follows from UCB-Imp. It starts by dividing the horizon into rounds and initializing parameters.
- It then creates *p* fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Pull all surviving arms equal number of times during a round.
- At the end of the round eliminate arms inside each cluster by comparing its performance against the best arm in the cluster.
- Also eliminate clusters with all of its arms by comparing its performance against the globally best arm.
- Reset parameters and move to the next round.
- At a higher level ClusUCB behaves like *p* independently running UCB-Imp with the exploration parameters ρ_a, ρ_s and ψ helping in overcoming early exploration.

ClusUCB Algorithm I

1: **Input:** Number of clusters p, time horizon T, exploration parameters ρ_a, ρ_s and ψ.

2: **Initialization:** Set $B_0 := A$, $S_0 = S$ and $\epsilon_0 := 1$.

3: Create a partition S_0 of the arms at random into p clusters of size up to $\ell = \left\lceil \frac{K}{p} \right\rceil$ each.

4: **for** $m = 0, 1, .. \left\lfloor \frac{1}{2} \log_2 \frac{14T}{K} \right\rfloor$ **do**

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

6: ***Arm Elimination***

7: For each cluster $s_k \in S_m$, delete arm $i \in s_k$ from B_m if

$$\hat{r}_i + \sqrt{\frac{\rho_a \log\left(\psi T \epsilon_m^2\right)}{2n_m}} < \max_{j \in s_k} \left\{ \hat{r}_j - \sqrt{\frac{\rho_a \log\left(\psi T \epsilon_m^2\right)}{2n_m}} \right\}$$

ClusUCB Algorithm II

Cluster Elimination

Delete cluster $s_k \in S_m$ and remove all arms $i \in s_k$ from B_m if

$$\max_{i \in s_k} \left\{ \hat{r}_i + \sqrt{\frac{\rho_s \log(\psi T \epsilon_m^2)}{2n_m}} \right\} < \max_{j \in B_m} \left\{ \hat{r}_j - \sqrt{\frac{\rho_s \log(\psi T \epsilon_m^2)}{2n_m}} \right\}.$$

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

Set $B_{m+1} := B_m$

Stop if $|B_m| = 1$ and pull $i \in B_m$ till T is reached.

end for

UCB1, MOSS, OCUCB, UCB-Imp, ClusUCB

Algorithm	Upper bound on Cumulative Regret
UCB1	$\min\left\{O\left(\frac{K\log T}{\Delta}\right), O\left(\sqrt{KT\log T}\right)\right\}$
MOSS	$\min\left\{O\left(\frac{K^2\log(T\Delta^2/K)}{\Delta}\right), O\left(\sqrt{KT}\right)\right\}$
UCB-Improved	$\min\left\{O\left(\frac{K\log(T\Delta^2)}{\Delta}\right), O\left(\sqrt{KT\log K}\right)\right\}$
ClusUCB	$\min\left\{O\left(\frac{K\log(T\Delta^2/\sqrt{\log K})}{\Delta}\right), O\left(\sqrt{KT\log K}\right)\right\}$
OCUCB	$\min\left\{O\left(\frac{K\log(T/H)}{\Delta}\right), O\left(\sqrt{KT}\right)\right\}$

Approach of EClusUCB

- One of the main problems of ClusUCB is that it's still a round based algorithm.

Approach of EClusUCB

- One of the main problems of ClusUCB is that it's still a round based algorithm.
- In every round it pulls all the arms equal number of times, which although is less compared to UCB-Improved but still we can be better.

Approach of EClusUCB

- One of the main problems of ClusUCB is that it's still a round based algorithm.
- In every round it pulls all the arms equal number of times, which although is less compared to UCB-Improved but still we can be better.
- One simple solution is to pull the arm with the highest UCB at every timestep. This is called optimistic greedy sampling for UCB-Imp (see [**?**]).

Approach of EClusUCB

- One of the main problems of ClusUCB is that it's still a round based algorithm.
- In every round it pulls all the arms equal number of times, which although is less compared to UCB-Improved but still we can be better.
- One simple solution is to pull the arm with the highest UCB at every timestep. This is called optimistic greedy sampling for UCB-Imp (see [**?**]).
- We introduce this in Efficient ClusUCB or EClusUCB.

Technical Details of EClusUCB

- EClusUCB has three exploration parameters ρ_a, ρ_s and ψ.

Technical Details of EClusUCB

- EClusUCB has three exploration parameters ρ_a, ρ_s and ψ.
- It also creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.

Technical Details of EClusUCB

- EClusUCB has three exploration parameters ρ_a, ρ_s and ψ.
- It also creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Divide the horizon into rounds and for each round allocate $|B_m| n_m$ pulls, where $n_m := \left\lceil \frac{2\log(\psi T \epsilon_m^2)}{\epsilon_m} \right\rceil$.

Technical Details of EClusUCB

- EClusUCB has three exploration parameters ρ_a, ρ_s and ψ.
- It also creates p fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Divide the horizon into rounds and for each round allocate $|B_m|n_m$ pulls, where $n_m := \left\lceil \frac{2\log(\psi T\epsilon_m^2)}{\epsilon_m} \right\rceil$.
- Within a round pull an arm that has the maximum UCB among all arms.

Technical Details of EClusUCB

- EClusUCB has three exploration parameters ρ_a, ρ_s and ψ.
- It also creates *p* fixed clusters (given as an input) and randomly assign arms into each of them such that each cluster contains equal number of arms.
- Divide the horizon into rounds and for each round allocate $|B_m|n_m$ pulls, where $n_m := \left\lceil \frac{2\log(\psi T\epsilon_m^2)}{\epsilon_m} \right\rceil$.
- Within a round pull an arm that has the maximum UCB among all arms.
- The arm and cluster elimination is done in same way.

EClusUCB Algorithm I

Input: Number of clusters p, time horizon T, exploration parameters ρ_a, ρ_s and ψ.
Initialization: Set $m := 0$, $B_0 := A$, $S_0 = S$, $\epsilon_0 := 1$,
$M = \left\lfloor \frac{1}{2}\log_2 \frac{14T}{K} \right\rfloor$, $n_0 = \left\lceil \frac{2\log(\psi T\epsilon_0^2)}{\epsilon_0} \right\rceil$ and $N_0 = Kn_0$.
Create a partition S_0 of the arms at random into p clusters of size up to $\ell = \left\lceil \frac{K}{p} \right\rceil$ each.
Pull each arm once
for $t = K+1, .., T$ **do**

Pull arm $i \in B_m$ such that $\operatorname{argmax}_{j \in B_m} \left\{ \hat{r}_j + \sqrt{\frac{\rho_s \log(\psi T\epsilon_m^2)}{2z_j}} \right\}$,
where z_j is the number of times arm j has been pulled

$t := t+1$

Arm Elimination

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

$$\hat{r}_i + \sqrt{\frac{\rho_a \log(\psi T \epsilon_m^2)}{2z_i}} < \max_{j \in s_k} \left\{ \hat{r}_j - \sqrt{\frac{\rho_a \log(\psi T \epsilon_m^2)}{2z_j}} \right\}$$

Cluster Elimination

Delete cluster $s_k \in S_m$ and remove all arms $i \in s_k$ from B_m if

$$\max_{i \in s_k} \left\{ \hat{r}_i + \sqrt{\frac{\rho_s \log(\psi T \epsilon_m^2)}{2z_i}} \right\} < \max_{j \in B_m} \left\{ \hat{r}_j - \sqrt{\frac{\rho_s \log(\psi T \epsilon_m^2)}{2z_j}} \right\}.$$

if $t \geq N_m$ and $m \leq M$ **then**

$\epsilon_{m+1} := \frac{\epsilon_m}{2}$

$B_{m+1} := B_m$

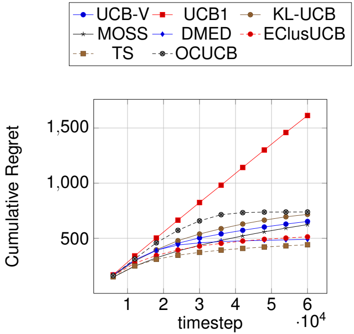
$n_{m+1} := \left\lceil \frac{2 \log(\psi T \epsilon_{m+1}^2)}{\epsilon_{m+1}} \right\rceil$

$N_{m+1} := t + |B_{m+1}| n_{m+1}$
$m := m + 1$
Stop if $|B_m| = 1$ and pull $i \in B_m$ till T is reached.
end if
end for

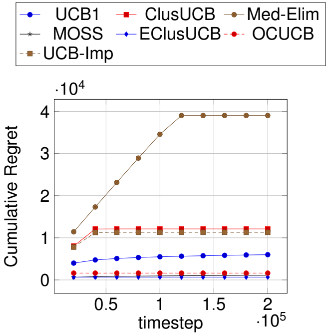
UCB1, MOSS, OCUCB, UCB-Imp, EClusUCB

Algorithm	Upper bound on Cumulative Regret
UCB1	$\min\left\{O\left(\frac{K\log T}{\Delta}\right), O\left(\sqrt{KT\log T}\right)\right\}$
MOSS	$\min\left\{O\left(\frac{K^2\log(T\Delta^2/K)}{\Delta}\right), O\left(\sqrt{KT}\right)\right\}$
UCB-Improved	$\min\left\{O\left(\frac{K\log(T\Delta^2)}{\Delta}\right), O\left(\sqrt{KT\log K}\right)\right\}$
EClusUCB	$\min\left\{O\left(\frac{K\log(T\Delta^2/\sqrt{\log K})}{\Delta}\right), O\left(\sqrt{KT\log K}\right)\right\}$
OCUCB	$\min\left\{O\left(\frac{K\log(T/H)}{\Delta}\right), O\left(\sqrt{KT}\right)\right\}$

Finally, experiment!!!



(a) Experiment 1: 20 Bernoulli-distributed arms with $r_{i_{i\neq *}} = 0.07$ and $r^* = 0.1$.



(b) Experiment 2: 100 Gaussian-distributed arms with $r_{i_{i\neq *:1-33}} = 0.1$, $r_{i_{i\neq *:34-99}} = 0.6$ and $r^*_{i=100} = 0.9$.

Figure: Cumulative regret for various bandit algorithms on two stochastic K-armed bandit environments.

Conclusions

- We introduced the ClusUCB and EClusUCB algorithms which achieves a better gap-dependent regret bound than UCB1, UCB-Imp and Moss.

Conclusions

- We introduced the ClusUCB and EClusUCB algorithms which achieves a better gap-dependent regret bound than UCB1, UCB-Imp and Moss.
- Empirically EClusUCB beats most of the UCB variants.

Conclusions

- We introduced the ClusUCB and EClusUCB algorithms which achieves a better gap-dependent regret bound than UCB1, UCB-Imp and Moss.
- Empirically EClusUCB beats most of the UCB variants.
- This are the first algorithms which employ clustering in SMABs.

Conclusions

- We introduced the ClusUCB and EClusUCB algorithms which achieves a better gap-dependent regret bound than UCB1, UCB-Imp and Moss.
- Empirically EClusUCB beats most of the UCB variants.
- This are the first algorithms which employ clustering in SMABs.
- We prove theoretically that for arm elimination algorithms, Clustering is indeed beneficial.

Conclusions

- We introduced the ClusUCB and EClusUCB algorithms which achieves a better gap-dependent regret bound than UCB1, UCB-Imp and Moss.
- Empirically EClusUCB beats most of the UCB variants.
- This are the first algorithms which employ clustering in SMABs.
- We prove theoretically that for arm elimination algorithms, Clustering is indeed beneficial.
- I am greatly thankful to Dr. L.A. Prashanth (University of Maryland) for guiding me through all the proofs and correcting several aspects of this work.

References I

Thank You