CS 769 Project Presentation
Comparison of Multi-Armed Bandits Policies

Presented by -

Prishat Bachhar - 213050078
Subir Kumar Parida - 21v980019
Yashwant Raghuwanshi - 213050002

Table of Contents

Introduction to Online Convex Optimization

K: convex decision set, and **F**: set of convex functions.

A repeated game of T rounds between a player and an adversary, at each round $\mathbf{t} \in \mathbf{T}$**,** then the OCO framework,

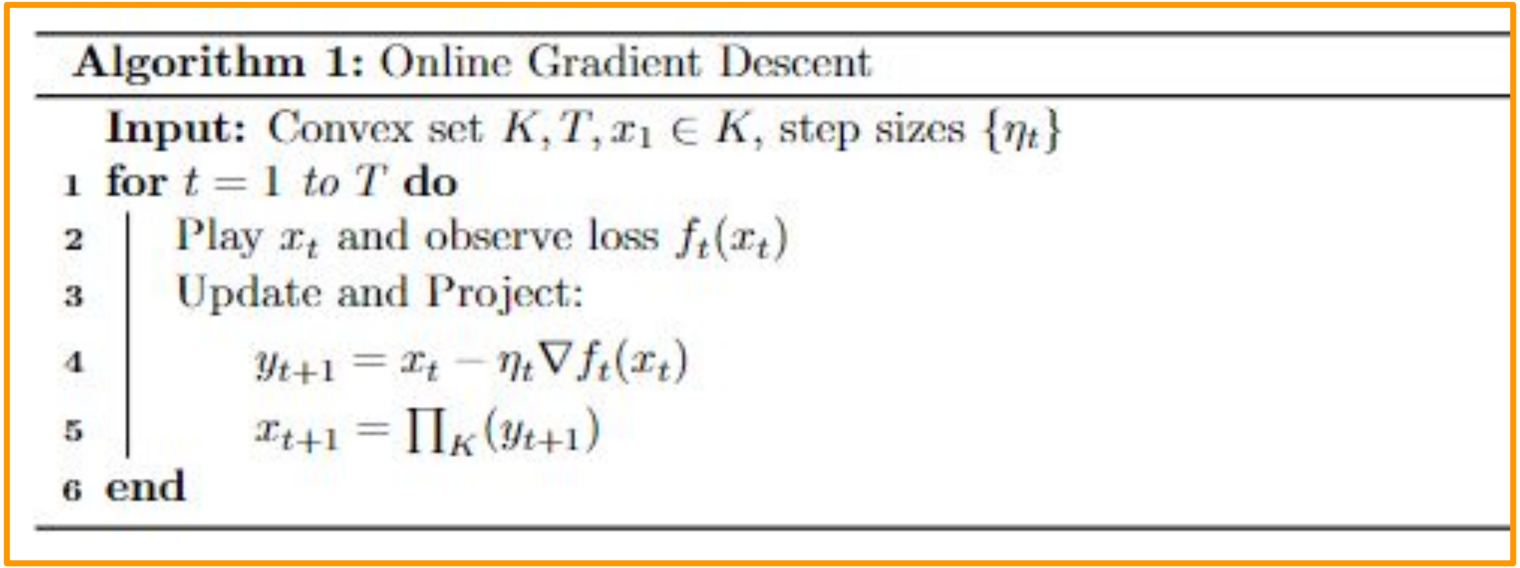
- player chooses a point $\mathbf{x}_t \in \mathbf{K}$
- adversary independently chooses a loss function $\mathbf{f}_t \in \mathbf{F}$
- player suffers a loss $\mathbf{f}_t\,(\mathbf{x}_t)$ and receives a feedback $\mathbf{F}_t$

The regret of the setting can be defined as:

$$\boxed{Regret_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x^* \in K} \sum_{t=1}^{T} f_t(x^*)}$$

Objective: To minimize the expected regret

Online Gradient Descent

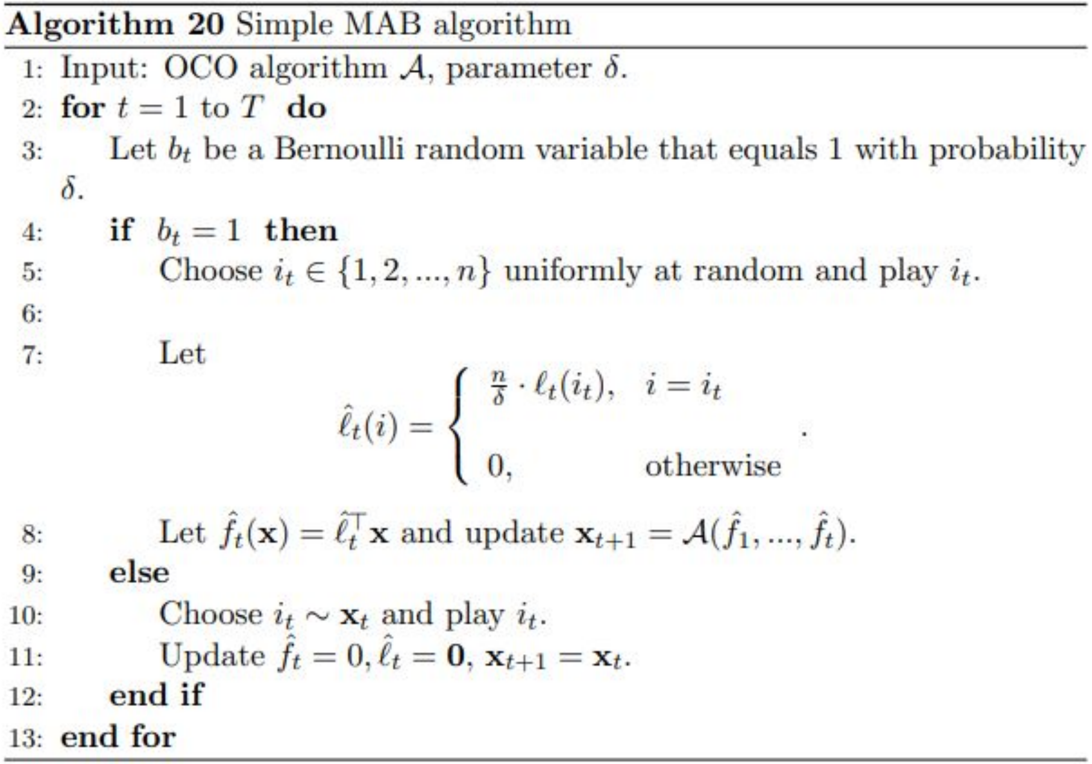


```
Algorithm 1: Online Gradient Descent
    Input: Convex set $K, T, x_1 \in K$, step sizes $\{\eta_t\}$
for $t = 1$ to $T$ do
    Play $x_t$ and observe loss $f_t(x_t)$
    Update and Project:
        $y_{t+1} = x_t - \eta_t \nabla f_t(x_t)$
        $x_{t+1} = \prod_K(y_{t+1})$
end
```

The Exploration-Exploitation Dilemma

- **Exploration** - improve knowledge for long-term benefit
- **Exploitation** - exploit knowledge for short-term benefit

Multi Armed Bandits (MAB)



Algorithm 20 Simple MAB algorithm

```
Input: OCO algorithm $\mathcal{A}$, parameter $\delta$.
for $t = 1$ to $T$ do
    Let $b_t$ be a Bernoulli random variable that equals 1 with probability $\delta$.
    if $b_t = 1$ then
        Choose $i_t \in \{1, 2, ..., n\}$ uniformly at random and play $i_t$.

        Let
            $$\hat{\ell}_t(i) = \begin{cases} \frac{n}{\delta} \cdot \ell_t(i_t), & i = i_t \\ 0, & \text{otherwise} \end{cases}.$$
        Let $\hat{f}_t(\mathbf{x}) = \hat{\ell}_t^\top \mathbf{x}$ and update $\mathbf{x}_{t+1} = \mathcal{A}(\hat{f}_1, ..., \hat{f}_t)$.
    else
        Choose $i_t \sim \mathbf{x}_t$ and play $i_t$.
        Update $\hat{f}_t = 0, \hat{\ell}_t = \mathbf{0}$, $\mathbf{x}_{t+1} = \mathbf{x}_t$.
    end if
end for
```

- In this setting the player only knows the loss of the action chosen and not the entire loss gradient.

Other Strategies for MAB

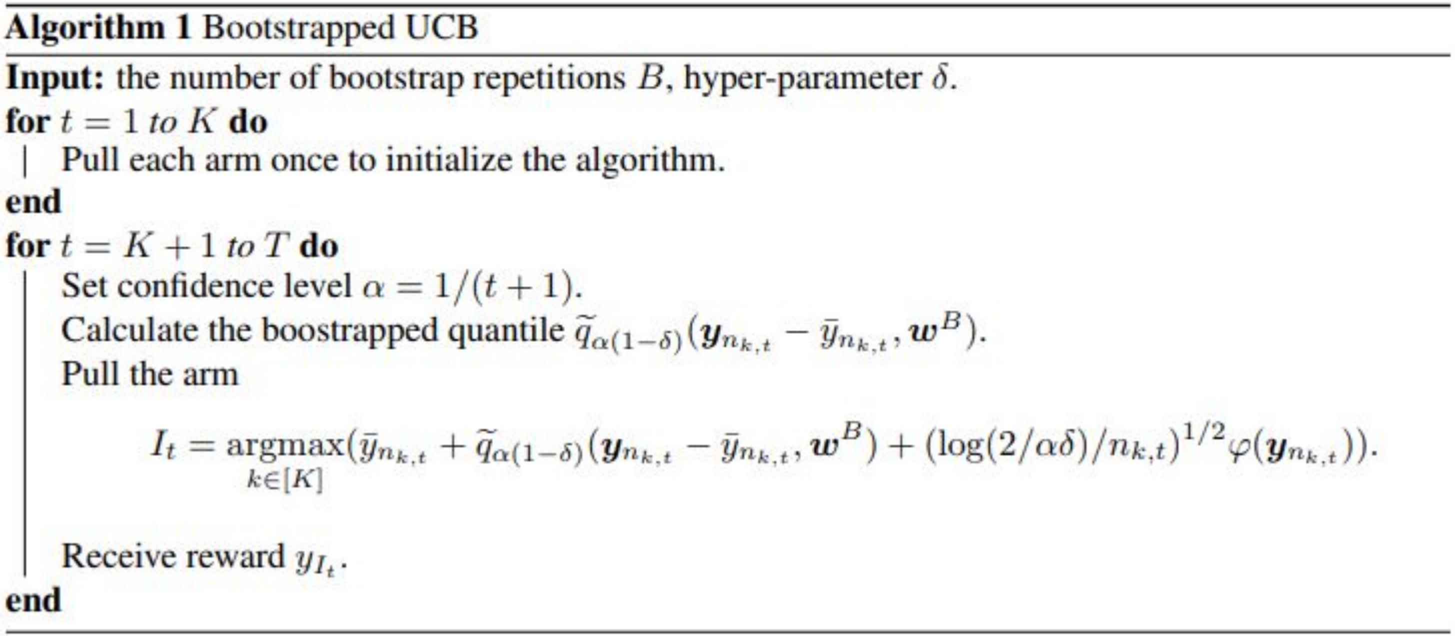
- Epsilon Decay Algorithm
    - Keep reducing the value of epsilon $\frac{1}{(1+\frac{t}{no.of arms})}$.
- UCB Algorithm
    - Reduce the uncertainty associated with an action
    - Done by sampling more number of times

$$Q(a) + \sqrt{\frac{2\ln t}{N_t(a)}}$$

- Thompson Sampling
    - Sample the next action using a Beta distribution of the number of times a reward was achieved and the number of times a loss was achieved by choosing that action.

$$Beta(\theta;\alpha;\beta) = \frac{1}{C}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

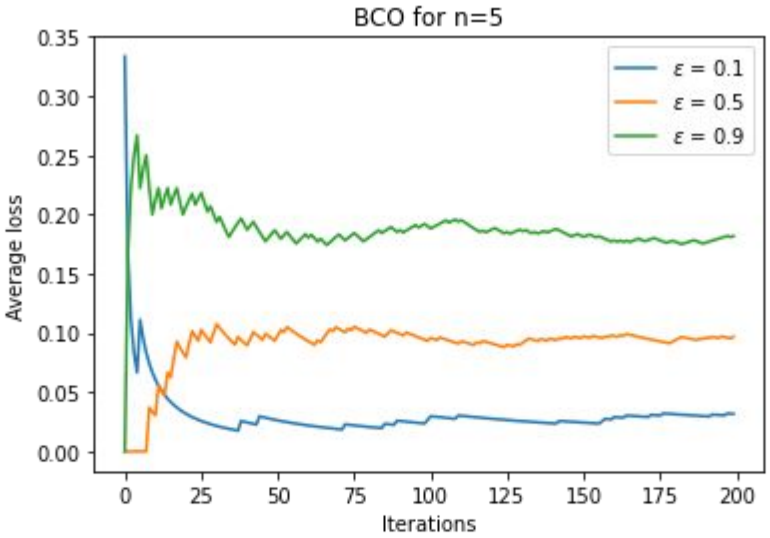
Bootstrapped UCB



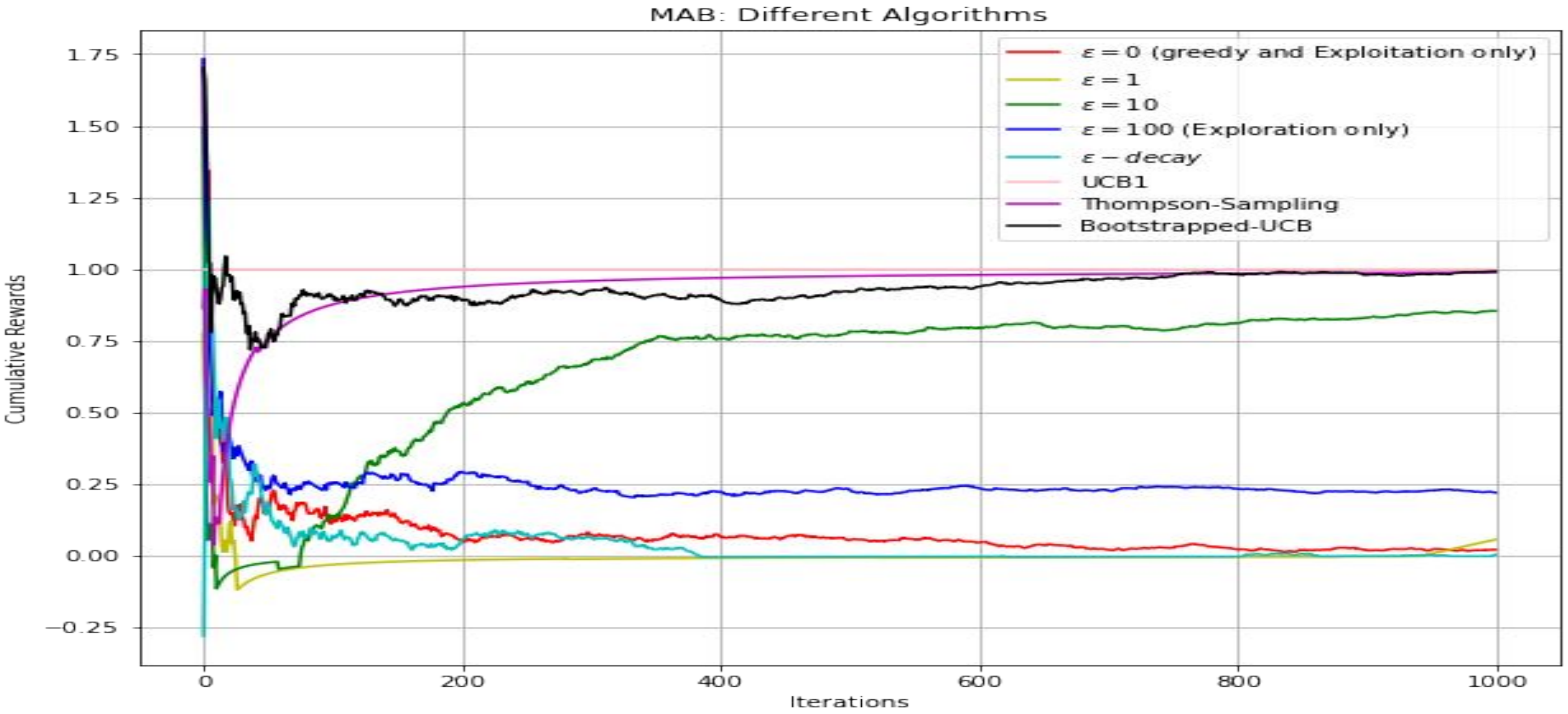
```
Algorithm 1 Bootstrapped UCB
Input: the number of bootstrap repetitions $B$, hyper-parameter $\delta$.
for $t = 1$ to $K$ do
    Pull each arm once to initialize the algorithm.
end
for $t = K + 1$ to $T$ do
    Set confidence level $\alpha = 1/(t + 1)$.
    Calculate the boostrapped quantile $\widetilde{q}_{\alpha(1-\delta)}(\boldsymbol{y}_{n_{k,t}} - \bar{y}_{n_{k,t}}, \boldsymbol{w}^B)$.
    Pull the arm
        $I_t = \operatorname{argmax}_{k \in [K]}(\bar{y}_{n_{k,t}} + \widetilde{q}_{\alpha(1-\delta)}(\boldsymbol{y}_{n_{k,t}} - \bar{y}_{n_{k,t}}, \boldsymbol{w}^B) + (\log(2/\alpha\delta)/n_{k,t})^{1/2}\varphi(\boldsymbol{y}_{n_{k,t}}))$.
    Receive reward $y_{I_t}$.
end
```

Results

- Higher exploration rate leads to higher average loss over time
- This is due to the fact that the algorithm explores even when it is almost sure of the best action
- The other strategies and their convergence is given in the following slides.



Results



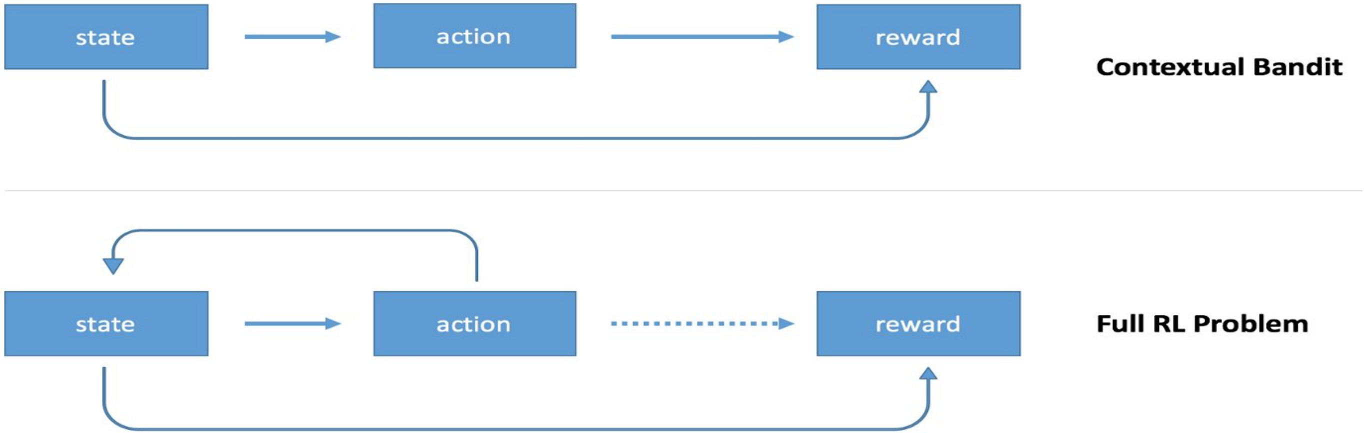
Convergence rates of different algorithms

Critical Analysis for MAB

- In the context of MAB, a higher probability of exploration leads to more regret as the algorithm is forced to explore bad choices.
- **ε-greedy family**: The algorithms are not at all concerned towards obtaining the true mean of an arm for exploitation
- **UCB, Thompson**: The algorithms tries to reduce the uncertainty about the true reward of an arm
- **UCB**:
    - UCB eliminates the randomness of the ε-greedy family algorithms
    - The algorithm gives the benefit of doubt to a less explored arm at a certain time t, and explores it to identify the true distribution/reward
- **Thompson**:
    - Tries to select the arm which has previously given more number of high rewards.
    - It tries to understand the true distribution of the rewards associated with the arms.

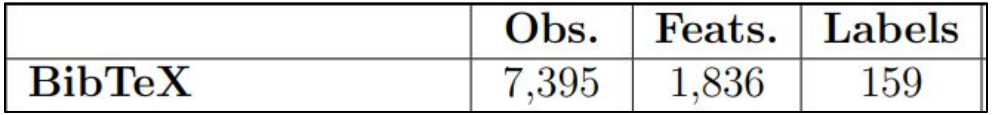
Contextual MAB

- It is an extension of multi-armed bandits.
- The algorithm observes a context, makes a decision, choosing one action from a number of alternative actions, and observes an outcome of that decision.



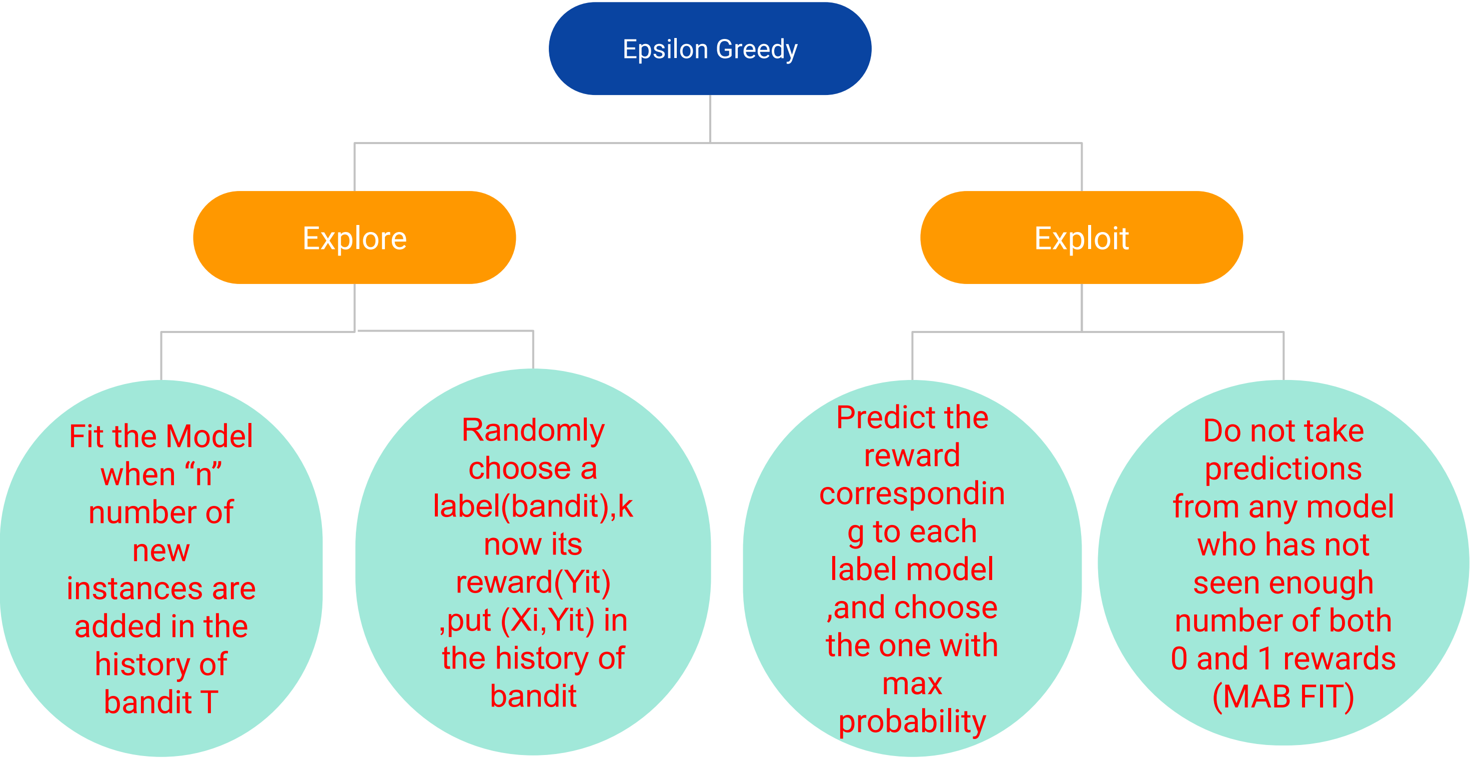
Bibtex Dataset:

- It is a Multilabel text classification for automated tag suggestion:
    - containing tags that people have assigned to different papers
    - the goal being to learn to suggest tags based on features from the papers
- dataset link



	Obs.	Feats.	Labels
BibTeX	7,395	1,836	159

WorkFlow:



Epsilon Greedy

Explore

Exploit

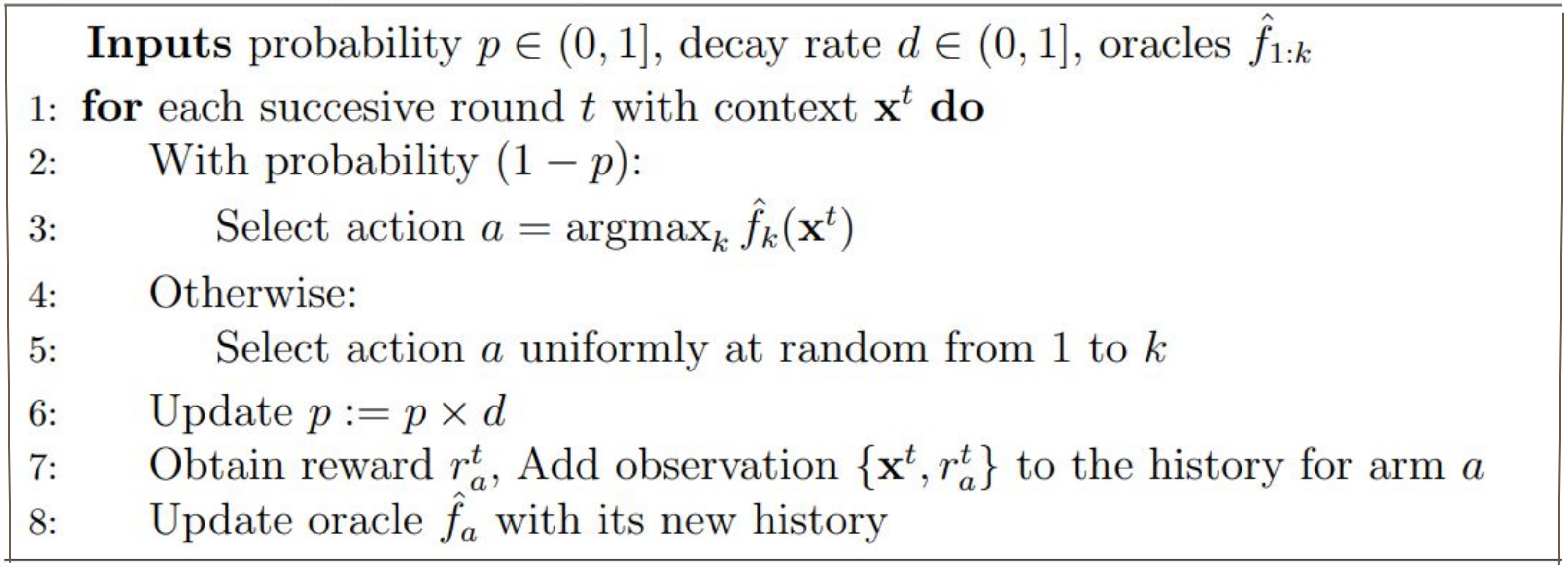
Fit the Model when “n” number of new instances are added in the history of bandit T

Randomly choose a label(bandit),k now its reward(Yit) ,put (Xi,Yit) in the history of bandit

Predict the reward corresponding to each label model ,and choose the one with max probability

Do not take predictions from any model who has not seen enough number of both 0 and 1 rewards (MAB FIT)

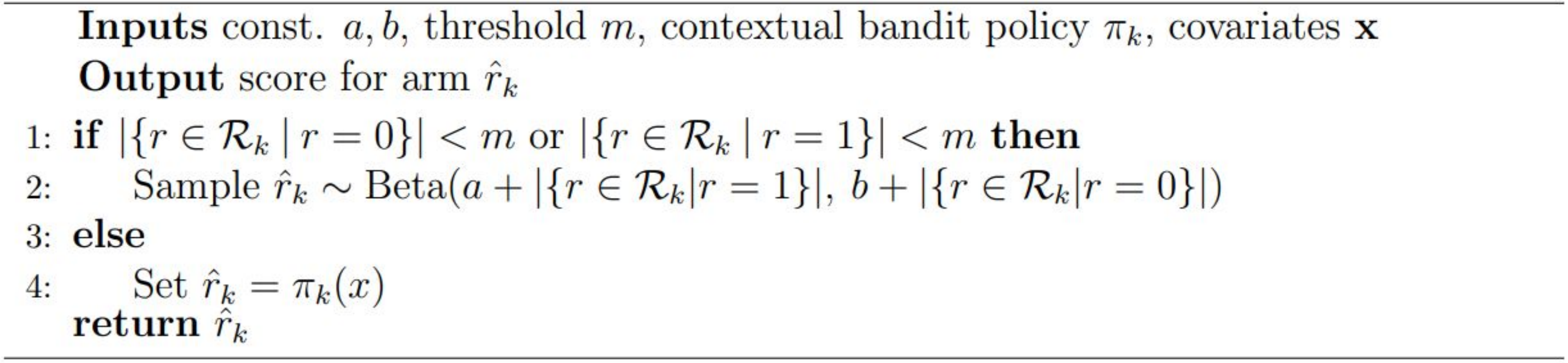
Epsilon-Greedy for Contextual Bandits:



```
**Inputs** probability $p \in (0, 1]$, decay rate $d \in (0, 1]$, oracles $\hat{f}_{1:k}$
**for** each succesive round $t$ with context $\mathbf{x}^t$ **do**
    With probability $(1 - p)$:
        Select action $a = \text{argmax}_k \hat{f}_k(\mathbf{x}^t)$
    Otherwise:
        Select action $a$ uniformly at random from 1 to $k$
    Update $p := p \times d$
    Obtain reward $r_a^t$, Add observation $\{\mathbf{x}^t, r_a^t\}$ to the history for arm $a$
    Update oracle $\hat{f}_a$ with its new history
```

Implementation:

- Firstly,the bibtex dataset is transformed into binary classification data for each label corresponding to each training instance.
- **MAB-FIRST:**

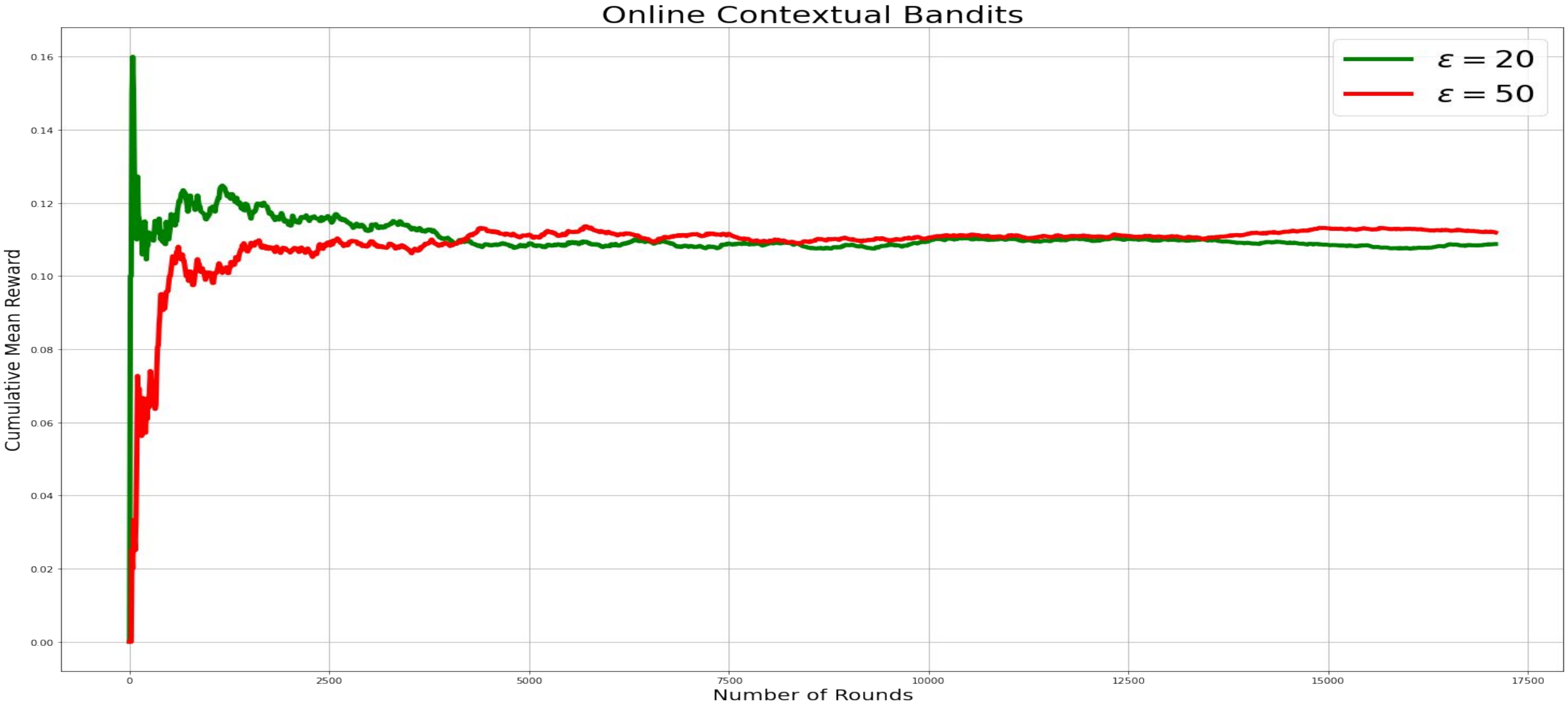


```
Inputs const. $a, b$, threshold $m$, contextual bandit policy $\pi_k$, covariates $\mathbf{x}$
Output score for arm $\hat{r}_k$
if $|\{r \in \mathcal{R}_k \mid r = 0\}| < m$ or $|\{r \in \mathcal{R}_k \mid r = 1\}| < m$ then
    Sample $\hat{r}_k \sim \text{Beta}(a + |\{r \in \mathcal{R}_k | r = 1\}|,\ b + |\{r \in \mathcal{R}_k | r = 0\}|)$
else
    Set $\hat{r}_k = \pi_k(x)$
return $\hat{r}_k$
```

Parameters being Hypertuned:

- a and b values for beta distribution (a=3,b=7)
- Subset of bandits (labels) on which models are being tuned due to sparsity of data(opt subset length =10)
- Refit the corresponding bandit depending on the number of new training instances being added to its history, a tradeoff against computational power utilized (opt. value=5)

Results


Online Contextual Bandits
Cumulative Mean Reward
Number of Rounds
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0
2500
5000
7500
10000
12500
15000
17500
$\varepsilon = 20$
$\varepsilon = 50$

Github Link

https://github.com/subirkumarparida/CS769-Project.git

Thank you