Rankings

Jayavarma,Muhilan,Mohan Kumar

Indian Institute of Technology
Madras

April 2023

Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons

The Abstract Coin Problem

- The abstract coin problem involves identifying the k most biased coins from a set of n coins with unknown biases.
- It is a generalization of several well-studied problems such as identifying the k best arms in a stochastic multi-armed bandit and top-k ranking from pairwise comparisons.
- The problem assumes that the learner has access to a feedback generation mechanism that provides noisy feedback about the outcomes of coin tosses.
- The goal of the learner is to identify the k most biased coins with high probability using a limited number of rounds of interaction with the feedback generation mechanism.

Active Learning vs. PAC Learning

PAC Learning:

- A learner is given a sample of labeled data drawn independently at random from an unknown distribution.
- The goal is to output a hypothesis that has low generalization error.
- The learner is a passive observer who is given a collection of randomly sampled observations from which to learn.

Active Models:

- Active learning involves the learner actively querying an oracle, or some other information source, to obtain additional information that can be used to improve the model.
- In other words, active learning algorithms choose which samples to label and use them to improve the model's accuracy.

Motivation for Limited Round of Adaptivity Models

- Limited round of adaptivity models have been developed in various fields, including computer science, economics, and clinical trials.
- The motivation behind these models is to strike a balance between exploration and exploitation.
- In many practical scenarios, we need to optimize a process with limited resources, and we cannot afford to explore indefinitely.
- Therefore, the limited round of adaptivity models aim to find the best solution within a limited number of rounds, using the available resources optimally.
- In clinical trials, these models are used to optimize the allocation of treatments to patients, while minimizing the risk of harmful effects and maximizing the likelihood of successful outcomes.

The Setup

- We will be using AGGRESSIVE-ELIMINATION Algorithm to solve this problem.
- The learner can query any number of data points in each round.
- All data points to be queried in a given round must be submitted simultaneously.
- The learner can only use the feedback received in previous rounds to decide which data points to query in the current round.

Notation

- For any integer $a \geq 1$, $[a] := 1, 2, \ldots, a$. For a (multi-)set of numbers $a_1, \ldots, a_n$, we define $a_{[i]}$ as the i-th largest value in this set (ties are broken arbitrarily).
- For any integer $r \geq 0$, $i\log^{(r)}(a)$ denotes the iterated logarithms of order r, i.e., $i\log^{(r)}(a) = \max\{\log(i\log^{(r-1)}(a)); 1\}$.
- The iterated logarithm of n, written $\log^* n$ (usually read "log star"), is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1.
- $\log^*(n) = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \log^*(\log n) & \text{if } n > 1 \end{cases}$

Example

The iterated logarithm of $n = 1000$ with base 2 can be calculated as follows:

$$\log_2 n = \log_2 1000 = 9.966$$

Since the result is greater than 1, we apply the logarithm function again:

$$\log_2 \log_2 1000 = 3.322$$

Iterate till n≤ 1 :

$$\log_2 \log_2 \log_2 1000 = 1.737$$

$$\log_2 \log_2 \log_2 \log_2 1000 = 0$$

Now the result is less than or equal to 1, so we stop applying the logarithm function. Therefore, the iterated logarithm of $n = 1000$ with base 2 is $\log^*(1000) = 4$.

Algorithm

Estimation Phase:

- We toss each coin $O(\frac{1}{\Delta^2}.(\text{ilog}^{(r)}(m) + \log(k/\delta)))$ times and estimate the bias of each coin based on the number of heads obtained.
- The formula for the number of tosses is chosen such that with high probability, the estimate for the bias of each coin has an error of at most $\Delta/2$.
- For example, we flip the coin a large number of times, say 1000 times, and count the number of heads we get. If we get 600 heads, we estimate that the probability of flipping a head is $p \approx 0.6$.

What happens when $\Delta = 0$?

- If the biases of some coins are the same, it may take longer to eliminate them since they will have the same number of heads and tails in the first few rounds.
- This is because the algorithm relies on a gap between the biases of the coins to eliminate them in each round, and if the gap is not large enough, then the algorithm may not be able to eliminate all but one coin.
- This can lead to an increased number of rounds and more queries to identify the biased coin.
- In extreme cases where all coins have the same bias, the algorithm may fail to identify the biased coin altogether.
- To fix this issue, we can modify the algorithm that incorporates tie-breaking rules to break the symmetry among the coins.
- These rules can be based on various criteria such as the lexicographic order of the coin IDs, or the order in which the coins were first queried.

Algorithm Continued

Elimination Phase:

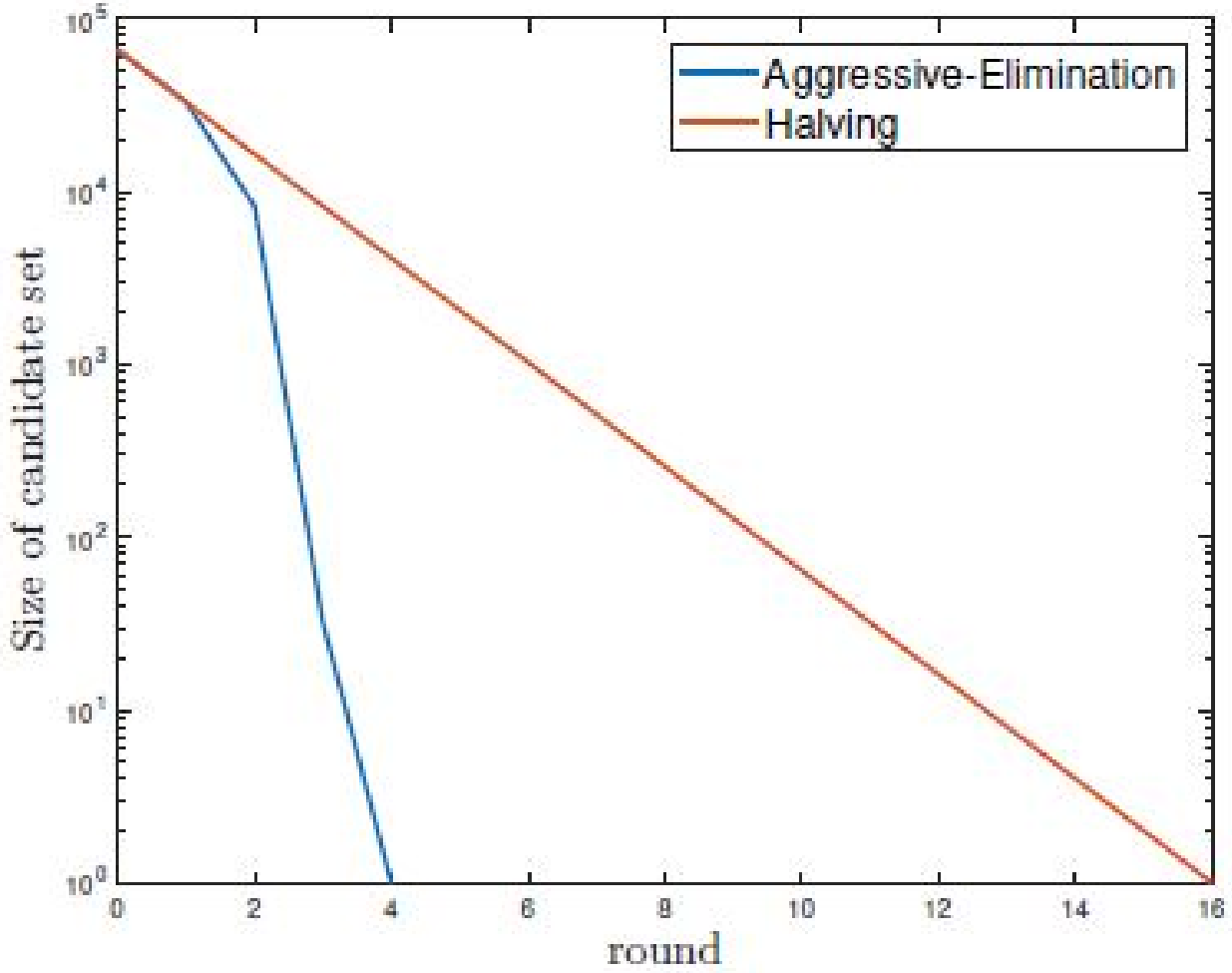
- Let $S^{'}$ be the set of $O(\frac{m}{i \log^{(r-1)}(m)})$ coins with the largest estimated biases. Recursively solve the problem for the set $S^{'}$ in the remaining $r-1$ rounds.
- The elimination phase becomes more aggressive over the rounds, reducing the number of remaining coins to an exponentially smaller fraction of the current coins.
- This allows for a larger budget allocation for the estimation phase in the next round. This process of aggressive elimination is repeated until only the top k coins remain.

Algorithm - Pseudo Code

Algorithm 1 AGRESSIVE-ELIMINATION$(S_r, k, r, \delta, \Delta)$

Input: set $S_r \subseteq [n]$ of coins, number of desired top items k, number of rounds r, confidence parameter $\delta \in (0,1)$, and lower bound on gap parameter $\Delta \leq \Delta_k$
Let $m = m_r = |S_r|$ and $t_r := \frac{2}{\Delta^2} \cdot \left(\text{ilog}^{(r)}(m) + \log(8k/\delta)\right)$.
Toss each coin $i \in S_r$ for t_r times.
For each $i \in S_r$, define $\widehat{p}_i$ as the fraction of times coin i turns up heads.
Sort the coins in S_r in a decreasing order of $\widehat{p}$-values.
if $r = 1$ **then**
    Return: the set of k most biased coins (according to $\widehat{p}$-values).
else
    Let $m_{r-1} := k + \frac{m}{\text{ilog}^{(r-1)}(m)}$ and S_{r-1} be the set of m_{r-1} most biased coins according to $\widehat{p}$.
end if
if $m_{r-1} \leq 2k$ **then**
    Return: AGRESSIVE-ELIMINATION$(S_{r-1}, k, 1, \delta/2, \Delta)$.
else
    Return: AGRESSIVE-ELIMINATION$(S_{r-1}, k, r-1, \delta/2, \Delta)$.
end if

Experiments



Experiments

```python
#Experiment
# Generate 100 biased coins with biases uniformly distributed between 0.4 and 0.6
biases = np.random.uniform(0.4, 0.6, 100)
coins = [(bias, i) for i, bias in enumerate(biases)]
# Find top 5 biased coins using AGGRESSIVE-ELIMINATION algorithm with r=7, delta=0.1, and Delta=0.1
top_coins = AGGRESSIVE_ELIMINATION(coins, 5, 7, 0.1, 0.1)
print(top_coins)
```

```
[15, 47, 74, 94, 80]
```

Problem Statement

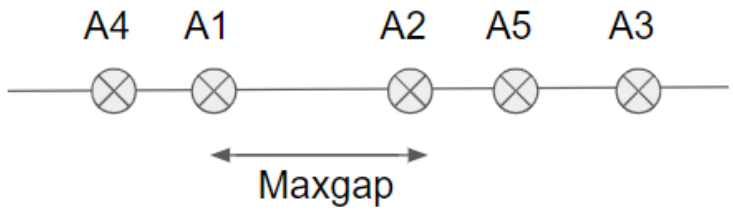
MaxGap Bandits

- Problem of adaptive sampling to find the largest gap in the true means of K unknown iids.

Problem Statement

MaxGap Bandits

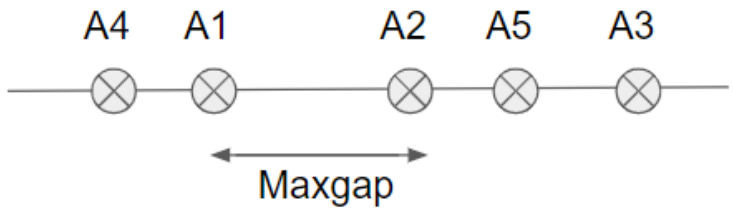
- Problem of adaptive sampling to find the largest gap in the true means of K unknown iids.
- The aim is to identify the largest gap between *adjacent arm* means.



Problem Statement

MaxGap Bandits

- Problem of adaptive sampling to find the largest gap in the true means of K unknown iids.
- The aim is to identify the largest gap between *adjacent arm* means.
- Partial ordering into groups/clusters of distributions with similar means.



Problem Statement

Notations

- K distributions $=>$ K arms
- Mean of Arm i $=>$ μ_i
- $\mu_{(l)}$ is the lth largest mean
- $\mu_{(K)} \leq ... \leq \mu_{(2)} \leq \mu_{(1)}$

Problem Statement

Notations

- K distributions $=>$ K arms
- Mean of Arm i $=>$ μ_i
- $\mu_{(l)}$ is the lth largest mean
- $\mu_{(K)} \leq ... \leq \mu_{(2)} \leq \mu_{(1)}$
- Gap of Arm i is defined as:

$$\Delta_i = \max\{\mu_{(l)} - \mu_{(l+1)}, \mu_{(l-1)} - \mu_{(l)}\} \tag{1}$$

where $\mu_i = \mu_{(l)}$

Problem Statement

Objective

- Find Δ_{max}, maximum over Δ_i for all i
- $m = \arg\max_{j \in [K-1]} \{\mu_{(j)} - \mu_{(j+1)}\}$

Problem Statement

Objective

- Find Δ_{max}, maximum over Δ_i for all i
- $m = \arg\max_{j \in [K-1]} \{\mu_{(j)} - \mu_{(j+1)}\}$
- Cluster 1: $\{(1), (2), ..., (m)\}$
  Cluster 2: $\{(m+1), (2), ..., (K)\}$

Algorithms

- A solution to the MaxGap-bandit problem is an algorithm which given a $\delta > 0$, samples the arms and upon stopping partitions K arms into the expected clusters with a probability greater than $1 - \delta$

Maxgap Elimination

1. $A = [K]$
2. In each timestep t,
   - Pull arms in A
   - Compute the sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A (Hoeffding's inequality)

Algorithms

- A solution to the MaxGap-bandit problem is an algorithm which given a $\delta > 0$, samples the arms and upon stopping partitions K arms into the expected clusters with a probability greater than $1 - \delta$

Maxgap Elimination

1. $A = [K]$
2. In each timestep t,
   - Pull arms in A
   - Compute the sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A (Hoeffding's inequality)
   - Compute *gap upperbounds* $U\Delta(t)$ for each arm in A
   - Compute $L\Delta_{max}(t)$

Algorithms

- A solution to the MaxGap-bandit problem is an algorithm which given a $\delta > 0$, samples the arms and upon stopping partitions K arms into the expected clusters with a probability greater than $1 - \delta$

Maxgap Elimination

1. $A = [K]$
2. In each timestep t,
   - Pull arms in A
   - Compute the sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A (Hoeffding's inequality)
   - Compute *gap upperbounds* $U\Delta(t)$ for each arm in A
   - Compute $L\Delta_{max}(t)$
   - Eliminate arms whose $U\Delta(t) < L\Delta_{max}(t)$,from A
   - Stop if $|A| = 2$. Else repeat.

Algorithms

Maxgap UCB

1. $A = [K]$
2. In each timestep t,
   - Pull and update sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A
   - Compute *gap upperbounds* $U\Delta(t)$ for all arms (i.e: $[K]$)

Algorithms

Maxgap UCB

1. $A = [K]$
2. In each timestep t,
   - Pull and update sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A
   - Compute *gap upperbounds* $U\Delta(t)$ for all arms (i.e: $[K]$)
   - Set A=$\{$set of arms whose gap upperbounds are maximum$\}$
   - Stop if for some $(i,j) \in [K], \{T_i(t) + T_j(t)\} \geq c.\sum_{a \notin \{i,j\}} T_a(t)$. Else repeat.

[2] Jamieson et al., *lil' UCB : An Optimal Exploration Algorithm for Multi-Armed Bandits*

Algorithms

Maxgap Top2 UCB

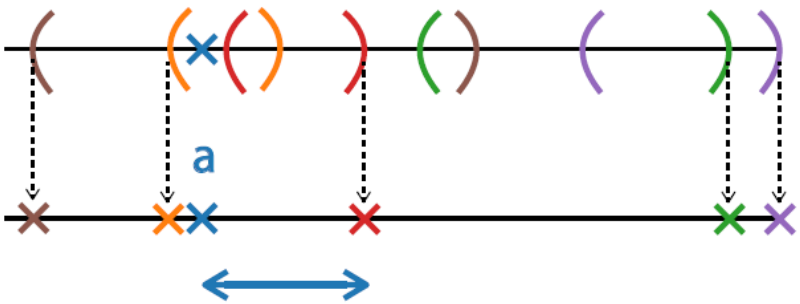
1. $A1 \cup A2 = [K]$
2. In each timestep t,
   - Pull and update sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A
   - Compute *gap upperbounds* $U\Delta(t)$ for all arms (i.e: $[K]$)
   - Compute $L\Delta_{max}(t)$

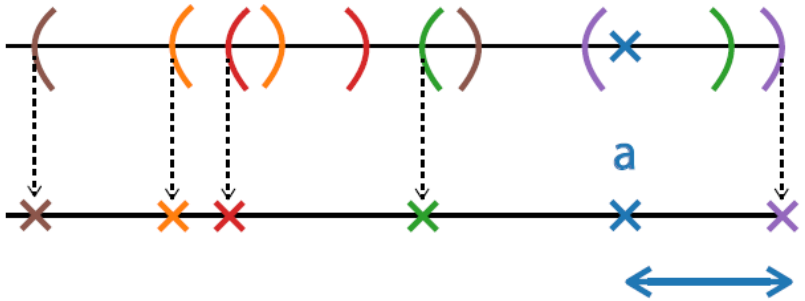
Algorithms

Maxgap Top2 UCB

1. $A1 \cup A2 = [K]$
2. In each timestep t,
   - Pull and update sample means for arms in A
   - Compute the confidence interval $[l_a(t), r_a(t)]$ for arms in A
   - Compute *gap upperbounds* $U\Delta(t)$ for all arms (i.e: $[K]$)
   - Compute $L\Delta_{max}(t)$
   - Set A1=$\{$set of arms whose gap upperbounds are maximum$\}$
   - Set A2=$\{$set of arms whose gap upperbounds are the second maximum$\}$
   - Stop if $U\Delta_{2^{nd}}(t) < L\Delta_{max}(t)$. Else repeat.

Key Ideas





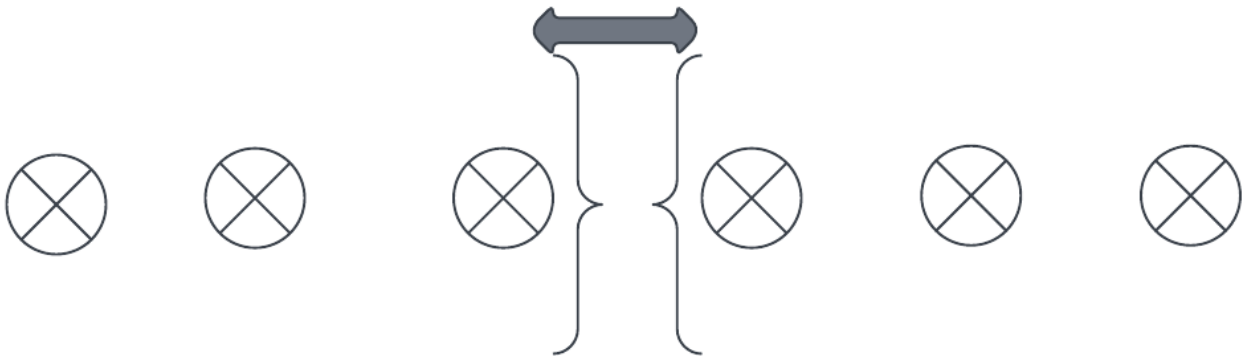
Gap Upperbounds $U\Delta_i(t)$

- $U\Delta_i(t) = \max\{U\Delta_i^r(t), U\Delta_i^l(t)\}$

$$G_a^r(x,t) = \begin{cases} \min_{\{j:l_j(t)>x\}} r_j(t) - x & \{j : l_j(t) > x\} \neq \emptyset \\ max_{j \neq a} r_j(t) - x & otherwise \end{cases}$$

[1] Katariya, Tripathy, and Nowak, *MaxGap Bandit: Adaptive Algorithms for Approximate Ranking*

Key Ideas



$L\Delta_{max}(t)$

$$\max_{k \in [K-1]} \left(\min_{a \in \{(1)_t,(2)_t,..,(k)_t\}} l_a(t) - \max_{a \in \{(k+1)_t,(2)_t,..,(K)_t\}} r_a(t) \right) \quad (2)$$

Sample Complexity Bound

- With probability $1 - \delta$, *MaxGap Elimination*, *MaxGap UCB* and *MaxGap Top2UCB* algorithms cluster the arms according to the maximum gap.

Sample Complexity Bound

- With probability $1-\delta$, *MaxGap Elimination*, *MaxGap UCB* and *MaxGap Top2UCB* algorithms cluster the arms according to the maximum gap.
- With probability $1-\delta$, the sample complexity of *MaxGap Elimination* and *MaxGap UCB* is bounded by,

$$O\left(\sum_{a\in[K]/\{(m),(m+1)\}}\frac{\log\left(K/\gamma_a\right)}{\gamma_a^2}\right). \tag{3}$$

Sample Complexity Bound

- With probability $1-\delta$, *MaxGap Elimination*, *MaxGap UCB* and *MaxGap Top2UCB* algorithms cluster the arms according to the maximum gap.
- With probability $1-\delta$, the sample complexity of *MaxGap Elimination* and *MaxGap UCB* is bounded by,

$$O\left(\sum_{a\in[K]/\{(m),(m+1)\}} \frac{\log\left(K/\gamma_a\right)}{\gamma_a^2}\right). \tag{3}$$

- γ_a acts as *hardness parameter* for a sub-optimal arm a; It depends on the gaps and the difference between gaps and the optimal/largest gap.

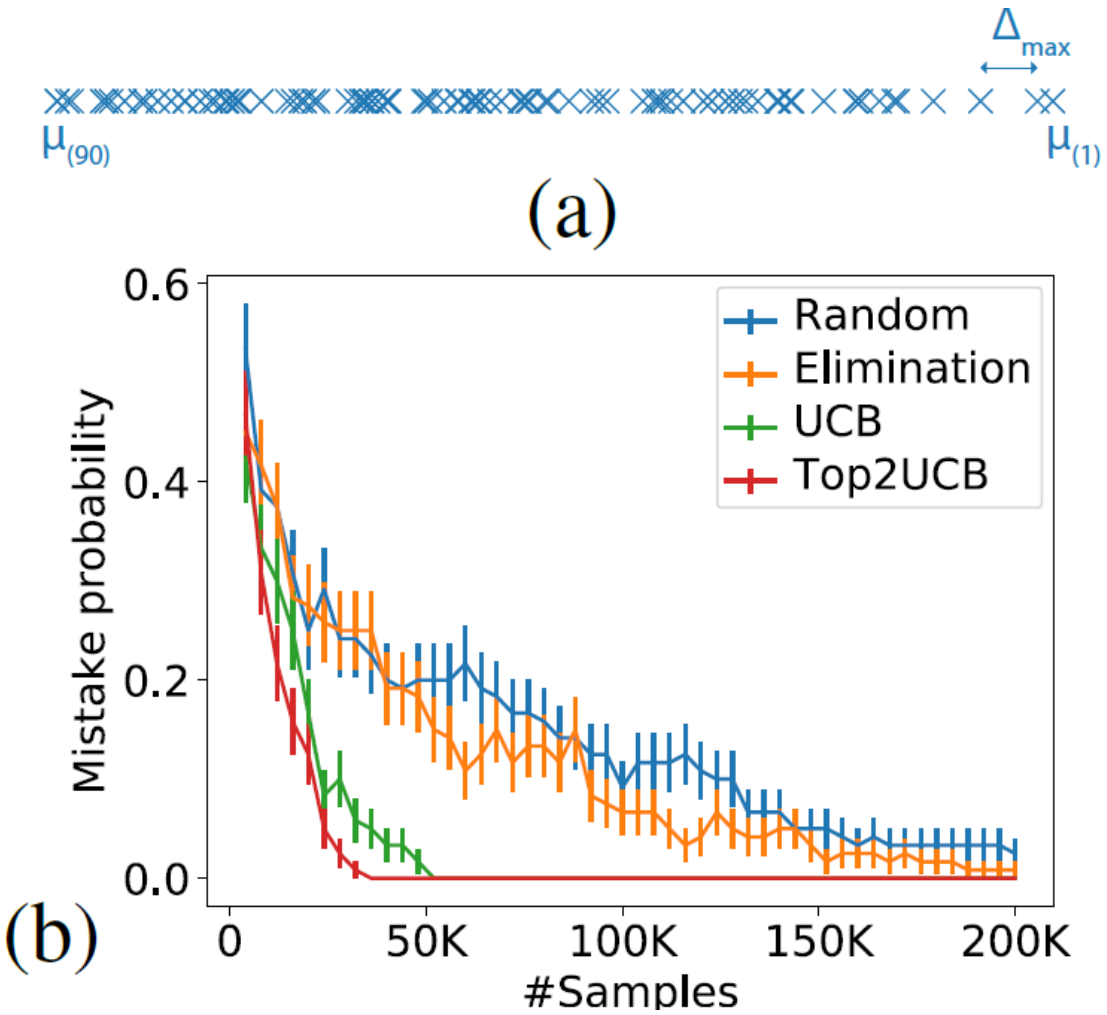
Sample Complexity Bound

- With probability $1-\delta$, *MaxGap Elimination*, *MaxGap UCB* and *MaxGap Top2UCB* algorithms cluster the arms according to the maximum gap.
- With probability $1-\delta$, the sample complexity of *MaxGap Elimination* and *MaxGap UCB* is bounded by,

$$O\left(\sum_{a\in[K]/\{(m),(m+1)\}} \frac{\log\left(K/\gamma_a\right)}{\gamma_a^2}\right). \tag{3}$$

- γ_a acts as *hardness parameter* for a sub-optimal arm a; It depends on the gaps and the difference between gaps and the optimal/largest gap.
- Intuitively, a sub-optimal arm will no longer be sampled if its $U\Delta$ falls below $L\Delta_{max}$.

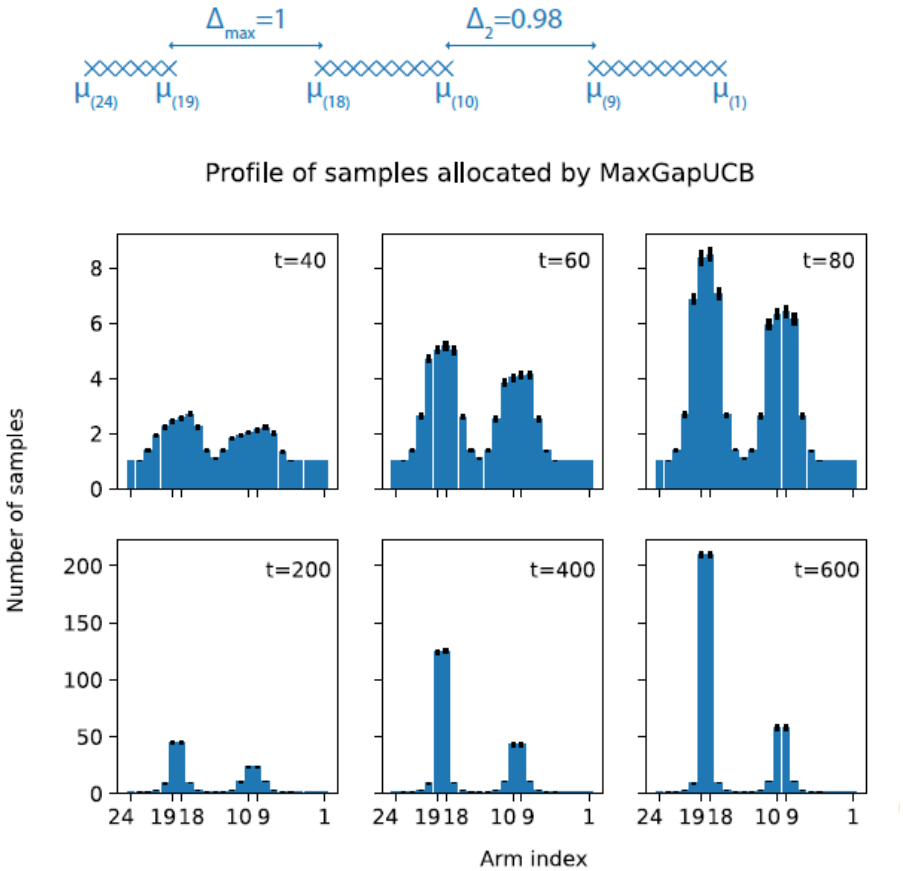
Results



Streetview Dataset: K=90,$\mathcal{N}((.), 0.05)$,$\Delta_{max} = 0.029$

[1] Katariya, Tripathy, and Nowak, *MaxGap Bandit: Adaptive Algorithms for Approximate Ranking*

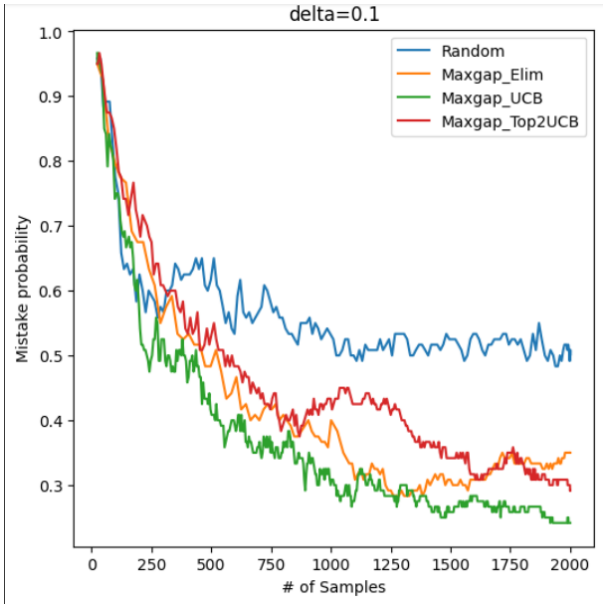
Results

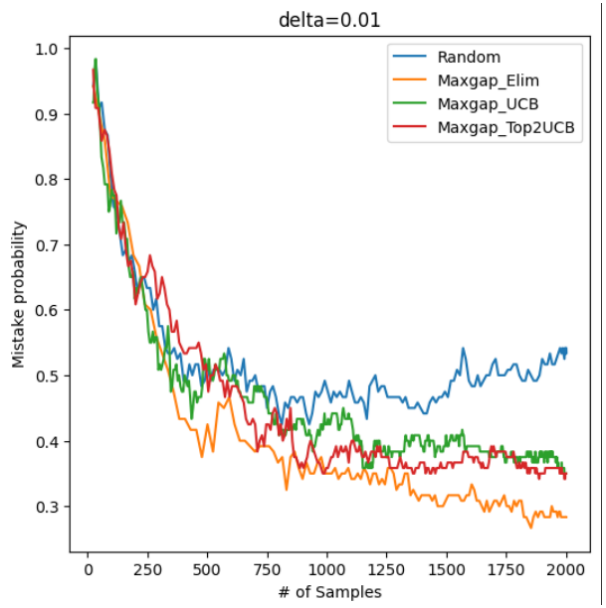


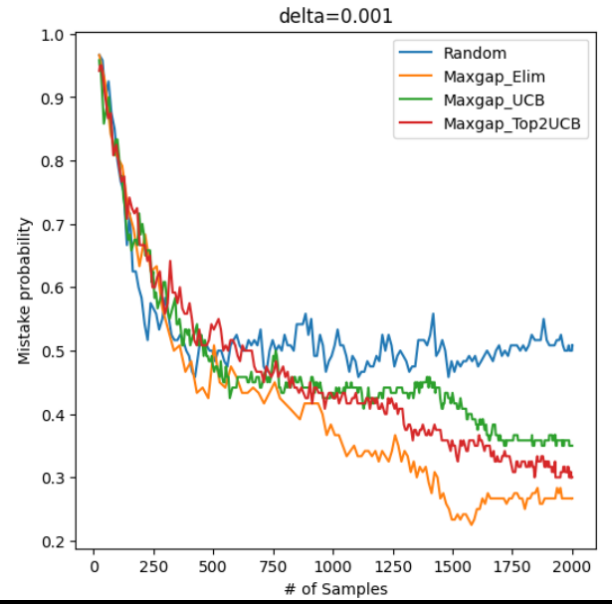
Performance of MaxGapUCB on a Simulated Dataset: K=24,$\mathcal{N}((.),1)$,$\Delta_{max} = 1$

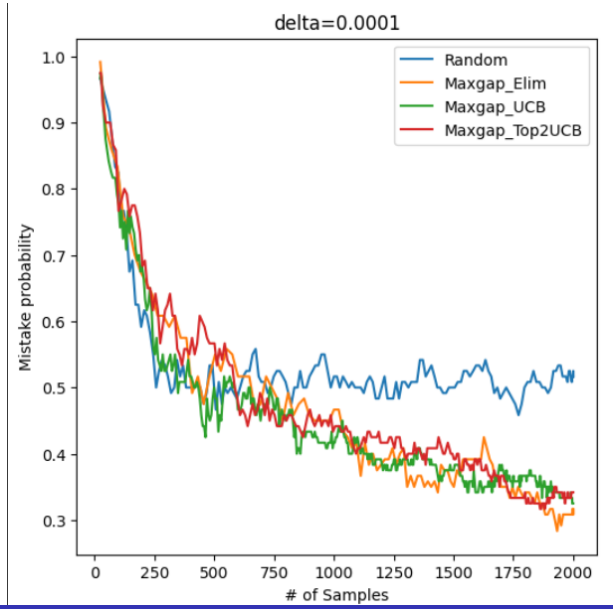
[1] Katariya, Tripathy, and Nowak, *MaxGap Bandit: Adaptive Algorithms for Approximate Ranking*

Simulations









Applications

- Maximum-gap identification problem can be used as a basic primitive for clustering and approximate ranking.

Applications

- Maximum-gap identification problem can be used as a basic primitive for clustering and approximate ranking.
- MaxGap bandits can be employed to preferentially recommend the item that has the highest expected utility or relevance to the user, considering the differences in expected rewards between different items.

Applications

- Maximum-gap identification problem can be used as a basic primitive for clustering and approximate ranking.
- MaxGap bandits can be employed to preferentially recommend the item that has the highest expected utility or relevance to the user, considering the differences in expected rewards between different items.
- MaxGap bandits can be used to identify the assets with the maximum expected return relative to the others, assisting in portfolio optimization.

References


Jamieson, Kevin et al. *lil' UCB : An Optimal Exploration Algorithm for Multi-Armed Bandits*. 2013. arXiv: 1312.7308 [stat.ML].


Katariya, Sumeet, Ardhendu Tripathy, and Robert Nowak. *MaxGap Bandit: Adaptive Algorithms for Approximate Ranking*. 2019. arXiv: 1906.00547 [stat.ML].

Introduction

- Multinomial logit bandit is a sequential subset selection problem which arises in many applications.
- In each round, the player selects a K-cardinality subset from N candidate items, and receives a reward which is governed by a multinomial logit (MNL) choice model considering both item utility and substitution property among items.
- Here, we try to study algorithms that tackle the dynamic assortment problem with linmited model adaptivity and obtain optimal minimax regret.

Need for Multinomial Logit Bandits

- The above mentioned subset selection problem can be modelled as an ordinary bandit problem with $\binom{n}{k}$ arms such that at time-step t, we choose any on of the arms from the combination based on our policy.
- The reward function is given by $R(S_t)$ where S_t represents the assortment set at time t.
- This formulation gives regret exponential in k and n and number of assortment switches exponential in k. Hence, to counter this problem we give an underlying model to the assortment selection based on parameters called the preference parameters.
- The goal of the problem is to learn an online policy to learn the parameters effectively based on an effective criteria(Risk-aware/ Minimum assortment switches).

Motivation

- All commercial trading(E-Commerce) have the necessity to solve the dynamic assortment problem to maximise their revenue and increase customer satisfaction. Thus, being a very common and intuitive problem motivated us to explore this topic

Application

- Online Advertising: Multinomial logit bandits can be used in online advertising to optimize the allocation of ads to different users. Advertisers can use bandit algorithms to learn about user preferences and adjust their ad allocation strategy accordingly.
- Food Delivery: Multinomial logit bandits can be used by food delivery services to optimize their recommendations to customers based on their ordering history and preferences. For example, a bandit algorithm might learn which types of cuisine a customer is most likely to order and adjust their restaurant recommendations accordingly.

Setting

- In this problem we have N distinct items, each of which is associated with a known reward r_i and an unknown preference parameter v_i
- In the MNL choice model, given a subset $S \subseteq [N] \stackrel{def}{=} \{1, 2, 3, ..., N\}$, the probability that a user chooses $i \in S$ is given by
-

$$p_i(S) = \begin{cases} \frac{\nu_i}{\nu_0 + \sum_{j \in S} \nu_j} & \text{if } i \in S \cup 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Setting- Continued

- We can model the reward r_i as a Random Variable $X(S_t, \nu)$ where $R(S_t, \nu)$ gives the expected value of X.
- The expected reward of the set S under the preference vector v = $\{v_0, v_1, ..., v_N\}$ can be represented mathematically as follows:
- $R(S_t, \nu) = \sum_{i \in S} r_i p_i(S) = \sum_{i \in S} \frac{r_i \nu_i}{1 + \sum_{j \in S} \nu_j}$
- $Reg_T \stackrel{def}{=} \sum_{t=1}^{T} (R(S^*, \nu) - R(S_t, \nu))$

Switching Costs

- Assortment Switching Cost: $\psi_T^{(asst)} \stackrel{def}{=} \sum_{t=1}^{T} \mathrm{II}[S_t \neq S_{t+1}]$
- Item Switching Cost : $\psi_T^{(item)} \stackrel{def}{=} \sum_{t=1}^{T} |S_t \oplus S_{t+1}|$
-

$$\psi_T^{(asst)} \leq \psi_T^{(item)} \leq min\{2K, N\}.\psi_T^{(asst)} \tag{5}$$

Exploration

Algorithm 1: EXPLORATION(S)

Initialize: $\Delta_i \leftarrow 0$ for all $i \in [N]$;
while TRUE **do**
  $t \leftarrow t + 1$;
  Offer assortment S, and observe purchase decision a_t;
  If $a_t = 0$ **then return** $\{\Delta_i\}$;
  $\Delta_{a_t} \leftarrow \Delta_{a_t} + 1$;

- In this part we divide the time steps into epochs, where in one epoch offer a particular assortment. The epoch ends when none of the elements are selected in the assortment.
- Let i be returned by EXPLORATION(S). For each $i \in S$, i is an independent geometric random variable with mean v_i.

Anytime Algorithm with Assortment Switches

Algorithm 2: Anytime Deferred Update UCB (AT-DUCB)

Initialize: $\hat{v}_i \leftarrow 1, T_i \leftarrow 0$ for all $i \in [N], t \leftarrow 0$;
for $\ell \leftarrow 1, 2, 3, \ldots,$ **do**
  Compute $S_\ell = \arg\max_{S \subseteq [N]:|S| \leq K} R(S, \hat{v})$;
  $\{\Delta_i\} \leftarrow \text{EXPLORATION}(S)$;
  for $i \in S$ **do**
    $n_i \leftarrow n_i + \Delta_i$ and $T_i \leftarrow T_i + 1$;
    if $T_i = 2^k$ *for some* $k \in \mathbb{Z}$ **then**
      $\bar{v}_i \leftarrow n_i / T_i$; $\hat{v}_i \leftarrow \min\left\{\hat{v}_i, \bar{v}_i + \sqrt{\frac{48\bar{v}_i \ln(\sqrt{N}\ell+1)}{T_i}} + \frac{48\ln(\sqrt{N}\ell+1)}{T_i}\right\}$;

- The regret of AT-DUCB algorithm is upper bounded as follows: $\mathbb{E}[Reg_T] \lesssim \sqrt{NTlogT}$
- $\mathbb{E}[\psi_T^{(asst)}] = \mathbb{E}\sum_{t=1}^{T-1} \mathrm{II}[S_t \neq S_{t+1}] \leq \sum_{l=1}^{L}\sum_{i=1}^{N} \mathrm{II}[\mathcal{D}_i^{(l)}]$

$= \sum_{i=1}^{N}\sum_{l=1}^{L} \mathrm{II}[\mathcal{D}_i^{(l)}] \lesssim NlogT$

Achieving assortment with known horizons(FH-DUCB)

Algorithm 3: UPDATE(i)

$\tau_i \leftarrow \tau_i + 1; T_i^{(\tau_i)} \leftarrow T_i^{(\tau_i - 1)} + |\mathcal{T}(i, \tau_i - 1)|;$

$n_i^{(\tau_i)} \leftarrow n_i^{(\tau_i - 1)} + n_{i,\tau_i - 1}; \bar{v}_{i,\tau_i} \leftarrow n_i^{(\tau_i)} / T_i^{(\tau_i)};$

$\hat{v}_{i,\tau_i} \leftarrow \min\left\{\hat{v}_{i,\tau_i - 1}, \bar{v}_{i,\tau_i} + \sqrt{\frac{48\bar{v}_{i,\tau_i}\ln(\sqrt{N}T^2+1)}{T_i^{(\tau_i)}}} + \frac{48\ln(\sqrt{N}T^2+1)}{T_i^{(\tau_i)}}\right\};$

Algorithm 4: Deferred Update UCB for Fixed Time Horizon (FH-DUCB)

Input : The time horizon T.

Initialize: $\tau_i \leftarrow 1, \hat{v}_{i,\tau_i} \leftarrow 1, n_{i,\tau_i} \leftarrow 0, \mathcal{T}(i,\tau_i) \leftarrow \emptyset, T_i^{(1)} \leftarrow 0, n_i^{(1)} \leftarrow 0$ for all $i \in [N]$;

$t \leftarrow 0, S_0 \leftarrow [N]$;

for $\ell \leftarrow 1, 2, 3, \ldots,$ **do**

  $S_\ell \leftarrow S_{\ell-1}$;

  if $\exists i : \mathcal{P}(i, \tau_i)$ *holds* **then**

    UPDATE(i) for all i such that $\mathcal{P}(i, \tau_i)$ holds;

    Compute $S_\ell \leftarrow \arg\max_{S \subseteq [N] : |S| \le K} R(S, \hat{\boldsymbol{v}}_\ell)$ where $\hat{\boldsymbol{v}}_\ell = (\hat{v}_{i,\tau_i(\ell)})_{i \in [N]}$;

  $\{\Delta_i\} \leftarrow$ EXPLORATION(S_ℓ);

  for $i \in S$ **do**

    $n_{i,\tau_i} \leftarrow n_{i,\tau_i} + \Delta_i$; Add ℓ to $\mathcal{T}(i, \tau_i)$;

$$\mathcal{P}(i,\tau_i) \stackrel{del}{=} \begin{cases} |\tau(i,\tau_i| \geq 1 + \sqrt{\dfrac{T.T_i^{(\tau_i)}}{N}} & if\ \tau_i < \tau_0 \\ |\tau(i,\tau_i| \geq 1 + \sqrt{\dfrac{T.T_i^{(\tau_i)}}{N.\hat{\nu_{i,\tau_i}}}}\ and\ \hat{\nu}_{i,\tau_0} > 1/\sqrt{NT} & if\ \tau_i \geq \tau_0 \end{cases} \tag{6}$$

- $\mathbb{E}[Reg_T] \lesssim \sqrt{NTln\sqrt{N}T^2 + 1}.loglog(T)$
- $\mathbb{E}[\psi_T^{(item)}] \lesssim NloglogT$

Risk Aware Case

Risk Criterion

- The Risk Criterion Defined by U is a function from D $\Rightarrow$ R, where D is the convex set of cumulative distribution functions F(S_t, ν).
- The goal is to find a policy that maximizes $\mathbb{E}\sum_{t=1}^{T} U(F(S_t, \nu))$, i.e to find the optimal assortment.

Examples of Risk-Criteria

- Given $\alpha \in (0, 1]$, the conditional value-at-risk at α percentile for F $\in$ D[0, 1] is defined as $CVaR_\alpha(F) = \frac{1}{\alpha}(\alpha - \int_0^1 (F(x)\hat{}\alpha)dx$
- Given the risk aversion parameter $\theta \geq 0$, the entropy risk measure for F $\in$ D[0, 1] is defined as $U^{ent}(F) = \frac{-1}{\theta} \ln(\int_0^1 e^{-\theta x} dF(x)$

Criterias to be satisfied by the risk criterion

Assumption1

(Quasiconvexity). U is quasiconvex on $D[0,1]$, i.e., for any $\lambda \in [0,1]$ and $F_1, F_2 \in D[0,1]$, it satisfies

$$U(\lambda F_1 + (1-\lambda)F_2) \leq \max\{U(F_1), U(F_2)\}$$

Assumption 2 (Boundedness). For any $F \in D[0,1]$, it holds that $|U(F)| \leq \gamma_1$.

Assumption3

(One-sided Lipschitz Condition). For any $v_0 > v$, i.e., $v_{0,i} > v_i$ for all $i \in [N]$, and $S \subseteq [N]$, it holds that

$$U(F(S, v_0)) - U(F(S, v)) \leq \frac{\gamma_2}{1 + \sum_{i \in S} v_i} \sum_{i \in S} (v_{0,i} - v_i)^+,$$

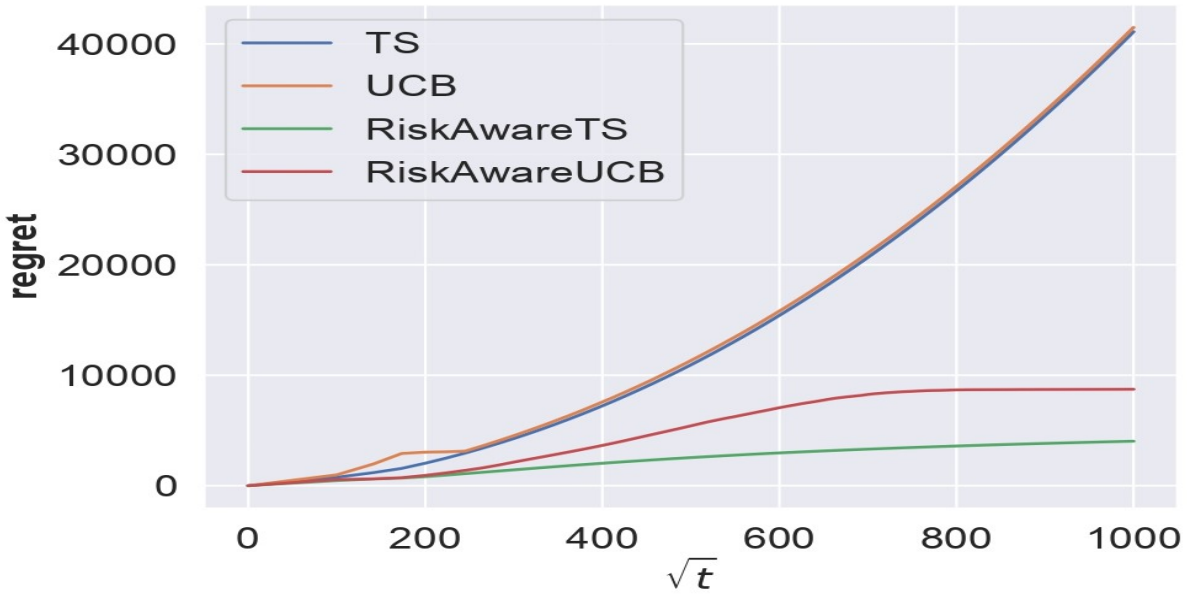
Risk Aware UCB

Algorithm 1: `RiskAwareUCB`$(N, K, \boldsymbol{r}, U)$

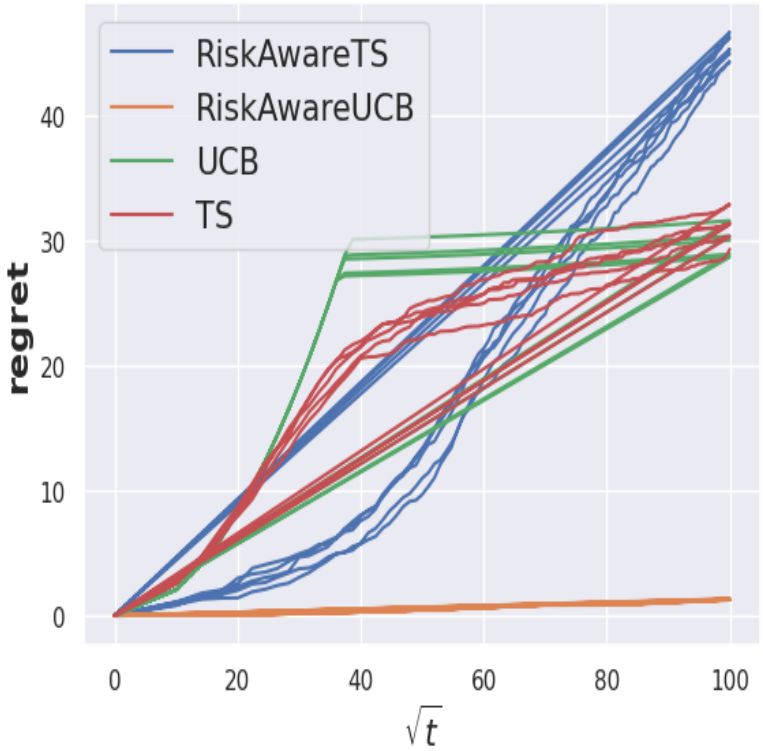
Initialize $t \leftarrow 1$, $\ell \leftarrow 1$, $\widetilde{v}_i^{\ell} \leftarrow 1$ for $i \in [N]$ and $\mathcal{T}_i(\ell) \leftarrow \emptyset$ for $i \in [N]$
while $t \leqslant T$ **do**
  $S_\ell \leftarrow \operatorname{argmax}_{S \subset [N], |S| \leqslant K} U(F(S, \widetilde{\boldsymbol{v}}^{\ell}))$
  Initialize $t_{i,\ell} \leftarrow 0$ for $i \in [N]$
  repeat
    Serve S_ℓ and observe customer choice c_t
    if $c_t \neq 0$ **then** $t_{c_t,\ell} \leftarrow t_{c_t,\ell} + 1$
    $t \leftarrow t + 1$
  until $t > T$ *or* $c_{t-1} = 0$
  for $i \in [N]$ **do**
    if $i \in S_\ell$ **then** $\mathcal{T}_i(\ell+1) \leftarrow \mathcal{T}_i(\ell) \cup \{\ell\}$
    else $\mathcal{T}_i(\ell+1) \leftarrow \mathcal{T}_i(\ell)$
  $\ell \leftarrow \ell + 1$, $T_i(\ell) \leftarrow |\mathcal{T}_i(\ell)|$ for $i \in [N]$
  $\bar{v}_i^{\ell} \leftarrow \frac{\sum_{\ell' \in \mathcal{T}_i(\ell)} t_{i,\ell'}}{T_i(\ell)}$ for $i \in [N]$
  $\widetilde{v}_i^{\ell} \leftarrow \min\left\{ \bar{v}_i^{\ell} + \sqrt{\bar{v}_i^{\ell} \cdot \frac{48 \ln(\sqrt{N}\ell+1)}{T_i(\ell)}} + \frac{48 \ln(\sqrt{N}\ell+1)}{T_i(\ell)}, 1 \right\}$ for $i \in [N]$

$$
\begin{aligned}
\mathfrak{R}_T &\leqslant 2\gamma_1(N+1)(\ln T + \gamma) \\
&\quad + \gamma_2 C_1 \sqrt{\ln(\sqrt{N}T+1)}(2\sqrt{NT}+N) \\
&\quad + \gamma_2 C_2 \ln(\sqrt{N}T+1)(N \ln T + N(1+\gamma)) \\
&= \widetilde{\mathcal{O}}(\sqrt{NT})
\end{aligned}
$$

Experiment Evaluation



In this experiment, we fix the number of products $N = 10$, cardinality limit $K = 4$, horizon $T = 10^6$, and set the goal to be $U = CVaR_{0.5}$. We generate 10 uniformly distributed random input instances where $v_i \in [0, 1]$ and $r_i \in [0.1, 1]$. For each input instance, we run 20 repetitions and compute their average as the regret. The above figure shows how the worst regret among all input instances changes with square root of time.


RiskAwareTS
RiskAwareUCB
UCB
TS
regret
0
10
20
30
40
0
20
40
60
80
100
$\sqrt{t}$