IE 613: Survey of Dueling Bandits

Anshul Nasery (170070015)
Rishabh Dahale (17D070008)
Mithilesh Vaidya (17D070011)

May 2019

1 Introduction

Conventional online learning settings require absolute feedback for each action (in the form of a reward or loss). However, in some cases, such feedback is not possible. E.g. search engine suggestions and their click-through-rate only tell us which action was preferred over the other. There is no absolute measure (such as how strongly was the clicked suggestion preferred over the ignored one). The Dueling Bandits framework solves this problem by assuming only the presence of feedback about the relative quality of each pair of actions.
The aim of this project is a survey of various algorithms used in the Dueling Bandit setting. This report is organised in the following manner:
We first introduce Dueling Bandits in Section 2 and define various terms used in describing the environment and the algorithm's performance. Section 3 contains the list of algorithms along with a brief description of each algorithm. They are compared with each other in the same environment and the results are mentioned in Section 4 and Section 5.

2 Definition

In several real world applications, decisions need to be taken based on feedback from comparison between pairs of actions. Dueling bandits try to model this problem.

In the Dueling Bandits problem, the following happens for each time step $t = 1, ..., T$:

- The algorithm chooses a pair of actions a_i and a_j from K available actions
- The world provides (independent stochastic) preference feedback of which action is more preferred. The first action is preferred with probability $P(a_i \succ a_j)$, and the second with $1 - P(a_i \succ a_j)$

In this setting, assuming that the first arm a_1 is the best arm, regret is defined as

$$R(T) = \sum_{t=1}^{T} \Delta_{1i} + \Delta_{1j}$$

where Δ_{1i} represents $P(a_1 \succ a_i)$

Another definition associated with the problem is that of a winner arm. An arm a_i is said to be a Condorcet winner if for all other arms a_j,

$$[P(ai \succ a_j) \geq P(a_j \succ a_i)]$$

3 Algorithms

Several algorithms have been proposed for the Dueling Bandit setting.
We studied and simulated the following:

3.1 Interleaved Filter [1]

This is an explore-then-exploit type of an algorithm. In the explore phase, all the arms are compared with a candidate arm in a round robin fashion, and they are discarded from the set if the candidate arm is better than them with a certain level of confidence. If some other arm is better than the candidate arm within a confidence bound, it becomes the new candidate arm. Once only one arm remains in the set to be considered, the algorithm enters the exploit phase, repeatedly pulling this arm. The algorithm has a regret bound of $\mathcal{O}(\frac{K \log K}{\epsilon_{12}} \log T)$. However, the variance of the regret is high due to the fact that arms are rapidly eliminated w.r.t the candidate arm.

Algorithm 2 Interleaved Filter 1 (IF1)

Input: $T, \mathcal{B} = \{b_1, \ldots, b_K\}$
$\delta \leftarrow 1/(TK^2)$
Choose $\hat{b} \in \mathcal{B}$ randomly
$W \leftarrow \{b_1, \ldots, b_K\} \setminus \{\hat{b}\}$
$\forall b \in W$, maintain estimate $\hat{P}_{\hat{b},b}$ of $P(\hat{b} > b)$
$\forall b \in W$, maintain $1-\delta$ confidence interval $\hat{C}_{\hat{b},b}$ of $\hat{P}_{\hat{b},b}$
while $W \neq \emptyset$ **do**
  for $b \in W$ **do**
    compare $\hat{b}$ and b
    update $\hat{P}_{\hat{b},b}, \hat{C}_{\hat{b},b}$
  end for
  while $\exists b \in W$ s.t. $\left(\hat{P}_{\hat{b},b} > 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b}\right)$ **do**
    $W \leftarrow W \setminus \{b\}$
  end while
  if $\exists b' \in W$ s.t. $\left(\hat{P}_{\hat{b},b'} < 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b'}\right)$ **then**
    $\hat{b} \leftarrow b'$, $W \leftarrow W \setminus \{b'\}$ *//new round*
    $\forall b \in W$, reset $\hat{P}_{\hat{b},b}$ and $\hat{C}_{\hat{b},b}$
  end if
end while
$\hat{T} \leftarrow$ Total Comparisons Made
return $(\hat{b}, \hat{T})$

3.2 Beat The Mean [2]

The algorithm maintains a working W for wach round, and looks at the probability of each bandit belonging to W to beat the mean bandit. It chooses the bandit with the least pulls and compares it with the mean bandit. The mean bandit is simulated by selecting one of the bandits uniformly. Once the empirical probabilities of the best and worst bandit are separated by a confidence level, the worst bandit is removed from W. The algorithm is often found to be slow to converge, especially for a larger number of arms, since the exploration strategy is akin to uniform exploration, and the confidence bounds often mean

that W remains unchanged for a long time. The theoretical bound on regret is proved to be $\mathcal{O}\left(\Sigma_{k=2}^{K}\frac{\gamma^8}{\eta_{1,k}}logT\right)$

Algorithm 1 BEAT-THE-MEAN

Input: $\mathcal{B}=\{b_1,\ldots,b_K\}$, N, T, $c_{\delta,\gamma}(\cdot)$
$W_1 \leftarrow \{b_1,\ldots,b_K\}$ *//working set of active bandits*
$\ell \leftarrow 1$ *//num rounds*
$\forall b \in W_\ell$, $n_b \leftarrow 0$ *//num comparisons*
$\forall b \in W_\ell$, $w_b \leftarrow 0$ *//num wins*
$\forall b \in W_\ell$, $\hat{P}_b \equiv w_b/n_b$, or 1/2 if $n_b = 0$
$n^* \equiv \min_{b\in W_\ell} n_b$
$c^* \equiv c_{\delta,\gamma}(n^*)$, or 1 if $n^* = 0$ *//confidence radius*
$t \leftarrow 0$ *//total number of iterations*
while $|W_\ell| > 1$ **and** $t < T$ **and** $n^* < N$ **do**
  $b \leftarrow \text{argmin}_{b\in W_\ell} n_b$ *//break ties randomly*
  select $b' \in W_\ell$ at random, compare b vs b'
  if b wins, $w_b \leftarrow w_b + 1$
  $n_b \leftarrow n_b + 1$
  $t \leftarrow t + 1$
  if $\min_{b'\in W_\ell} \hat{P}_{b'} + c^* \leq \max_{b\in W_\ell} \hat{P}_b - c^*$ **then**
    $b' \leftarrow \text{argmin}_{b\in W_\ell} \hat{P}_b$
    $\forall b \in W_\ell$, delete comparisons with b' from w_b, n_b
    $W_{\ell+1} \leftarrow W_\ell \setminus \{b'\}$ *//update working set*
    $\ell \leftarrow \ell + 1$ *//new round*
  end if
end while
return $\text{argmax}_{b\in W_\ell} \hat{P}_b$

Algorithm 2 BEAT-THE-MEAN (Online)

Input $\mathcal{B}=\{b_1,\ldots,b_K\}$, γ, T
$\delta \leftarrow 1/(2TK)$
Define $c_{\delta,\gamma}(\cdot)$ using (4)
$\hat{b} \leftarrow$ BEAT-THE-MEAN($\mathcal{B}$, ∞, T, $c_{\delta,\gamma}$)

3.3 Relative Minimum Empirical Divergence [3]

This is a state of the art algorithm which achieves the optimal asymptotic regret bound. The algorithm computes the empirical KL divergence $I_i(t)$ as

$$I_i(t) = \sum_{j\in O_i(t)} N_{i,j}(t) d(\hat{\mu_{i,j}}(t), \frac{1}{2})$$

, where $O_i(t)$ is the set of arms whose probability of beating a_i is less than 0.5. It considers $\exp(-I_i(t))$ to be the likelihood of an arm being the optimal arm, uses this to decide which arm to compare next.

Algorithm 1 Relative Minimum Empirical Divergence (RMED) Algorithm

Input: K arms, $f(K) \geq 0$. $\alpha > 0$ (RMED2FH, RMED2). T (RMED2FH).

$L \leftarrow \begin{cases} 1 & \text{(RMED1, RMED2)} \\ \lceil \alpha \log \log T \rceil & \text{(RMED2FH)} \end{cases}$.

Initial phase: draw each pair of arms L times. At the end of this phase, $t = L(K-1)K/2$.

if RMED2FH **then**

  For each arm $i \in [K]$, fix $\hat{b}^*(i)$ by (6).

end if

$L_C, L_R \leftarrow [K], L_N \leftarrow \emptyset$.

while $t \leq T$ **do**

  if RMED2 **then**

    Draw all pairs (i, j) until it reaches $N_{i,j}(t) \geq \alpha \log \log t$. $t \leftarrow t + 1$ for each draw.

  end if

  for $l(t) \in L_C$ in an arbitrarily fixed order **do**

    Select $m(t)$ by using $\begin{cases} \text{Algorithm 2} & \text{(RMED1)} \\ \text{Algorithm 3} & \text{(RMED2, RMED2FH)} \end{cases}$.

    Draw arm pair $(l(t), m(t))$.

    $L_R \leftarrow L_R \setminus \{l(t)\}$.

    $L_N \leftarrow L_N \cup \{j\}$ (without a duplicate) for any $j \notin L_R$ such that $\mathcal{J}_j(t)$ holds.

    $t \leftarrow t + 1$.

  end for

  $L_C, L_R \leftarrow L_N, L_N \leftarrow \emptyset$.

end while

Algorithm 2 RMED1 subroutine for selecting $m(t)$

$\hat{\mathcal{O}}_{l(t)}(t) \leftarrow \{j \in [K] \setminus \{l(t)\} | \hat{\mu}_{l(t),j}(t) \leq 1/2\}$

if $i^*(t) \in \hat{\mathcal{O}}_{l(t)}(t)$ or $\hat{\mathcal{O}}_{l(t)}(t) = \emptyset$ **then**

  $m(t) \leftarrow i^*(t)$.

else

  $m(t) \leftarrow \arg\min_{j \neq l(t)} \hat{\mu}_{l(t),j}(t)$.

end if

3.4 Doubler [4]

The Stochastic Multi-Armed Bandit(SMAB) has been widely studied over the last few decades. We can benefit from this research if we can convert the Dueling Bandit problem into a MAB setting. The Doubler algorithm works by treating the MAB algorithm as a 'blackbox' and interacts with it using three functions:

- Reset: Clears all the weights
- Advance: Play an arm
- Feedback: Supply reward to the black box for updating it's weights

Algorithm 2 (**Doubler**): Reduction for finite and infinite X with internal structure.

$S \leftarrow$ new SBM over X
$\mathcal{L} \leftarrow$ an arbitrary singleton in X
$i \leftarrow 1, t \leftarrow 1$
while true **do**
  reset(S)
  for $j = 1...2^i$ **do**
    choose x_t uniformly from $\mathcal{L}$
    $y_t \leftarrow$ advance(S)
    play (x_t, y_t), observe choice b_t
    feedback(S, b_t)
    $t \leftarrow t + 1$
  end for
  $\mathcal{L} \leftarrow$ the multi-set of arms played as y_t in the last for-loop
  $i \leftarrow i + 1$
end while

Algorithm 1 UCB algorithm for MAB with $|X| = K$ arms. Parameter α affects tail of regret per action in X.

$\forall x \in X$, set $\hat{\mu}_x = \infty$
$\forall x \in X$, set $t_x = 0$
set $t = 1$
while true **do**
  let x be the index maximizing $\hat{\mu}_x + \sqrt{\frac{(\alpha+2)\ln(t)}{2t_x}}$
  play x and update $\hat{\mu}_x$ as the average of rewards so far on action x; increment t_x by 1.
  $t \leftarrow t + 1$
end while

Any standard MAB algorithm can be used. We used UCB since it gives good performance without large performance overheads. Doubler proceeds in epochs of exponentially increasing size (hence "doubler"). In each epoch, the left arm is sampled from a fixed distribution, and the right arm is chosen using the MAB black-box to minimise regret against the left arm. The feedback sent to the blackbox is the wins and losses the right arm encounters when compared against the left arm. In other words, the goal of the right arm is to beat the fixed distribution from which the left arm is sampled. The distribution the left arm plays is the empirical distribution (histogram) of arms that were chosen for the right arm in the previous epoch. The regret bound for doubler is proved to be $\mathcal{O}\left(c\frac{\alpha}{\alpha+1}log^{\alpha+1}T\right)$

3.5 SAVAGE [5]

Sensitivity Analysis of Variables for Generic Exploration (SAVAGE) is a popular algorithm which outperforms most when the number of arms is moderate. It can be briefly explained in the following manner:
It compares pairs of arms in a round-robin fashion and drops arms from consideration when they meet a particular criterion. Since the Condorcet assumption is valid, an arm which loses to others with high probability can be safely dropped since it cannot be the winner.

Algorithm 1 SAVAGE algorithm

Input: $\mathbf{X} = (X_1, \ldots, X_N)$, f, $\mathcal{F}$, T, δ
Initialization:
$\mathcal{W} := \{1, \ldots, N\}$, $\mathcal{H} := \mathcal{F}$, $s := 1$
$\forall i \in \mathcal{W}$: $\hat{\mu}_i := 1/2$, and $t_i := 0$
while $\neg$**Accept**$(f, \mathcal{H}, \mathcal{W}) \wedge s \leqslant T$ **do**
  Pick a variable index $i \in \arg\min_{\mathcal{W}} \{t_1, \ldots, t_N\}$
  $t_i := t_i + 1$
  Sample the i^{th} distribution $x_i \leftarrow X_i$
  $\hat{\mu}_i := (1 - \frac{1}{t_i})\hat{\mu}_i + \frac{1}{t_i} x_i$
  $\mathcal{H} := \mathcal{H} \cap \{\mathbf{x} \mid |x_i - \hat{\mu}_i| < c(t_i)\}$
  $\mathcal{W} := \mathcal{W} \backslash \{j \parallel$ **IndepTest**$(f, \mathcal{H}, j)\}$
  $s := s + 1$
end while
return $\hat{d} \in f(\mathcal{H})$

Algorithm 2 Parameters Elimination by Sampling

Input: f,$\mathcal{H}$,$\mathcal{W}$,m,M
Initialization:
$\mathcal{S} \leftarrow \varnothing$
for $l = 1, \ldots, m$ **do**
  Sample $\mathbf{x}$ uniformly from $\mathcal{H}$
  $\mathbf{x}' \leftarrow \mathbf{x}$
  for $s = 1, \ldots, M$ **do**
    Pick a random parameter $i \in \mathcal{W} \backslash \mathcal{S}$
    Re-sample x_i until $\mathbf{x} \in \mathcal{H}$
    if $f(\mathbf{x}) \neq f(\mathbf{x}')$ **then**
      $\mathcal{S} := \mathcal{S} \cup \{i\}$
    end if
    $x'_i \leftarrow x_i$
  end for
end for
$\mathcal{W} := \mathcal{W} \cap \mathcal{S}$

3.6 RUCB [6]

The main idea of RUCB is to maintain optimistic estimates of the probabilities of all possible pairwise outcomes, and

1. use these estimates to find an arm that has a chance of being the best arm (potential champion)

2. select an arm to compare to this potential champion by performing regular UCB

Algorithm 1 Relative Upper Confidence Bound

Input: $\alpha > \frac{1}{2}, T \in \{1, 2, \ldots\} \cup \{\infty\}$

$\mathbf{W} = [w_{ij}] \leftarrow \mathbf{0}_{K \times K}$ // 2D array of wins: w_{ij} is the number of times a_i beat a_j

$\mathcal{B} = \varnothing$

for $t = 1, \ldots, T$ **do**

  $\mathbf{U} := [u_{ij}] = \frac{\mathbf{W}}{\mathbf{W}+\mathbf{W}^T} + \sqrt{\frac{\alpha \ln t}{\mathbf{W}+\mathbf{W}^T}}$ // All operations are element-wise; $\frac{x}{0} := 1$ for any x.

  $u_{ii} \leftarrow \frac{1}{2}$ for each $i = 1, \ldots, K$.

  $\mathcal{C} \leftarrow \left\{a_c \,|\, \forall\, j : u_{cj} \geq \frac{1}{2}\right\}$.

  If $\mathcal{C} = \varnothing$, then pick c randomly from $\{1, \ldots, K\}$.

  $\mathcal{B} \leftarrow \mathcal{B} \cap \mathcal{C}$.

  If $|\mathcal{C}| = 1$, then $\mathcal{B} \leftarrow \mathcal{C}$ and a_c to be the unique element in $\mathcal{C}$.

  if $|\mathcal{C}| > 1$ **then**

    Sample a_c from $\mathcal{C}$ using the distribution:

$$p(a_c) = \begin{cases} 0.5 & \text{if } a_c \in \mathcal{B}, \\ \frac{1}{2^{|\mathcal{B}|}\,|\mathcal{C} \setminus \mathcal{B}|} & \text{otherwise.} \end{cases}$$

  end if

  $d \leftarrow \arg\max_j u_{jc}$, with ties broken randomly. Moreover, if there is a tie, d is not allowed to be equal to c.

  Compare arms a_c and a_d and increment w_{cd} or w_{dc} depending on which arm wins.

end for

Return: An arm a_c that beats the most arms, i.e., c with the largest count $\#\left\{j | \frac{w_{cj}}{w_{cj}+w_{jc}} > \frac{1}{2}\right\}$.

RUCB starts from putting all the arms in a pool of potential champions, and then starts comparing every arm against every other arm optimistically. If the arm i, the upper bound $u_{ij}(t) = \mu_{ij}(t) + c_{ij}(t)$ satisfies $u_{ij}(t) < 0.5$ then the arm i is removed from the potential champions. The term $\mu_{ij}(t)$ defines the frequentist bonus at time t. It can be also considered as the measure of exploitation performed by an arm. The term $c_{ij}(t)$ is called as optimism bonus that increases with time t and decreases with number of comparisons between arms i and j. This is performed until we get a single best arm, which by the assumption of Condorcet winner is always present.

In short RUCB can be summed in two steps. First is the optimistic comparison of an arm with other arms, making it easier for it to become a champion. And second is the pessimistic comparison of an arm against others, making it more difficult for an arm to be compared against itself. This is important because comparing an arm with itself yields no information. Thus RUCB tries to avoid auto-comparison until there is great certainty that the chosen arm is indeed the Condorcet winner. The regret for RUCB is proved to be $\mathcal{O}\left(KlogT\right)$

3.7 RCS [7]

RCS (Relative confidence sampling) proceeds in two phases to determine which ranker to interleave at each iteration. First, it use the results of the comparisons conducted so far to simulate a round-robin tournament among the rankers. Second, the champion of this tournament is compared against a challenger deemed to have the best chance of beating it. As more comparisons are conducted, the best ranker is increasingly likely to be selected as both champion and challenger, causing regret to fall steeply over time.

Algorithm 1 Relative Confidence Sampling (RCS)

Input: A set of rankers $\rho_1, \ldots, \rho_K$ and an oracle that can take a pair of rankers and return one as the winner (e.g., an interleaved comparison method)

Choose $\alpha > \frac{1}{2}$
$\mathbf{W} \leftarrow \mathbf{0}_{K \times K}$ // *2D array of wins:* $\mathbf{W}_{ij}$ *is the number of times* ρ_i *beat* ρ_j
for $t = 1, 2, \ldots$ **do**
  // **I***: Run a simulated "tournament":*
  $\boldsymbol{\Theta}(t) \leftarrow \frac{\mathbf{1}_{K \times K}}{2}$
  for $i, j = 1, \ldots, K$ with $i < j$ **do**
    $\boldsymbol{\Theta}_{ij}(t) \sim \text{Beta}(\mathbf{W}_{ij} + 1, \mathbf{W}_{ji} + 1)$
    $\boldsymbol{\Theta}_{ji}(t) = 1 - \boldsymbol{\Theta}_{ij}(t)$
  end for
  Pick c such that $\boldsymbol{\Theta}_{cj}(t) \geq \frac{1}{2}$ for all j. If no such ranker exists, pick the ranker that has been chosen champion least often.
  // **II***: Run UCB in relation to c:*
  $\mathbf{U}(t) = \frac{\mathbf{W}}{\mathbf{W}+\mathbf{W}^T} + \sqrt{\frac{\alpha \ln t}{\mathbf{W}+\mathbf{W}^T}}$ // *All operations are element-wise, with* $\frac{x}{0} := 1$ *for any* x.
  $\mathbf{U}_{ii}(t) \leftarrow \frac{1}{2}$ for each $i = 1, \ldots, K$.
  $d \leftarrow \arg\max_j \mathbf{U}_{jc}(t)$
  // **III***: Update* $\mathbf{W}$
  Compare rankers ρ_c and ρ_d and increment either $\mathbf{W}_{cd}$ if ρ_c beat ρ_d or $\mathbf{W}_{dc}$ otherwise.
end for

While RCS is related to RUCB, which is also designed for the ongoing regret minimisation setting, it differs in one crucial respect: the use of sampling when conducting a round-robin tournament to select a champion. The goal in doing so is to exploit one of the key lessons that has been learned in the study of regular K-armed bandits: that much better performance can be obtained by maintaining posterior distributions over the expected value of each ranker and sampling from those posteriors to determine which ranker to select. RCS has one parameter α which controls how exploratory the algorithm's behaviour is: higher value of α means the more the algorithm explores and slower it settles on a single ranker. The RCS maintains a scoresheet W, which records the comparision result and proceeds in two phases:

- A tournament is simulated based on current scoresheet, i.e. samples Θ_{ij} are collected for each pair of ranker (i, j) with $i > j$, from a posterior beta distribution maintained on p_{ij}. Also since $p_{ji} = 1 - p_{ij}$, RCS sets $\Theta_{ji} = 1 - \Theta_{ij}$. Also it sets $\Theta_{ii} = 0.5$ as $p_{ii} = 0.5$. From these sampled results, ranker i beats ranker j in the simulated

tournament if $\Theta_{ij} > \frac{1}{2}$ for $i \neq j$. At this stage there are two possibilities: First, there is a champion ranker c that beats all other rankers in this tournament i.e. $\Theta_{cj} \geq \frac{1}{2}$. Second, no ranker beats all other ranker, in which case RCS sets $c = argmin_i N_i$ where N_i is the number of times arm i has been previously chosen as champion. Eventually once the condorcet winner has been compared against the rest of the rankers often enough, it's superiority over the rest will cause their eliminating in this phase of the algorithm.

- The UCB algorithm is applied to the K-armed bandit problem with means $\{p_{1_c}, ..., p_{k_c}\}$. RCS picks the ranker d for which u_{dc} is higher than all other u_{jc}

Finally rankers c and d are compared against each other using a real interleaved comparison and W is updated accordingly.

4 Experimental Results

We experimented on 4 datasets - 2 synthetic and 2 real-world
Reference for both synthetic datasets is [10]

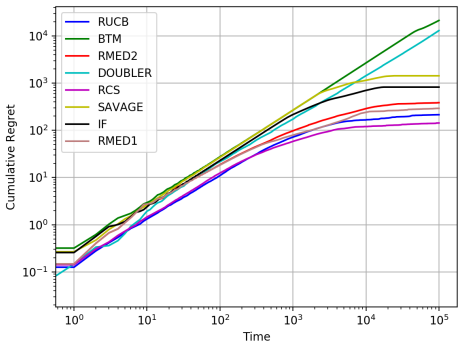
- Synthetic with K = 5 - Best arm is the first arm.
- Synthetic with K = 10 - It is generated by using the rule:
  $\mu_{i,j} = 0.5 + 0.05(j - i)$. Best arm in the case is the first arm.

The real-world datasets are the MSLR dataset [9] and car preference dataset [8]

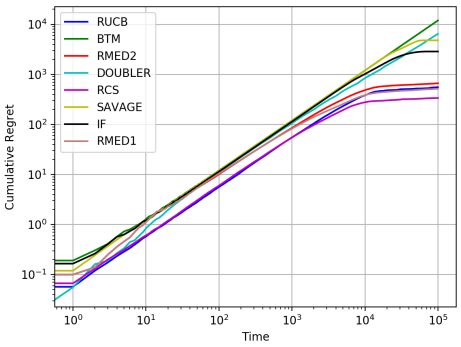
- MSLR - K = 5 - First arm is the best arm. It is generated by considering the preferences of users ranking features for URLs while using the search engine Bing! The raw dataset had relative rankings of each feature per user.
- Car preference - K = 10 - First arm is the best arm. It is generated by considering the user preferences for various models of cars. The raw dataset features comparisons between 2 cars for each user, so it could be directly formulated as a dueling bandit problem.

Let $\delta(env)$ denote difference between the highest mean and the next highest mean. From the datasets, we can conclude that:
$\delta(Artificial, K = 5) < \delta(Real, K = 10) < \delta(Artificial, K = 10) < \delta(Real, K = 5)$.
In the following work, we will refer to it as the 'sub-optimal gap'

The results are given below. 50 samples of each algorithm are taken to average out the noise. Note that both regret and horizon are plotted on a log scale.

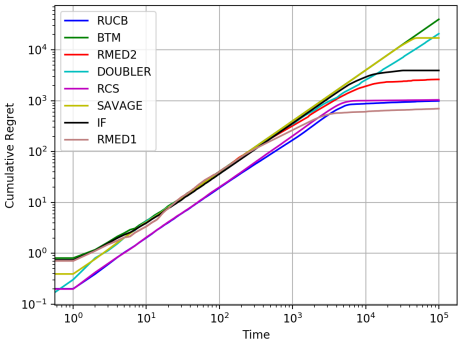


(a) MSLR dataset with K = 5

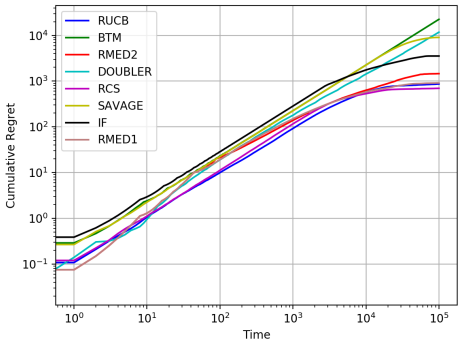


(b) Synthetic dataset with K = 5

Figure 1: For K = 5



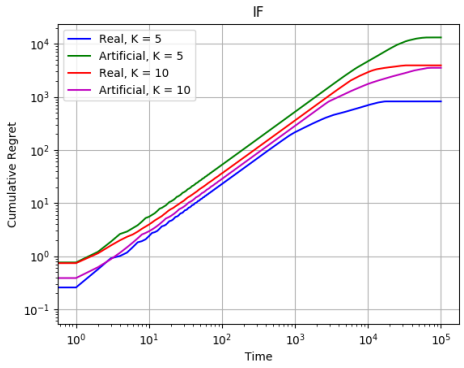
(a) Car preference dataset with K = 10



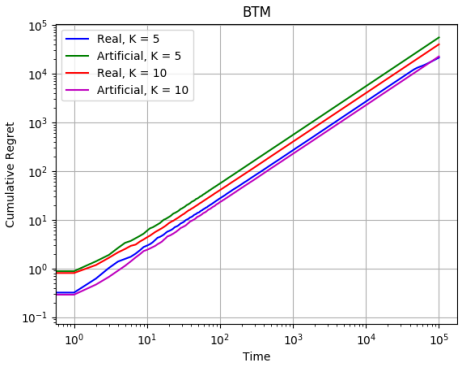
(b) Synthetic dataset with K = 10

Figure 2: For K = 10

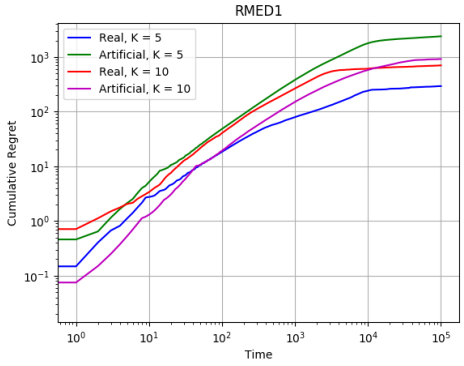
Given below are the plots for each algorithm:



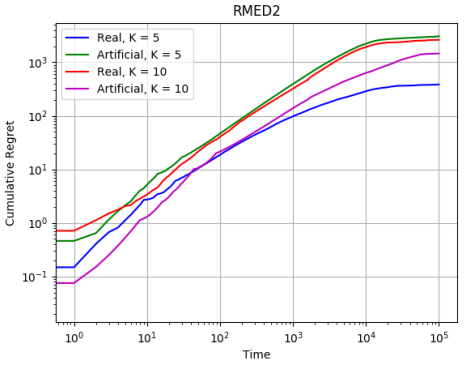
(a) Interleaved Filter



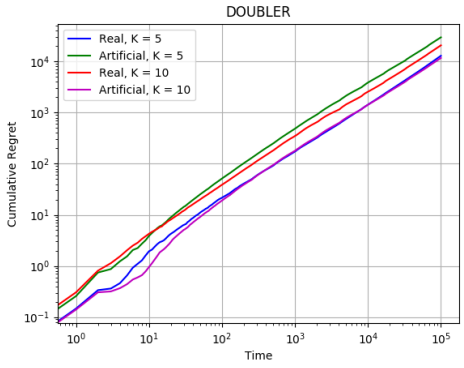
(b) Beat the Mean (BTM)



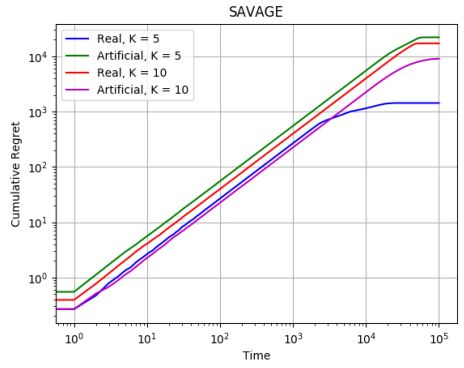
(c) RMED1



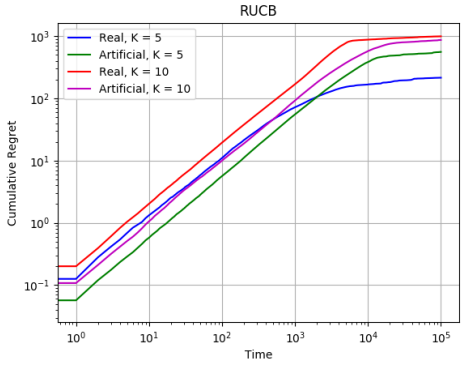
(d) RMED2



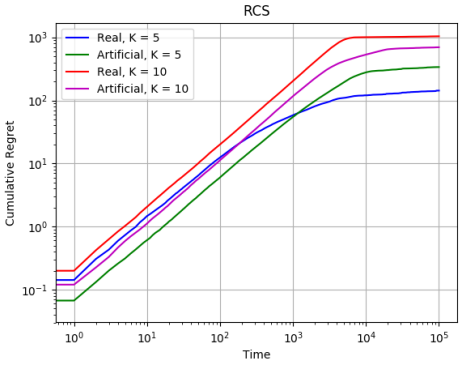
(e) Doubler



(f) SAVAGE



(g) RUCB



(h) RCS

5 Discussion

- For all the algorithms surveyed, the MSLR dataset yielded the least cumulative regret and converged the quickest consistently. This is because the dataset has arms with well separated means, which makes it 'easier' to learn than the synthetic dataset with 5 arms. It indicates that the sub-optimal gap is the most important factor in determining the performance.
- In case of Interleaved Filter, arms are eliminated only if their probability of being the best arm is outside a certain confidence interval. The artificial dataset has pairs of arms having close means, and this means that separating these two with a high confidence takes a higher number of rounds. Hence, it's regret is inversely proportional to the sub-optimal gap.
- For the RMED algorithm, the empirical divergence forces the number of comparisons between the top two arms to be low for the artificial dataset and hence takes time to converge when sub-optimal gap is low
- Doubler performs poorly under all settings because it takes time to converge. The set of left arms always contains all the arms since it takes time for the confidence bounds to shrink. Since shrinking of confidence bounds is heavily dependant on the means, lesser differences in means leads to higher regret, as shown in the plots.
- BTM also fails to achieve sub-linear regret because the set W of contesting arms often remains unchanged due to uniform exploration, especially when the number of arms is large.
- Of all the algorithms surveyed, RUCB and RCS's regret grows the slowest. Also, since the regret is sharply bounded by $Klog(T)$, the regret noticeably increases as we go from K = 5 to K = 10.

We can summarise the above discussion by mentioning the best algorithm in each environment as follows: (δ denotes sub-optimal gap)

	Small K	Large K
Small δ	RUCB/RCS	RMED
Large δ	RUCB/RCS	RMED

References

[1] *Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. Journal of Computer and System Sciences, 2012.*

[2] *Yisong Yue and Thorsten Joachims. Beat the mean bandit. In ICML, 2011.*

[3] *Junpei Komiyama, Junya Honda, Hisashi Kashima, and Hiroshi Nakagawa. Regret lower bound and optimal algorithm in dueling bandit problem. In COLT, 2015.*

[4] [Ailon et al., 2014] Nir Ailon, Zohar Karnin, and Thorsten Joachims. Reducing dueling bandits to cardinal bandits. In ICML, 2014.

[5] *[Urvoy et al., 2013] Tanguy Urvoy, Fabrice Clerot, Raphael Feraud, and Sami Naamane. Generic exploration and k-armed voting bandits. In ICML, 2013.*

[6] *Masrour Zoghi, Shimon Whiteson, Remi Munos, and Maarten de Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. In ICML, 2014.*

[7] *Masrour Zoghi, Shimon Whiteson, Maarten de Rijke, and Remi Munos. Relative confidence sampling for efficient on-line ranker evaluation. In WSDM, 2014.*

[8] *Car preference dataset*

[9] *MSLR Dataset*

[10] *Regret Lower Bound and Optimal Algorithm in Dueling Bandit Problem*