CS-747 Project Report

Syamantak Kumar 16D070025
Sushil Khyalia 160050035
Kartik Khandelwal 160070025
Karan Taneja 15D070022

November 29, 2019

Contents

1 Introduction

We explore the problem of top-K ranking in an adaptive setting with stochastic pairwise comparisons, aiming at minimising the total number of comparisons required. We consider an arbitrary instance of the problem as being given n players numbered $1, 2...n$ as part of a tournament, with the probability of the i^{th} winning over the j^{th} player being given by $p_{ij}, \forall i, j$ such that $1 \leq i, j(i \neq j) \leq n$. Note that all p_{ij} follow the constraint $p_{ij} + p_{ji} = 1, \forall i, j$. For each player i, we also define a score s_i which can be used to determine the ranking of the players. This score can be defined based on factors such as the type of tournament (league or knock-out) etc. We can use this score to define the ranking among players i.e. if $s_i > s_j$ then i^{th} player is ranked higher than j^{th}, breaking ties arbitrarily.

2 Prior/Related Work

Algorithms for determining top-K ranking in a stochastic setting have been discussed in [1] and [2]. [1] attempts to solve the top-K ranking problem for noisy (stochastic) pairwise comparisons within a tournament setting, under the ranking assumption that:

$$if\ i \succ j \implies p_{ij} > 0.5\ (p_{ij} \neq 0.5\ \forall i, j). \tag{1}$$

They provide an upper bound on the sample size: $O\Big(C\Big(n+Klog(K)\Big)max\{loglogn, logK\}\Big)$ where C is a constant dependent on the problem instance, along with an algorithm which achieves this bound. However, assumption 1 is very strict and in many practical situations it may be possible that such an order does not exist in which case the algorithm would fail. We relax assumption 1 in our experiments.

In PMLR'17, [6] proposed 'aggressive elimination' algorithm which is a recursive algorithm to determine the the top-k arms in MAB with limited rounds of adaptivity. More specifically, they use a sample complexity of $\mathcal{O}(\frac{n\log(k/\delta)}{\Delta_k^2})$ and aggressively eliminate implausible arms by recursive calls on a smaller set of arms, ultimately converging to top-k arms with probability $1-\delta$ in $log^*(n)$ rounds where log^* is iterated logarithm function. However, their approach does not provide a probably-"approximately" correctness proof, but only a probabilistic guarentee. We aim to work on providing a probablisitic guarentee of correctness and also quantify the degree of incorrectness in the result.

Top-k ranking in multi-armed stochastic bandit is a related problem with the objective to minimise the number of samples drawn from the bandit to determine the top-K arms with the highest means in the PAC setting. [3] devises an (ϵ, m, δ)-optimal algorithm in the active setting to determine (ϵ, m)-optimal arms with a probability $1-\delta$, as opposed to [5], which derives a bound on the sampling complexity for the passive setting. The active setting benefits from the using a sampling strategy that adapts itself based on observed data which leads to an improvement in the overall sampling complexity of $O\Big(H^{\epsilon/2}log(\frac{H^{\epsilon}/2}{\delta})\Big)$. We aim to use a similar approach as used in the LUCB [3] algorithm to apply to the tournament ranking problem.

3 Proposed Approach

As mentioned earlier, before devising an algorithm, we need to clearly define a scheme for scoring each player as s_i, on the basis of which the optimal ranking is defined. The criteria we selected for the purpose of our work is

$$s_i = \frac{\sum_{j=1, i\neq j}^{j=n} p_{ij}}{N-1} \tag{2}$$

where N denotes the total number of players in the tournament. This is the average probability of each player winning in a match against any other player. It can be interpreted as the probability of winning against against an opponent selected uniformly at random from the opponents available. Such a measure may be a good heuristic in the case of a league tournament where every team would

play against every other team and the winner is characterised by the most number of wins.

We then attempt to reduce an instance of the tournament ranking problem into an instance of selecting subset selection in Stochastic Multi-armed Bandits in a PAC setting. [3] provide the (ϵ, m, δ)-optimal LUCB for fully-sequential sampling to determining (ϵ, m) with probability at least $1-\delta$. They quantify both the probability of converging to the correct answer and the error made by the algorithm. Section 3.1 defines the reduction we use and the algorithm used for conducting matches in the tournament and Section 3.2 provides justification for the PAC correctness.

[4] further improves on the bounds used in [3] by providing an algorithm KL-LUCB, which uses the Kullback–Leibler divergence instead of Hoeffding's inequality. We implement both the algorithms and perform a number of experiments, described in section 4, to provide insights into the problem and give an empirical verification of the PAC correctness.

3.1 Reduction to a Multi-Armed Bandit Instance

The reduction of a tournament instance to a multi-armed stochastic bandit is as follows :

- Consider each player i of the tournament as an arm of a bandit with mean s_i as defined in 2.
- The algorithm maintains an empirical estimate, $\hat{s_i}$, for each arm, which is defined as

$$\hat{s_i} = \frac{\text{Number of Wins of player } i}{\text{Total Number of matches played by player } i} = \frac{wins_i}{pulls_i} \tag{3}$$

  where the number of matches played by player i is equal to the number of pulls of the arm corresponding to player i. This estimate is updated at each timestep, t.

- Similar to the bounds used in LUCB, we define a bound function, $\beta(u_t, t)$, such that ideally, s_i is supposed to lie between $\hat{s_i} + \beta(u_t, t)$ and $\hat{s_i} - \beta(u_t, t)$. Here, u_t is pulls-i at timestep t.
- Each pull of an arm, i, corresponds to sampling an opponent, j, uniformly at random, conducting a match between them and then updating the empirical estimate$\hat{s_i}$, the total number of pulls, $pulls_i$, and the number of wins, $wins_i$, for **player** i **only**. Therefore, the total number of matches conducted correspond to the total number of samples drawn from the bandit instance since exactly one match is played by the selected players at each timestep. The sampling complexity of the algorithm used represents the variation of the number of matches with the participating teams given (ϵ, m, δ).
- We use the same defintions for h_*^t, l_*^t and the stopping criteria as given in LUCB.

3.2 Justification of PAC Correctness

First let us show that, under the sampling strategy presented in Section 3.1, $\mathbb{E}(\hat{s_i}) = s_i$ at all timesteps. Let us consider a player i having played m_i number of matches. Let r_n, n = 1,2,...m_i, be the reward obtained by the player i at n^{th} match. $r_n = 1$ if player i won the n^{th} match and is 0 otherwise. Therefore, r_n is a bernoulli random variable, with mean $\mathbb{P}$(i winning the n^{th} match).

Let us consider the n^{th} match being played by player i.

$$\mathbb{P}(\text{i winning it's } n^{th} \text{ match}) = \sum_{j=1, i\neq j}^{j=n} \mathbb{P}(\text{i playing j in it's nth match}) * \mathbb{P}(\text{i winning against j})$$

Since the opponent, j, is being selected at random, therefore,

$$\mathbb{P}(\text{i playing j it's nth match}) = \frac{1}{N-1}$$

where N denotes the total number of players in the tournament. Also, $\mathbb{P}$(i winning against j) $= p_{ij}$ is given by the win-probability matrix defined in Section 1.

Therefore,

$$\mathbb{P}(\text{i winning the } n^{th} \text{ match}) = \frac{\sum_{j=1, i\neq j}^{j=n} p_{ij}}{N-1} = s_i \tag{4}$$

Now, the empirical estimate of the score, $\hat{s_i}$ is defined in equation 3 as

$$\hat{s_i} = \frac{\text{Number of Wins of player i}}{\text{Total Number of matches played by player i}} = \frac{\text{wins-i}}{\text{pulls-i}} = \frac{\sum_{n=1}^{n=m_i} r_n}{m_i}$$

Note that, $m_i = pulls_i$ = Total number of matches player by player i. r_n will be 0 for the matches in which player i lost and 1 for the matches in which player i won. Therefore, $\sum_{n=1}^{n=m_i} r_n$ = Number of Wins of player i.

Now,

$$\mathbb{E}(\hat{s_i}) = \mathbb{E}(\frac{\sum_{n=1}^{n=m_i} r_n}{m_i})$$

By linearity of expectation,

$$\implies \mathbb{E}(\hat{s_i}) = \frac{\sum_{n=1}^{n=m_i} \mathbb{E}(r_n)}{m_i}$$

Since each r_n is a bernoulli random variable, therefore, $\mathbb{E}(r_n) = \mathbb{P}(\text{i winning the } n^{th} \text{ match}) = s_i$ Therefore,

$$\implies \mathbb{E}(\hat{s_i}) = \frac{\sum_{n=1}^{n=m_i} s_i}{m_i}$$

$$\implies \mathbb{E}(\hat{s_i}) = \frac{s_i * m_i}{m_i} = s_i \tag{5}$$

Having established Equations 4, 5, we extend the analysis of LUCB for the bandit instance defined in Section 3.1. Note that the rewards, r_n are bounded between 0 and 1 since they are bernoulli random variables. The empirical and actual means of the arms are also bounded between 0 and 1. Since Hoeffding's inequality is the sole concentration bound used in the analysis of LUCB, therefore, proving that the rewards and the means are bounded extends the proof mentioned in [3] to our bandit instance for tournament ranking.

4 Experiments and Results

For the purpose of our experiments, we had to synthesize the win-probability matrices. We initially started by assigning probability matrices by selecting p_{ij} uniformly at random and setting $p_{ji} = 1 - p_{ij}$. However, such an assignment causes the scores $s_i = \frac{\sum_{j=1, i\neq j}^{j=n} p_{ij}}{N-1}$ to tend towards 0.5 as N increased. This happened because each p_{ij} was drawn from $U(0,1)$ therefore, as N increased, s_i started converging towards the mean of $U(0,1)$ by law of large numbers. This resulted in all the players having roughly the same score and there was very fine distinction between them making it difficult to observe the advantage provided by the adaptive sampling algorithm.

To avoid this problem, we referred to [1], which defines various models for pariwise comparisions. Under the BTL model,

$$p_{ij} = \frac{w_i}{w_i + w_j}$$

Here, w_i and w_j are a measure of the skill of the individuals on the basis of which ranking is performed. Drawing inspiration from this, we thought of defining weights w_i for each player i. These weights can be interpreted as when two players i and j play against each other, then out of $w_i + w_j$ matches, player i wins w_i matches and player j wins w_j matches. Under this assumption, we assigned p_{ij} as being drawn from $\beta(w_i + 1, w_j + 1)$. We could also have just assigned $p_{ij} = \frac{w_i}{w_i + w_j}$, however, we drew

a sample from $\beta(w_i + 1, w_j + 1)$ so that there would be variation across different instances. Observe that $\frac{w_i}{w_i+w_j}$ would be the mode of the $\beta(w_i+1, w_j+1)$ distribution. To ensure that the scores are well distributed, we assigned $w_i = i^2$ so that the players would have a significant difference in skill. The following graph compares the distribution of the scores for 50 teams following the above 2 techniques.

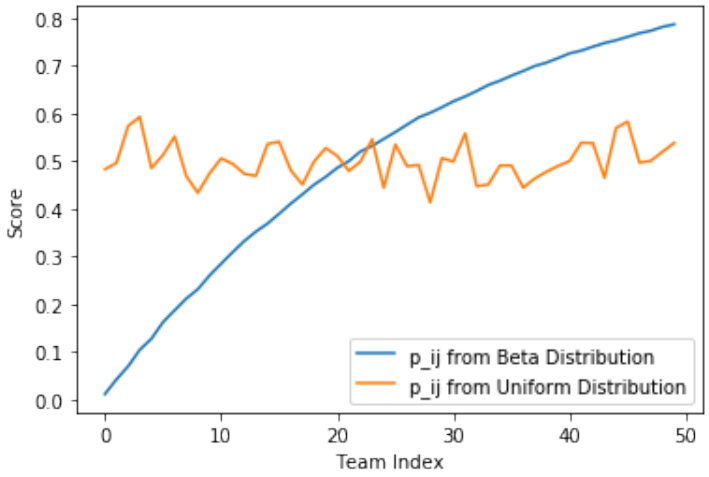

Figure 1: Distribution of scores for sampling from Uniform and Beta Distrbutions

4.1 Experiment 1

Sample complexity of LUCB and KL-LUCB with varying number of teams keeping $\epsilon = 0.1, \delta = 0.1$ We plotted the number of matches averaged over 1000 simulations. Observe that the KL-LUCB reduces the sample-complexity substantially as compared to LUCB and the difference is more clearly visible as the number of teams increase.

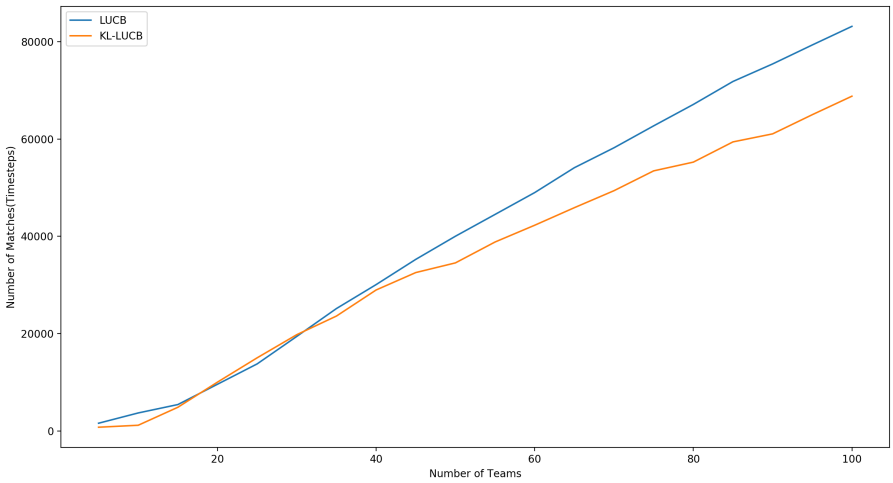

Figure 2: Number of matches averaged over 1000 simulations for LUCB and KL-LUCB algorithm varying with number of teams in the tournament.

4.2 Experiment 2

Empirical Mistake probability for LUCB and KL-LUCB algorithm as the function of number of samples for n = 20 teams with $\epsilon = 0.1, \delta = 0.1$. Even if the algorithms have a mistake probability smaller that δ, they stop much later. In the above graphs, the empirical mistake probability drops down to 0.1 after only about 80 matches.

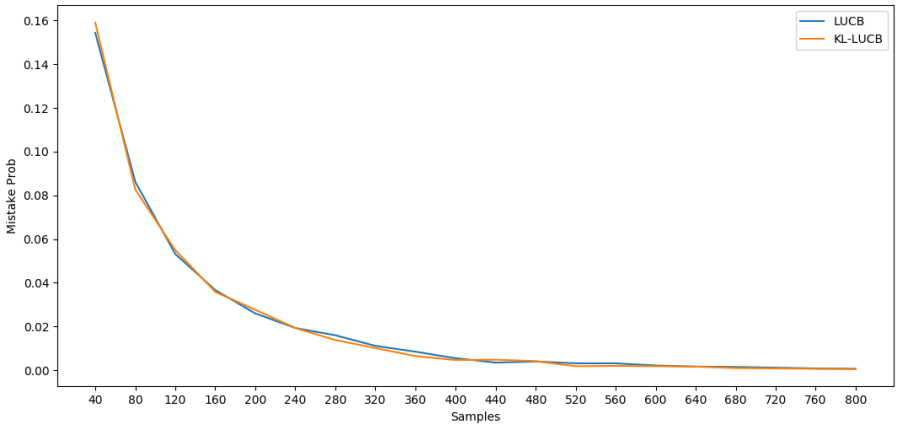

Figure 3: Fraction of wrong (k, ϵ) arms averaged over 1000 simulation at different intervals of sampling for LUCB and KL-LUCB algorithm for n=20 teams

4.3 Experiment 3

In this experiment we plot the distribution of the number of timesteps taken by the LUCB and KL-LUCB algorithms for different values of ϵ. Note that for smaller values of epsilon, both the algorithms take a greater amount of timesteps as is expected.

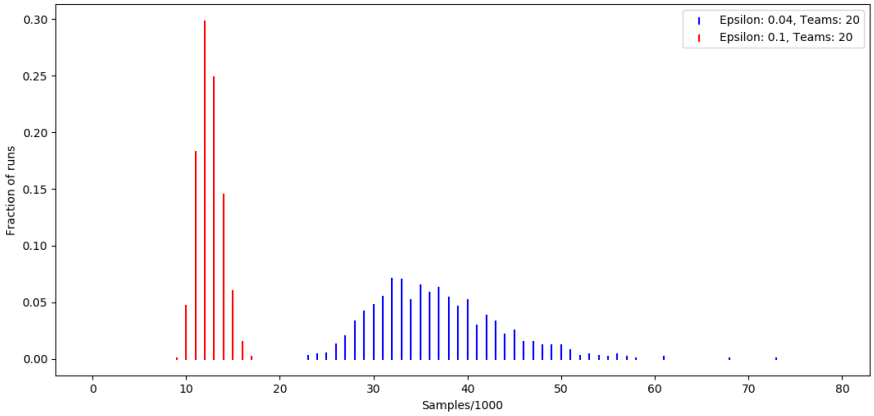

Figure 4: Fraction of runs that use a particular number of samples in 1000 simulations for different values of ϵ in LUCB algorithm

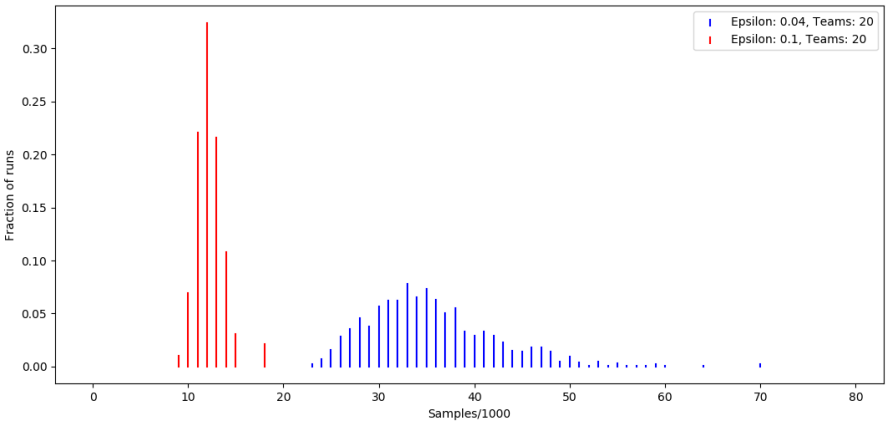

Figure 5: Fraction of runs that use a particular number of samples in 1000 simulations for different values of ϵ in KL-LUCB algorithm

4.4 Experiment 4

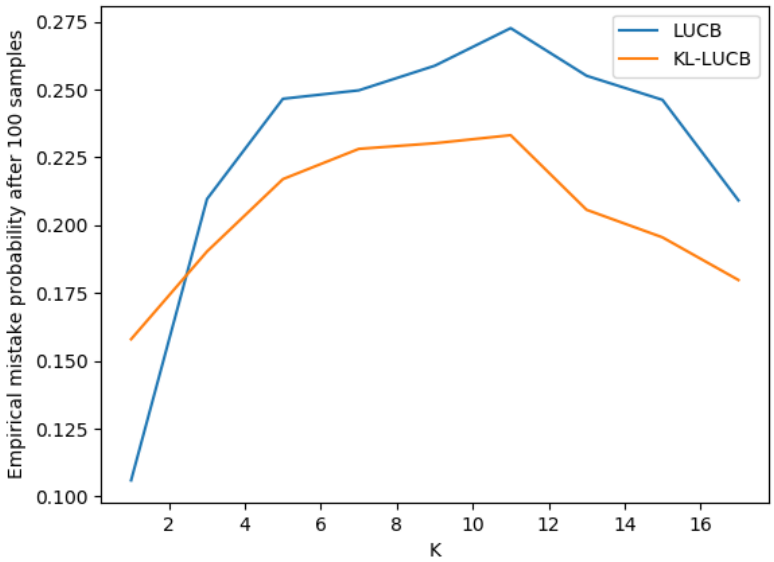

Figure 6: Empirical mistake probability after 100 samples for a competition involving 50 teams averaged over 1000 simulations

In this experiment, we plot the Empirical mistake probability when prematurely stopping after 100 samples for a competition involving 50 teams averaged over 1000 simulations where K ranges from 1 to 17. Initially, the mistake probability increases as K increases, since for a greater value of K, the algorithm would require greater number of timesteps to drive down the mistake probability. However, as K increases beyond a threshold, the mistake probability starts going down, since in essence the competitors for the controversial teams, h_*^t, l_*^t, starts decreasing. Note that the algorithm does not need to rank the top-K teams, it only needs to determine the set of top-K teams.

5 Conclusion and Future Work

We have worked on developing a fully-sequential adaptive scheme to conduct matches strategically in a tournament to determine the top-K players in a PAC setting. The algorithm proposed in this work assumes that the rewards obtained by each team on a match played is independent of the rewards obtained by other teams. However, since the values of p_{ij} and p_{ji} are constrained to sum to 1, therefore, ideally it should be possible to exploit this dependence to further to reduce the sample complexity.

6 Implementation Details

We implemented LUCB and KL-LUCB algorithms in python. The implementations can be found at the following Google Colab Notebook. The simulations were implemented in python and we used concurrent programming to generate the graphs and average over 1000 experiments. These implementations can be found at the following Github link.

References

[1] Soheil Mohajer, Changho Suh, and Adel Elmahdy, 2017
Active Learning for Top-K Rank Aggregation from Noisy Comparisons.

[2] Arun Rajkumar, 2015
Ranking from Stochastic Pairwise Preferences: Recovering Condorcet Winners and Tournament Solution Sets at the Top

[3] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone, 2012
PAC Subset Selection in Stochastic Multi-armed Bandits

[4] Emilie Kaufmann and Shivaram Kalyanakrishnan, 2013
Information Compexity in Bandit Subset Selection

[5] Shivaram Kalyanakrishnan and Peter Stone, 2010
Efficient Selection of Multiple Bandit Arms: Theory and Practice

[6] Arpit Agarwal, Shivani Agarwal, Sepehr Assadi, Sanjeev Khanna, 2017
Learning with Limited Rounds of Adaptivity: Coin Tossing, Multi-Armed Bandits, and Ranking from Pairwise Comparisons