F-036

Online Rank Aggregation

Shota Yasutake[1] Kohei Hatano[1] Eiji Takimoto[1] Masayuki Takeda[1]

[1] Department of Informatics, Kyushu University

Abstract: We consider an online learning framework where the task is to predict a permutation which represents a ranking of n fixed objects. At each trial, the learner incurs a loss defined as the Kendall tau distance between the predicted permutation and the true permutation given by the adversary. This setting is quite natural in many situations such as information retrieval and recommendation tasks. We propose algorithms for this problem and prove relative loss bounds with regret only depending on $O(n^2)$. Further, we also prove a matching lower bound of the regret, which shows our algorithms are almost optimal.

1 Introduction

The *rank aggregation* problem have gained much attention due to developments of information retrieval on the Internet, online shopping stores, recommendation systems and so on. The problem is, given m permutations of n fixed elements, to find a permutation that minimizes the sum of "distances" between itself and each given permutation. Here, each permutation represents a ranking over n elements. So, in other words, the ranking aggregation problem is to find an "average" ranking which reflects the characteristics of given rankings. In particular, the optimal ranking is called Kemeny optimal [19, 20] when the distance is defined as Kendall tau distance (which we will define later). From now on, we only consider Kendall tau distance as our distance measure.

The ranking aggregation problem is a classical problem in social choice literature which deals with voting and so on [6, 10]. These days, the rank aggregation problem also arises in information retrieval tasks such as combining several search results given by different search engines. The rank aggregation problem is being studied extensively in theoretical computer science [12, 14, 3]. It is known that the rank aggregation problem is NP-hard [5], even when $m \geq 4$ [12]. Some approximation algorithms are known as well. For example, Ailon et al. proposed a $11/7$-approximation algorithm [2]. Further, Kenyon-Mathieu and Schudy proposed a PTAS (polynomial time approximation scheme) which runs in doubly exponential in the precision parameter $\varepsilon > 0$ [21]. Ailon also gives algorithms for aggregation of partial rankings [1].

In this paper, we consider an online version of the ranking aggregation problem, which we call "online rank aggregation". This problem is about online prediction of permutations. Let S_n be the set of all permutations of n fixed elements. Then the online rank aggregation problem consists of the following protocol for each trial t:

1. The learner predicts a permutation $\widehat{\sigma}_t \in S_n$.
2. The adversary gives the learner the true permutation $\sigma_t \in S_n$.
3. The learner receives the loss $d(\sigma_t, \widehat{\sigma}_t)$.

The goal of the learner is to minimize the *cumulative regret*

$$\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t) - \min_{\sigma \in S_n} \sum_{t=1}^{T} d(\sigma_t, \sigma).$$

As there have been extensive researches on online learning with experts (e.g., Weighted Majority proposed by Littlestone and Warmuth [22] and Vovk's Aggregating Algorithm [23]), it is natural for us to apply existing algorithms for the online rank aggregation problem.

First of all, a naive method would be to apply Hedge Algorithm [15] with $n!$ possible permutations as experts. In this case, we can prove that the cumulative loss bound is at most

$$(1+\varepsilon) \min_{\sigma \in S_n} \sum_{t=1}^{T} d(\sigma_t, \sigma) + O\left(\frac{n^3 \ln n}{\varepsilon}\right),$$

for any $\varepsilon > 0$. The disadvantage of this approach is that the running time at each trial is $O(n!)$.

Next, let us consider PermELearn proposed by Helmbold and Warmuth [16]. Although this algorithm is not designed to deal with Kendall tau distance, it can use Spearman's footrule, another distance measure for permutations. It is well known that the following relationship holds for Kendall tau distance d and Spearman's footrule d_F [11]: $d(\sigma, \sigma') \leq d_F(\sigma, \sigma') \leq 2d(\sigma, \sigma')$. So, by using this relationship, we can prove that the relative loss bound of

PermELearn is at most

$$2(1+\varepsilon)\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma)+O\left(\frac{n^2\ln n}{\varepsilon}\right).$$

Its running time per trial is $\tilde{O}(n^6)$ [1].

In this paper, we propose new algorithms for online rank aggregation. For the first algorithm which we call PermRank, we prove its expected cumulative loss bound is at most

$$4(1+\varepsilon)\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma)+O\left(\frac{n^2}{\varepsilon}\right),$$

and its running time per trial is $O(n^2)$. For the second algorithm, PermRank2, we show that its expected cumulative loss is at most

$$(1+\varepsilon)\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma)+O\left(\frac{n^2}{\varepsilon}\right).$$

In particular, if we set its parameter appropriately, its expected cumulative loss is at most

$$\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma)+O(n^2\sqrt{T}).$$

We have not analyzed its efficiency so far, but, unfortunately, the time complexity is unlikely to be polynomial. This is because, as we will show later, within polynomial number of iterations, this algorithm can solve an offiline rank aggregation problem, which is NP-hard.

We further derive a lower bound of the cumulative loss of any learning algorithm for online rank aggregation, which matches our upper bound of PermuRank2. More precisely, we show that there exists a probabilistic adversary such that for any learning algorithm for online rank aggregation, the cumulative loss is at least

$$\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma)+\Omega(n^2\sqrt{T}).$$

Therefore, our algorithm PermRank2 is almost optimal [2].

[1] Main computation in PermELearn is normalization of probability matrices called Sinkhorn balancing. For this procedure, there is an approximation algorithm running in time $O(n^6\ln(n/\varepsilon))$, where $\varepsilon>0$ is a precision parameter [4].

[2] We note that there exists another approach that achieves bounds similar to ours. Kakade et al. proposed a general method to construct an online approximation algorithm, given an offline approximation algorithm as an oracle [18]. Their method is indeed applicable for online rank aggregation as well. It can be shown that by using their method and Ailon et al's approximation algorithms [2], one can obtain approximation factors 2 and 11/7 instead of ours 1 and 4, while keeping the second additive term in the same order. However, the time complexity of the conversion method at each trial depends on T, the given horizon. Further, to obtain the approximation factor 2, the space also needs T since the algorithm needs to keep all the past given permutations. On the other hand, our algorithms are more practical since time and space complexity are independent of T.

Finally, we show some experimental results on synthetic data sets. In our experiments, our algorithm PermRank performs much better than Hedge algorithm with permutations as experts and PermELearn.

2 Preliminaries

Let n be a fixed integer s.t. $n\geq 1$. Let S_n be the set of permutations on $\{1,\ldots,n\}$. The *Kendall tau distance* $d(\sigma_1,\sigma_2)$ between permutations $\sigma_1,\sigma_2\in S_n$ is defined as

$$d(\sigma_1,\sigma_2)=\sum_{i,j=1}^{n} I(\sigma_1(i)>\sigma_1(j)\wedge\sigma_2(i)<\sigma_2(j)),$$

where $I(\cdot)$ is the indicator function, i.e., $I(\text{true})=1$ and $I(\text{false})=0$. That is, Kendall tau distance between two permutations is the total number of pairs of elements for which the orders in two permutations disagree. By definition, it holds that $0\leq d(\sigma_1,\sigma_2)\leq n(n-1)/2$, and Kendall tau distance satisfies the conditions of metric.

There is another distance for permutations. The *Spearman's foot rule* between two permutations $\sigma_1,\sigma_2\in S_n$ is defined as $d_F(\sigma_1,\sigma_2)=\sum_{i=1}^{n}|\sigma_1(i)-\sigma_2(i)|$. $d(\sigma_1,\sigma_2)\leq d_F(\sigma_1,\sigma_2)\leq 2d(\sigma_1,\sigma_2)$.

Let $N=n(n-1)/2$. A *comparison vector* $\boldsymbol{q}$ is a vector in $\{0,1\}^N$. We define the following mapping $\phi: S_n\to[0,1]^N$ which maps a permutation to a comparison vector:

$$\phi(\sigma)_{ij}=\begin{cases}1 & \sigma(i)<\sigma(j),\\ 0 & \text{otherwise.}\end{cases}$$

where $i,j\in\{1,\ldots,n\}$ and $i\neq j$.

Then note that the Kendall tau distance between two permutations is represented as 1-norm distance between corresponding comparison vectors.

$$d(\sigma_1,\sigma_2)=\|\phi(\sigma_1)-\phi(\sigma_2)\|_1,$$

where 1-norm $\|\boldsymbol{x}\|_1=\sum_{i=1}^{N}|x_i|$. For example, for a permutation $\sigma=(1,3,2)$, the corresponding comparison vector is given as $\phi(\sigma)=(1,1,0)$. In general, for some comparison vectors, there is no corresponding permutation. For example, the comparison vector $(1,0,1)$ represents that $\sigma(1)>\sigma(2),\sigma(2)>\sigma(3),\sigma(3)>\sigma(1)$, for which no permutation σ exists. In particular, if a comparison vector $\boldsymbol{q}\in\{0,1\}^N$ has a corresponding permutation, we say that $\boldsymbol{q}$ is consistent. We denote $\phi(S_n)$ as the set of consistent comparison vectors in $\{0,1\}^N$.

For $p,q\in[0,1]$, the *binary relative entropy* $\Delta_2(p,q)$ between p and q is defined as $\Delta_2(p,q)=p\ln\frac{p}{q}+(1-p)\ln\frac{1-p}{1-q}$. Further, we extend the definition of the binary

Algorithm 1 PermRank

1. Let $\boldsymbol{p}_1 = (\frac{1}{2}, \ldots, \frac{1}{2}) \in [0,1]^N$.
2. For $t = 1, \ldots, T$
   (a) Choose a comparison vector $\boldsymbol{q}_t \in \{0,1\}^N$ randomly according to $\boldsymbol{p}_t$, i.e., $q_{ij} = 1$ with probability $p_{t,ij}$.
   (b) Predict a permutation $\widehat{\sigma}_t =$ **KWIKSORT**$(\boldsymbol{q}_t)$.
   (c) Get a true permutation σ_t and let $\boldsymbol{y}_t = \phi(\sigma_t)$.
   (d) Update $\boldsymbol{p}_{t+1}$ as

$$p_{t+1,i} = \frac{p_{t,i} e^{-\eta(1-y_{t,i})}}{(1-p_{t,i})e^{-\eta y_{t,i}} + p_{t,i} e^{-\eta(1-y_{t,i})}}.$$

Algorithm 2 KWIKSORT (Ailon et al. [2])

Input: a N-dimensional $\{0,1\}$-valued vector $\boldsymbol{q}$, n
Output: a permutation

1. Let S_L and S_R be empty sets, respectively.
2. Pick an integer i from $\{1, \ldots, n\}$ randomly.
3. For each $j \in \{1, \ldots, n\}$ such that $j \neq i$
   (a) If $q_{ij} = 1$ ($q_{ji} = 1$ when $j < i$), put j in S_L.
   (b) Otherwise, put j in S_R.
4. Let $\boldsymbol{q}_L, \boldsymbol{q}_R$ be the vector induced by S_L and S_R, respectively.
5. Output $(\mathbf{KWIKSORT}(\boldsymbol{q}_L), i, \mathbf{KWIKSORT}(\boldsymbol{q}_R))$.

relative entropy for vectors in $[0,1]^N$. That is, for any $\boldsymbol{p}, \boldsymbol{q} \in [0,1]^N$, the binary relative entropy is given as $\Delta_2(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^N \Delta_2(p_i, q_i)$.

3 Our Algorithm

In this section we propose our first algorithm PermRank. Our idea behind PermRank consists of two parts. The first idea is that we regard a permutation as a $N(= n(n-1)/2)$ dimensional comparison vector and deal with a problem of predicting comparison vectors. More precisely, we consider a Bernoulli trial model for each component ij of a comparison vector. In other words, for each component ij, we assume a biased coin for which head appears with probability p_{ij}, and we estimate each parameter p_{ij} in an online fashion.

The second idea is how we generate a permutation from the estimated comparison vector. As we mentioned earlier, for a given comparison vector, there might not exists a corresponding permutation. To overcome this situation, we use KWIKSORT algorithm proposed by Ailon et al. [2]. Originally, KWIKSORT is used to solve the rank aggregation problem. The basic idea of KWIKSORT is to sort elements in a brute-force way by looking at local pairwise order only. We will show later that by using KWIKSORT we can obtain a permutation whose corresponding comparison vector is close enough to the estimated comparison vector.

We describe the details of our algorithm and KWIKSORT in Algorithm 1 and Algorithm 2, respectively.

3.1 Derivation of the update

In this subsection, we derive the update rule in PermRank. The update is motivated by the following optimization problem:

$$\min_{\boldsymbol{p}} \eta \|\boldsymbol{y} - \boldsymbol{p}'\|_1 + \Delta_2(\boldsymbol{p}, \boldsymbol{p}').$$

To solve this, we use the following relationship: For any $y_i \in \{0,1\}$ and $p_i \in [0,1]$, $|y_i - p_i| = p_i(1-y_i) + (1-p_i)y_i$. Then we define the Lagrangian as

$$L(\boldsymbol{p}) = \eta \sum_i |y_i - p_i| + \sum_i \Delta_2(p_i, p_i'),$$

where $\boldsymbol{p}'$ is the probability vector before the update. By taking the partial derivative of L and enforcing the derivative to be zero, we get the update:

$$p_i = \frac{p_i' e^{-\eta(1-y_i)}}{(1-p_i')e^{-\eta y_i} + p_i' e^{-\eta(1-y_i)}}.$$

3.2 Our Analysis

Then we show our relative loss bound of PermRank.

Lemma 1. *For each $t = 1, \ldots, T$ and any comparison vector $\boldsymbol{q}$,*

$$\Delta_2(\boldsymbol{q}, \boldsymbol{p}_t) - \Delta_2(\boldsymbol{q}, \boldsymbol{p}_{t+1}) \geq -\eta \|\boldsymbol{y}_t - \boldsymbol{q}\|_1 + (1 - e^{-\eta}) \|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1.$$

Proof.

$$\begin{aligned}
&\Delta_2(q_i, p_{t,i}) - \Delta_2(q_i, p_{t+1,i}) \\
=&q_i \ln \frac{p_{t+1}}{p_{t,i}} + (1-q_i)\ln\frac{1-p_{t+1,i}}{1-p_{t,i}} \\
=&-q_i\eta(1-y_{t,i}) - (1-q_i)\eta y_{t,i} \\
&-\ln\left((1-p_i')e^{-\eta y_{t,i}} + p_i' e^{-\eta(1-y_{t,i})}\right) \\
=&-\eta|y_{t,i}-q_i| - \ln\left((1-p_i')e^{-\eta y_{t,i}} + p_i' e^{-\eta(1-y_{t,i})}\right).
\end{aligned}$$

Since $e^{-\eta y_i} = 1-(1-e^{-\eta})y_i$ for $y_i \in \{0,1\}$, the terms above becomes

$$\begin{aligned}
&\Delta_2(q_i, p_{t,i}) - \Delta_2(q_i, p_{t+1,i}) \\
=&-\eta|y_{t,i}-q_i| \\
&-\ln\left(1-(1-e^{-\eta})((1-p_{t,i})y_{t,i} + p_{t,i}(1-y_{t,i}))\right) \\
\geq&-\eta|y_{t,i}-q_i| + (1-e^{-\eta})|y_{t,i}-p_{t,i}|.
\end{aligned}$$

Finally, summing up the inequality for $i = 1, \ldots, N$, we complete the proof. □

Next, we derive an upper bound of the cumulative 1-norm loss.

Theorem 1. *For any comparison vector $\boldsymbol{q} \in \{0,1\}^N$,*

$$\sum_{t=1}^{T} \|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1 \leq \frac{\eta\sum_{t=1}^{T}\|\boldsymbol{y}_t - \boldsymbol{q}\|_1 + \frac{n(n-1)}{2}\ln 2}{1-e^{-\eta}}.$$

Proof. By summing up the inequality in Lemma 1 for $t = 1, \ldots, T$, we get that

$$\frac{\eta\sum_{t=1}^{T}\|\boldsymbol{y}_t - \boldsymbol{q}\|_1 - \Delta_2(\boldsymbol{q}, \boldsymbol{p}_{T+1}) + \Delta_2(\boldsymbol{q}, \boldsymbol{p}_1)}{1-e^{-\eta}}.$$

Since $\Delta_2(q, p_{T+1}) = 0$ and $\Delta_2(q,p) \leq \ln 2$, we complete the proof. □

Then we have the following lemma which is proved by Ailon et al.

Lemma 2 (Ailon et al.[2]). *For each trial t, let $\boldsymbol{q}_t'$ be the comparison vector corresponding to $\widehat{\sigma}_t$. Then for any $t = 1, \ldots, T$,*

$$\mathbf{E}\left[\|\boldsymbol{q}_t - \boldsymbol{q}_t'\|_1\right] \leq 3 \min_{\boldsymbol{q}\in\phi(S_n)} \|\boldsymbol{q}_t - \boldsymbol{q}\|_1,$$

where the expectation is about the randomization in KWIKSORT.

This lemma states that, even if we apply KWIKSORT for a vector $\boldsymbol{q}_t$, the resulting comparison vector $\boldsymbol{q}_t'$ is still close to $\boldsymbol{q}_t$ in terms of 1-norm distance. By using this lemma, we obtain the following useful lemma.

Lemma 3. *For any $t = 1, \ldots, T$, we have*

$$\mathbf{E}[d(\sigma_t, \widehat{\sigma}_t)] \leq 4\|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1,$$

where the expectation is defined with respect to the randomization in KWIKSORT and PermRank, respectively.

Proof. By the triangle inequality and Lemma 2, we have

$$\begin{aligned}
\mathbf{E}[\|\boldsymbol{y}_t - \boldsymbol{q}_t'\|_1] &\leq \|\boldsymbol{y}_t - \boldsymbol{q}_t\|_1 + \mathbf{E}[\|\boldsymbol{q}_t' - \boldsymbol{q}_t\|_1] \\
&\leq \|\boldsymbol{y}_t - \boldsymbol{q}_t\|_1 + 3\min_{\boldsymbol{q}\in\phi(S_n)}\|\boldsymbol{q} - \boldsymbol{q}_t\|_1 \\
&\leq \|\boldsymbol{y}_t - \boldsymbol{q}_t\|_1 + 3\|\boldsymbol{y}_t - \boldsymbol{q}_t\|_1 \\
&= 4\|\boldsymbol{y}_t - \boldsymbol{q}_t\|_1,
\end{aligned}$$

where the randomization is about KWIKSORT. Then, for any fixed $\boldsymbol{y}_{t,i} \in \{0,1\}^N$, the 1-norm distance $\|y_t - \boldsymbol{p}\|$ is linear in $\boldsymbol{p}$. Thus, we have

$$\begin{aligned}
\mathbf{E}[d(\sigma_t, \widehat{\sigma}_t)] = \mathbf{E}[\|\boldsymbol{y}_t - \boldsymbol{q}_t'\|_1] &\leq 4\mathbf{E}[\|\boldsymbol{y}_t - \boldsymbol{q}_t\|_1] \\
&\leq 4\|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1.
\end{aligned}$$

□

Now we are ready to prove the main theorem on PermRank.

Theorem 2. *The expected cumulative loss bound of PermRank is the following:*

$$\begin{aligned}
&\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t)\right] \\
\leq&\frac{4\eta\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t, \sigma) + 2n(n-1)\ln 2}{1-e^{-\eta}}.
\end{aligned}$$

Proof. By summing up the inequality in Lemma 3 for $t = 1, \ldots, T$, we obtain that

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t)\right] \leq 4\sum_{t=1}^{T}\|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1.$$

Then, by combining this inequality and Theorem 1, for any comparison vector $\boldsymbol{q} \in [0,1]^N$,

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t)\right] \leq \sum_{t=1}^{T}\frac{4\eta\|\boldsymbol{y}_t - \boldsymbol{q}\|_1 + 2n(n-1)\ln 2}{1-e^{-\eta}}.$$

Since $\boldsymbol{q}$ is arbitrary, this inequality holds when $\boldsymbol{q}$ is a consistent comparison vector that minimizes the cumulative loss in hindsight. □

In particular, when we set $\eta = 2\ln(1+\varepsilon)$, by the fact that $\eta \leq e^{\frac{\eta}{2}} - e^{-\frac{\eta}{2}}$, we get

$$\frac{\eta}{1-e^{-\eta}} \leq e^{\frac{\eta}{2}} = (1+\varepsilon), \text{ and } \frac{1}{1-e^{-\eta}} = \frac{(1+\varepsilon)^2}{\varepsilon^2+2\varepsilon},$$

respectively. Thus we have the following corollary.

Corollary 3. *For any $\varepsilon > 0$ and $\eta = 2\ln(1+\varepsilon)$,*

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t)\right] \leq 4(1+\epsilon)\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma) + O\left(\frac{n^2}{\varepsilon}\right).$$

4 Our second algorithm

In this section, we show our second algorithm PermRank2. The basic idea of PermRank2 is similar to that of PermRank, except that, in stead of applying KWIKSORT, the second algorithm uses projection techniques which are now standards in online learning researches (see, e.g.,[17, 16]). More precisely, after the update (and before applying KWIKSORT) in PermRank , PermRank2 projects the updated vector onto the convex hull of consistent permutations.

As a result, we obtain a stronger loss bound for online rank aggregation. But, unfortunately, we have not yet succeeded to prove the time and space complexity of PermRank2. A naive implementation of PermRank2 would take more than exponential time and space. The details of PermRank2 are shown in Algorithm 3.

In order to prove the cumulative loss bound of PermRank2, we will use the Generalized Pythagorean Theorem for Bregman divergences [7] (For details of the definition of Bregman divergences, see, e.g., [9]). Since the binary relative entropy is a Bregman divergence, so does our generalized version Δ_2. In the following, we show a version of the Generalized Pythagorean Theorem adapted for the binary relative entropy.

Lemma 4 (Generalized Pythagorean Theorem[7]). *Let S be a convex set in $[0,1]^N$ and $\boldsymbol{p}$ be a point in $[0,1]^N$ with strictly positive entries. Let $\boldsymbol{p}' \in S$ be the projection of $\boldsymbol{p}$ onto S in terms of Δ_2, i.e., $\boldsymbol{p}' = \arg\min_{\boldsymbol{q}\in S}\Delta_2(\boldsymbol{q},\boldsymbol{p})$. Then, for any $\boldsymbol{q} \in [0,1]^N$,*

$$\Delta_2(\boldsymbol{q},\boldsymbol{p}) \geq \Delta_2(\boldsymbol{q},\boldsymbol{p}') + \Delta_2(\boldsymbol{p}',\boldsymbol{p}).$$

In particular, if S is affine, the inequality holds with equality.

Using Lemma 4, we prove the next lemma.

Lemma 5. *For each $t = 1,\ldots,T$ and any comparison vector $\boldsymbol{q}$,*

$$\Delta_2(\boldsymbol{q},\boldsymbol{p}_t) - \Delta_2(\boldsymbol{q},\boldsymbol{p}_{t+1}) \geq -\eta\|\boldsymbol{y}_t - \boldsymbol{q}\|_1 + (1-e^{-\eta})\|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1.$$

Algorithm 3 PermRank2

1. Let $\boldsymbol{p}_1 = (\frac{1}{2},\ldots,\frac{1}{2}) \in [0,1]^N$.
2. For $t = 1,\ldots,T$
   (a) Find coefficients $\boldsymbol{\alpha}_t$ such that $\boldsymbol{p}_t = \sum_{\boldsymbol{q}\in\phi(S_n)}\alpha_{t,\boldsymbol{q}}\boldsymbol{q}$, where each $\boldsymbol{q}$ is a consistent comparison vector in $\{0,1\}^N$.
   (b) Choose a comparison vector $\boldsymbol{q}_t \in \{0,1\}^N$ randomly according to $\boldsymbol{\alpha}_t$, i.e., $q_t = \boldsymbol{q}$ with probability $\alpha_{t,\boldsymbol{q}}$.
   (c) Predict a permutation $\widehat{\sigma}_t$ = **KWIKSORT**$(\boldsymbol{q}_t)$.
   (d) Get a true permutation σ_t and let $\boldsymbol{y}_t = \phi(\sigma_t)$.
   (e) Update $\boldsymbol{p}_{t+\frac{1}{2}}$ as
   $$\boldsymbol{p}_{t+\frac{1}{2}} = \arg\min_{\boldsymbol{p}} \eta\|\boldsymbol{y}_t - \boldsymbol{p}\|_1 + \Delta_2(\boldsymbol{p},\boldsymbol{p}_t).$$
   (f) Let $\boldsymbol{p}_{t+1}$ be the projection of $\boldsymbol{p}_{t+\frac{1}{2}}$ onto the set $\mathcal{H}$ of convex combination of consistent comparison vectors:
   $$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p}\in\mathcal{H}} \Delta_2(\boldsymbol{p},\boldsymbol{p}_{t+\frac{1}{2}}).$$

Proof. By applying Lemma 4, we obtain

$$\begin{aligned}&\Delta_2(\boldsymbol{q},\boldsymbol{p}_t) - \Delta_2(\boldsymbol{q},\boldsymbol{p}_{t+1})\\ \geq&\Delta_2(\boldsymbol{q},\boldsymbol{p}_t) - \Delta_2(\boldsymbol{q},\boldsymbol{p}_{t+\frac{1}{2}}) + \Delta_2(\boldsymbol{p}_{t+1},\boldsymbol{p}_{t+\frac{1}{2}})\\ \geq&\Delta_2(\boldsymbol{q},\boldsymbol{p}_t) - \Delta_2(\boldsymbol{q},\boldsymbol{p}_{t+\frac{1}{2}}).\end{aligned}$$

Further, by Lemma 1,

$$\begin{aligned}&\Delta_2(\boldsymbol{q},\boldsymbol{p}_t) - \Delta_2(\boldsymbol{q},\boldsymbol{p}_{t+\frac{1}{2}})\\ \geq&-\eta\|\boldsymbol{y}_t - \boldsymbol{q}\|_1 + (1-e^{-\eta})\|\boldsymbol{y}_t - \boldsymbol{p}_t\|_1,\end{aligned}$$

which completes the proof. □

Again, by summing up the inequality in Lemma 5, we have the cumulative loss bound of PermRank2 as follows:

Theorem 4. *For any $\eta > 0$, the expected cumulative loss of PermRank2 is bounded as follows:*

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t,\widehat{\sigma}_t)\right] \leq \frac{\eta\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma) + \frac{n(n-1)\ln 2}{2}}{1-e^{-\eta}}.$$

Specifically, by setting $\eta = 2\ln(1+\varepsilon)$ or $\eta = \ln(1+\sqrt{1/T})$ (when T is known) we get the two corollaries.

Corollary 5. *For any $\varepsilon > 0$ and $\eta = 2\ln(1+\varepsilon)$,*

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t)\right] \leq (1+\epsilon)\min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma) + O\left(\frac{n^2}{\varepsilon}\right).$$

Corollary 6. *For $\eta = \ln(1+\sqrt{1/T})$,*

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t)\right] \leq \min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma) + O\left(n^2\sqrt{T}\right).$$

Finally, we discuss the time complexity of PermRank2. We will show that the time complexity is unlikely to be polynomial time by showing that any online algorithm that achieves the optimal cumulative loss bound can solve an offline rank aggregation problem as well. Let us fix such an online algorithm A. Given m fixed permutations, we choose a permutation uniformly randomly among them and let run A for it by $O(n^4)$ times repeatedly. Then the total rounds T is cn^4 for some constant c. For a sufficiently large $c > 0$, the average expected cumulative loss of A is that of the best permutation plus 0.5. Note that, since Kendall tau distance takes integers in $[0, n(n-1)/2]$, the average expected cumulative loss of A is exactly the same with that of the best permutation. Finally, by picking up a permutation randomly from $\widehat{\sigma}_1, \ldots, \widehat{\sigma}_T$, expected average loss of the chosen permutation is the same with that of the best permutation. So the randomized online algorithm A can probabilistically find an optimal solution of an offline rank aggregation problem in polynomial time. Since rank aggregation is NP-hard, this implies that $NP \subseteq BPP$, which is widely believed to be false. Yet, there might exist a PRAS (polynomial-time randomized approximation scheme) for online aggregation.

5 Lower bound

In this section, we derive a $\Omega(n^2\sqrt{T})$ lower bound of the regret for online rank aggregation, which shows the regret bounds for PermRank and PermRank2 are almost tight. In particular, our lower bound is obtained when the adversary is probabilistic.

Theorem 7. *For any online prediction algorithms of permutations and any integer $T \geq 1$, there exists a sequence $\sigma_1, \ldots, \sigma_T$ such that*

$$\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t) - \min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma) = \Omega(n^2\sqrt{T}).$$

Proof. The proof partly follows a well known technique in [8, 9]. We consider the following strategy of the adversary: At each trial t, give the learning algorithm either the permutation $\widehat{\sigma}_t = \sigma^1 = (1, ..., n)$ or $\widehat{\sigma}_t = \sigma^0 = (n, n-1, ..., 1)$ randomly with probability half. Note that the corresponding comparison vectors are $\phi(\sigma^0) = (0, ..., 0)$ and $\phi(\sigma^1) = (1, ..., 1)$, respectively.

Then, for any $t \geq 1$ and any permutation σ_t, $\mathbf{E}[d(\sigma_t - \widehat{\sigma}_t)] = \binom{n}{2}/2$. This implies that the expected cumulative loss of any learning algorithm is exactly $\frac{\binom{n}{2}}{2}T$, because of the linearity of the expectation.

Next, we consider the expected cumulative loss of the best of σ^0 and σ^1, that is, $\mathbf{E}\left[\min_{i=0,1}\sum_{t=1}^{T} d(\sigma_t - \sigma^i)\right]$. By our construction of the adversary, this expectation is reduced to

$$\mathbf{E}\left[\min_{p=0,1}\sum_{t=1}^{T} d(\sigma_t - \sigma^p)\right] = \binom{n}{2}\mathbf{E}_{y_1,\ldots,y_T}\left[\min_{p=0,1}\sum_{t=1}^{T}|p-y_t|\right],$$

where $y_1, \ldots, y_T$ are independent random $\{0,1\}$-variables. The above expectation can be further written as

$$\binom{n}{2}\mathbf{E}_{y_1,\ldots,y_T}\left[\min_{p=0,1}\sum_{t=1}^{T}|p-y_t|\right] = \frac{\binom{n}{2}T}{2} - \frac{\binom{n}{2}}{2}\mathbf{E}_{y_1,\ldots,y_T}\left[|(\#\text{ of 0s}) - (\#\text{ of 1s})|\right].$$

Then the second term in the last equality is bounded as $-\binom{n}{2}\Omega(T)$. Finally, we have

$$\mathbf{E}\left[\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t) - \min_{p=0,1}\sum_{t=1}^{T} d(\sigma_t,\sigma^p)\right] \geq \Omega(n^2\sqrt{T}).$$

So, there exists a sequence $\sigma_1, \ldots, \sigma_T$ such that

$$\sum_{t=1}^{T} d(\sigma_t, \widehat{\sigma}_t) \geq \min_{p=0,1}\sum_{t=1}^{T} d(\sigma_t,\sigma^p) + \Omega(n^2\sqrt{T}) \geq \min_{\sigma\in S_n}\sum_{t=1}^{T} d(\sigma_t,\sigma) + \Omega(n^2\sqrt{T}).$$

□

Note that, by Corollary 6, the cumulative loss of PermRank2 matches our lower bound.

6 Experiments

We show preliminary experimental results for artificial data. The algorithms we examine are Hedge Algorithm, PermELearn and PermRank.

For our artificial data, we specify the following way of generating permutations. First we fix a base permutation in S^n. Then, at each trial, pick up a pair over n elements randomly and reverse the order of the pair in the fixed permutation. After repeating this procedure s times, give each learning algorithm the resulting permutation. In our experiments, we fix $n = 9$,$s = 0, 5, 10$, and $T = 600$, respectively.

We run Hedge algorithm with $n!$ permutations as experts and PermRank. For these algorithms, the parameter η is specified as $\eta = 2\ln(1+\varepsilon)$, where $\varepsilon = 0.01$. We also compare them with the predictor that always predicts the base permutation. We regard this predictor as the best permutation. Note that each learning algorithm is probabilistic. So we repeat running each algorithm 5 times for the fixed sequence of permutations, and compute the average of cumulative losses.

We the results in Figure 1. As can bee seen in Figure 1, the cumulative losses of PermRank are smaller than those of Hedge algorithm and PermELearn for each choice of s. Also, PermRank is competitive with the best permutation especially for $s = 5, 10$.

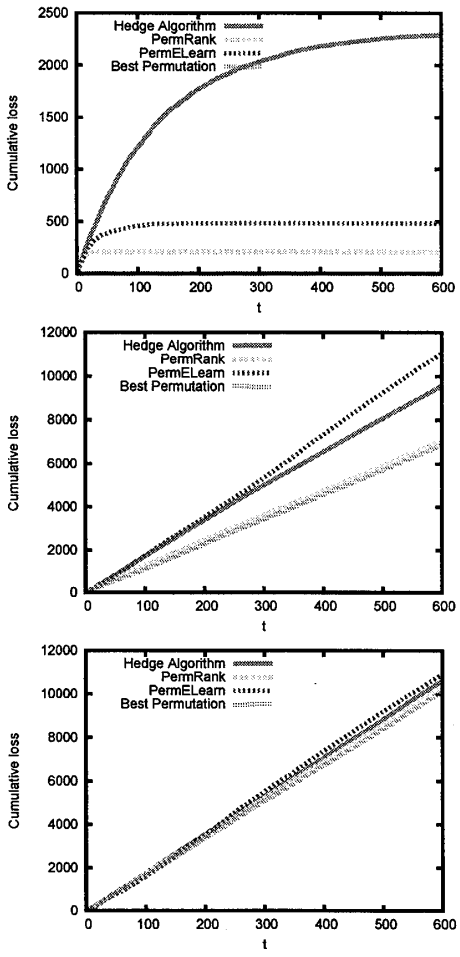

Figure 1: Cumulative losses of Hedge, PermELearn, PermRank, and the best permutation for $s = 0$ (upper left), $s = 5$ (upper right), and $s = 10$ (lower).

7 Conclusions and Future Work

In this paper, we consider online rank aggregation, the online version of the rank aggregation problem. We proposed two online learning algorithms PermRank and PermRank2 for online rank aggregation and prove their cumulative loss bounds. Then we prove a lower bound for online rank aggregation which matches the upper bound of PermRank2. Finally, our experimental results show that PermRank performs much better than the naive implementation of Hedge algorithm with $n!$ permutations as experts.

There are some open questions.

1. Does there exist a polynomial time prediction algorithm for online rank aggregation with $(1+\varepsilon)$ approximation factor?
2. Can we generalize our results for partial rankings such as top k lists?

Especially, the second problem is important in practice (see, e.g., [1, 13] for researches on partial ranking).

References

[1] N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. *Algorithmica*, 57(2):284–300, 2008.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5), 2008.

[3] A. Andoni, R. Fagin, R. Kumar, M. Patrascu, and D. Sivakumar. Corrigendum to "efficient similarity search and classification via rank aggregation" by ronald fagin, ravi kumar and d. sivakumar (proc. sigmod'03). In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1375–1376, 2008.

[4] H. Balakrishnan, I. Hwang, and C. J. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *43rd IEEE Conference on Decision and Control*, pages 4874–4879, 2004.

[5] J. Bartholdi, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice ad Welfare*, 6:157–165, 1989.

[6] J. C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.

[7] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

[8] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44(3):427–485, 1997.

[9] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[10] M. J. Condorcet. Éssai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, 1785.

[11] P. Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(2):262–268, 1977.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference (WWW'01)*, pages 613–622, 2001.

[13] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing partial rankings. *SIAM Journal on Discrete Mathematics*, 20(3):628–648, 2006.

[14] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 301–312, 2003.

[15] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[16] D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.

[17] M. Herbster and M. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[18] S. Kakade, A. T. Kalai, and L. Ligett. Playing games with approximation algorithms. In *Proceedings of the 39th annual ACM symposium on Theory of Computing (STOC'07)*, pages 546–555, 2007.

[19] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959.

[20] J. G. Kemeny and J. Snell. *Mathematical Models in the Social Sciences*. Blaisdell, 1962. (Reprinted by MIT Press, Cambridge, 1972.).

[21] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07)*, pages 95–103, 2007.

[22] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[23] V. Vovk. Aggregating strategies. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 371–386, 1990.