On Sample Complexity Upper and Lower Bounds for Exact Ranking from Noisy Comparisons

Wenbo Ren, The Ohio State University, Dept. CSE, ren.453@osu.edu
Jia Liu, Iowa State University, Dept. CS, jialiu@iastate.edu
Ness B. Shroff, The Ohio State University, Dept. ECE and CSE, shroff.11@osu.edu

THE OHIO STATE UNIVERSITY

IOWA STATE UNIVERSITY

Introduction

Problem

- **Items**. n items indexed by $1, 2, 3, ..., n$.
- **Preferences**. Users have preferences over items.
- **Noisy Comparisons**. Comparisons are noisy. Each comparison over some items returns a noisy result about the most preferred item.
- **Active ranking**. Adaptively select items to compare according to past observations.
- **Goal**. i) To fully rank n items and ii) use as few comparisons as possible.

Motivations

- **Active ranking v.s. passive ranking**. Passive ranking: first have comparison results, then deduce the ranking, higher sample complexity. Active ranking: Adaptively choose items to compare, needs less samples.
- **Applications**. For instance, a server of an app can adaptively choose items to present to the users and collect the feedback, and learn the users' preferences in shorter time.
- **Exact ranking vs PAC ranking**. i) For applications such that a tiny error can cause a huge loss; ii) To obtain instance-wise upper and lower bounds but not worst-case ones. For some instances, bounds are better than the PAC ones.

Notations

- **Preferred**. $i \succ j$ if i is more preferred than j.
- **Winning probability**. If we compare i and j, then i wins with probability $p_{i,j}$.
- **Confidence** $\delta \in (0, 1/2)$. The error probability should be no more than δ.
- **Gaps**. $\Delta_{i,j} := |p_{i,j} - 1/2|$; $\Delta_i = \min_{j \neq i} \Delta_{i,j}$; $\tilde{\Delta}_i := \min\{\Delta_{i,j}$, where there is no item k has $i \succ k \succ j$ or $j \succ k \succ i\}$.

Assumptions

- **Independence**. The comparisons are independent across time and items.
- **Unique true ranking.** There is a unique true ranking $r_1 \succ r_2 \succ \cdots \succ r_n$, and this ranking is unknown.
- **Weak stochastic transitivity**. Item $i \succ j$ if and only if $p_{i,j} > 1/2$.

Lower bound

Definition 1 (δ-correct algorithms)**.** *An algorithm is said to be δ-correct for a problem if for any input instance of this problem, it, with probability at least $1 - \delta$, returns a correct result in finite time.*

Generic lower bound

Theorem 2 (Lower bound for pairwise ranking)**.** *Given $\delta \in (0, 1/12)$ and an instance $\mathcal{I}$ with n items, then the number of comparisons used by a δ-correct algorithm $\mathcal{A}$ with no prior knowledge about the gaps of $\mathcal{I}$ is lower bounded by*

$$\Omega\Big(\sum_{i\in[n]}[\tilde{\Delta}_i^{-2}(\log\log\tilde{\Delta}_i^{-1} + \log(1/\delta))] + \min\{\sum_{i\in[n]}\tilde{\Delta}_i^{-2}\log(1/x_i) : \sum_{i\in[n]} x_i \leq 1\}\Big). \quad (1)$$

If $\delta \preceq 1/poly(n)$, or $\max_{i,j\in[n]}\{\tilde{\Delta}_i/\tilde{\Delta}_j\} \preceq n^{1/2-p}$ for some constant $p > 0$, then lower bound is

$$\Omega\big(\sum_{i\in[n]}\tilde{\Delta}_i^{-2}(\log\log\tilde{\Delta}_i^{-1} + \log(n/\delta))\big). \quad (2)$$

Remark. If the model satisfies strong stochastic transitivity (SST, which means $i \succ j \succ k$ implies $p_{i,k} \geq \max\{p_{i,j}, p_{j,k}\}$), then Eq. (2) is tight (up to a constant factor).

Multinomial logit model

- **Preference score**. Each item i holds a number $\theta_i > 0$. Larger θ_i implies more preferred.
- **Comparison marginal**.

$$p_{i,j} = \theta_i/(\theta_i + \theta_j).$$

- **Listwise comparison**. The comparison over m items $S = \{i_1, i_2, ..., i_m\}$ returns item i with probability

$$p_{i,S} = \theta_i/(\theta_{i_1} + \theta_{i_2} + \cdots + \theta_{i_m}).$$

- **Gaps**. For pairwise ranking, the gap is the same as the generic model. For listwise ranking, define $\Delta_{i,j} := |\theta_i/(\theta_i + \theta_j) - 1/2|$. Also, we have $\tilde{\Delta}_i = \Delta_i$ for all items i under the MNL model, i.e., it satisfies SST.

Theorem 3. *Under the MNL model, Given an algorithm knows that the instance satisfies the MNL model and can perform m-wise comparisons for all $m \in \{2, 3, ..., n\}$, the **sample complexity lower bound is the same as Theorem 1**.*

Algorithm

Basic idea

- **Previous works**. If Δ_i is priorly known, then we can insert item i to a sorted list with probability $1 - \delta_i$ by $O(\Delta_i^{-2}\log(1/\delta_i))$ comparisons.
- **But Δ_i's are unknown**. Method: take a guess ϵ of Δ_i, and attempts to insert item i.
- **If guess is not good**, then the algorithm does not get wrong result with a large probability.
- **If guess if good**, then the algorithm can get the correct result with a large probability.
- **Our method**. Gradually decrease the guess value and choose a proper confidence for each guess.

Subroutine: Attempting-Comparison

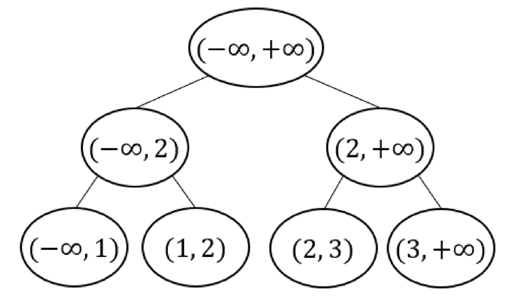
Algorithm 1 Attempting-Comparison(i, j, ϵ, δ) (ATC)

Initialize: $\forall t$, let $b^t = \sqrt{\frac{1}{2t}\log\frac{\pi^2 t^2}{3\delta}}$; $b^{max} \leftarrow \lceil\frac{1}{2\epsilon^2}\log\frac{2}{\delta}\rceil$; $w_i \leftarrow 0$;
Goal: Attempts to order i and j with ϵ, a guess of $\Delta_{i,j}$.

for $t \leftarrow 1$ to b^{max} **do**
  Compare i and j once; Update $w_i \leftarrow w_i + 1$ if i wins;
  Update $\hat{p}_i^t \leftarrow w_i/t$;
  if $\hat{p}_i^t > 1/2 + b^t$ **then return** i;
  if $\hat{p}_i^t < 1/2 - b^t$ **then return** j;
end for
return i **if** $\hat{p}_i^t > 1/2$; **return** j **if** $\hat{p}_i^t < 1/2$; and **return** a random item **if** $\hat{p}_i^t = 1/2$;

Lemma 4 (Theoretical Performance of ATC)**.** *Sample complexity of ATC is $O(\epsilon^{-2}\log(1/\delta))$. It returns the more preferred item with probability $\geq 1/2$. Further, if $\epsilon \leq \Delta_{i,j}$, then ATC returns the more preferred item with probability $\geq 1 - \delta$.*

Random walk on a tree



- **Preference Interval Tree** (Feige et al, 1994) is an interval tree constructed from a sort list of items.
- Algorithm performs a random walk on the tree.
- Each node holds lchild, rchild, parent, left, right, mid. Item i is in (left, right) iff right $\succ i \succ$ left. We also have u.left $= u$.lchild.left, u.right $= u$.rchild.right and u.mid $= u$.lchild.right $= u$.rchild.left.
- If $\epsilon \leq \Delta_i$, then for each iteration, X goes to the "right" direction with probability at least $q = 15/16$, which guarantees the correctness.
- If $\epsilon > \Delta_i$, then for each iteration, X goes to the "wrong" direction with probability at most $1/2$, which ensures i will not be inserted to a wrong place with probability at least $1 - \delta$.

Algorithm 2 Basic idea of Attempting-Insertion(i, S, ϵ, δ) (ATI).

Let T be a PIT of S, $h \leftarrow \lceil 1 + \log_2(1 + |S|)\rceil$, the depth of T; For all leaf nodes u of T, set $c_u \leftarrow 0$; Set $t^{\max} \leftarrow \lceil\max\{4h, \frac{512}{25}\log\frac{2}{\delta}\}\rceil$; $q \leftarrow 15/16$;

$X \leftarrow$ the root node of the PIT of S;
for $t \leftarrow 1$ to t^{max} **do**
  Use ATC to check $X \in (X.\text{left}, X.\text{right})$ with confidence $q^{2/3}$;
  if The answer is yes **then**
    if X is a leaf node **then**
      $c_X \leftarrow c_X + 1$;
      if $c_X > \frac{1}{2}t + \sqrt{\frac{t}{2}\log\frac{\pi^2 t^2}{3\delta}} + 1$ **then**
        Insert i to X and return *inserted*;
    else if ATC returns $i \succ X.\text{mid}$ with confidence $q^{1/3}$ **then**
      $X \leftarrow X.\text{rchild}$;
    else $X \leftarrow X.\text{lchild}$;
  else
    if X is a leaf node and $c_X > 0$ **then**
      $c_X \leftarrow c_X - 1$;
    else $X \leftarrow X.\text{parent}$;
end for
if there is a leaf node u with $c_u \geq 1 + \frac{5}{16}t^{\max}$ **then**
  Insert i into the corresponding interval of u and **return** *inserted*;
else return *unsure*;

Upper bound

Algorithm 3 Iterative-Attempting-Insertion (IAI).

Input parameters: (i, S, δ);
Initialize: For all $\tau \in \mathbb{Z}^+$, set $\epsilon_\tau = 2^{-(\tau+1)}$ and $\delta_\tau = \frac{6\delta}{\pi^2\tau^2}$; $t \leftarrow 0$; $Flag \leftarrow$ *unsure*;

repeat $t \leftarrow t + 1$;
  $Flag \leftarrow$ATI$(i, S, \epsilon_t, \delta_t)$;
until $Flag =$ *inserted*

Algorithm 4 Iterative-Insertion-Ranking (IIR).

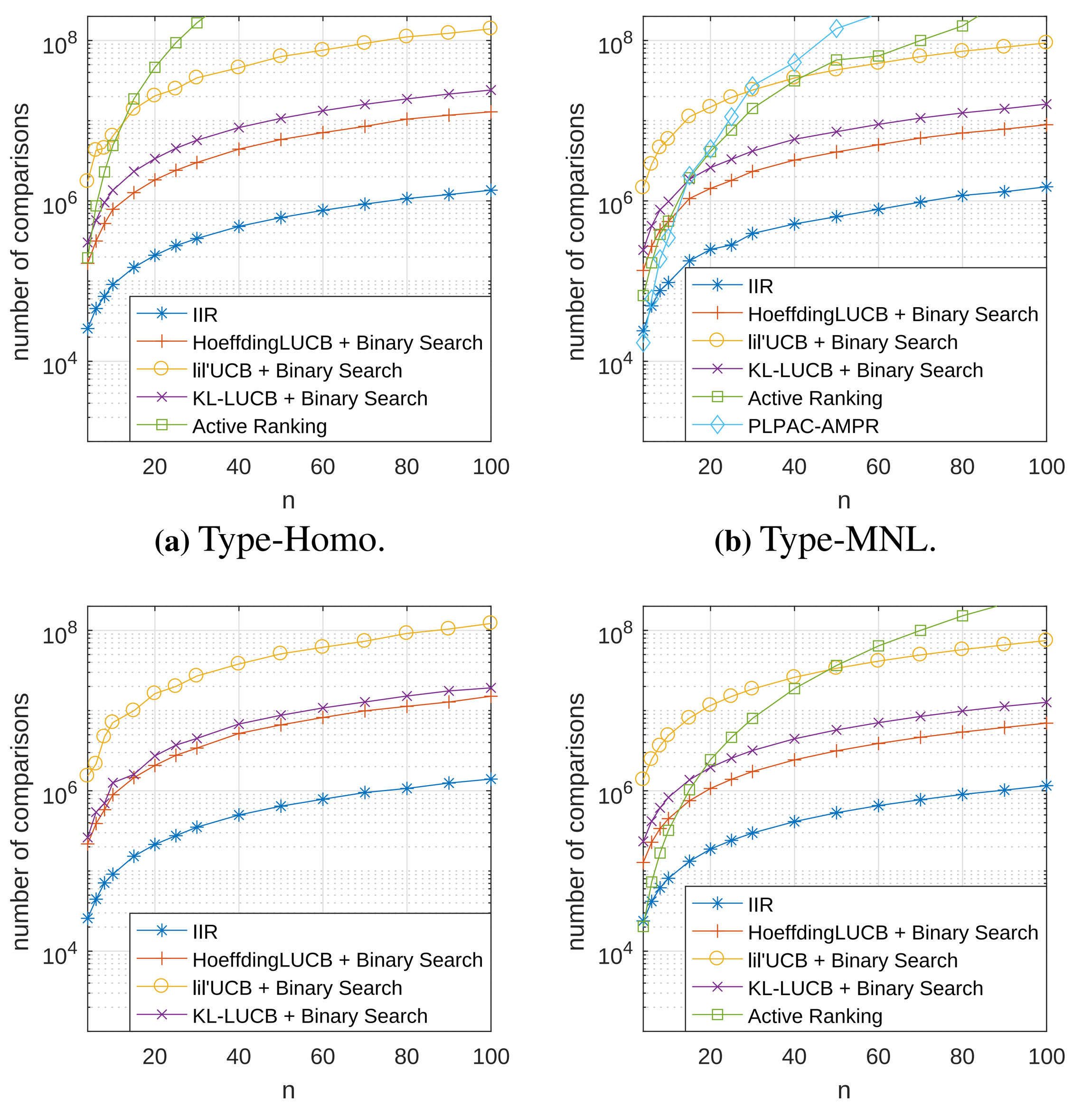
Input: $S = [n]$, and confidence $\delta > 0$;

$Ans \leftarrow$ the list $[S[1]]$;
for $t \leftarrow 2$ to $|S|$ **do**
  IAI$(S[t], Ans, \delta/(n-1))$;
end for
return Ans;

Theorem 5 (Theoretical Performance of IIR)**.** *With probability at least $1 - \delta$, IIR returns the exact ranking of $[n]$ and conducts at most $O(\sum_{i\in[n]}\Delta_i^{-2}(\log\log\Delta_i^{-1} + \log(n/\delta)))$ comparisons.*

Numerical Results

- $\Delta = 0.1$ and $\delta = 0.01$. Repeat 100 times and take average.
- **Type-Homo**. For all $i \succ j$, $p_{i,j} = 1/2 + \Delta$.
- **Type-MNL**. MNL model, and preference score of item i is θ_i, drawn from Uniform($[0.9 * 1.5^{n-i}, 1.1 * 1.5^{n-i}$).
- **Type-Random**. For all $i \succ j$, probability $p_{i,j}$ is drawn from Uniform($[0.5 + 0.8\Delta, 0.5 + 1.5\Delta]$).
- **Type-Easy**. For all $i \succ j$, if there is an item k such that $i \succ k \succ j$, then $p_{i,j} = 1$. Otherwise, $p_{i,j} = 1/2 + \Delta$.



(a) Type-Homo. **(b)** Type-MNL. **(c)** Type-Random. **(d)** Type-Easy.

Figure 1: Comparisons between IIR and existing methods.

Acknowledgments. This work has been supported in part by NSF grants ECCS-1818791, CCF-1758736, CNS-1758757, CNS-1446582, CNS-1901057; ONR grant N00014-17-1-2417; AFRL grant FA8750-18-1-0107, and by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT), (2017-0-00692, Transport-aware Streaming Technique Enabling Ultra Low-Latency AR/VR Services).