Passive and Active Ranking from Pairwise Comparisons

Songbai Yan*

University of California, San Diego

Abstract

In the problem of ranking from pairwise comparisons, the learner has access to pairwise preferences among n objects and is expected to output a total order of these objects. This problem has a wide range of applications not only in computer science but also in other areas such as social science and economics. In this report, we will give a survey of passive and active learning algorithms for ranking from pairwise comparisons. We will first survey algorithms for learning to rank in a general setting, and then we will discuss ranking problems with special structures like bounded noise, the Bradley-Terry-Luce model, and embedding in a low dimensional Euclidean space. Finally, we will briefly survey efficient algorithms for practical applications.

1 Introduction

Ranking (a.k.a. learning to rank) is, given some information about a set of objects, to output a total order of these objects. This problem has a wide range of applications including data mining, information retrieval, as well as economics and sociology [Ail13]. For example, in information retrieval, the search engine needs to list the results for user queries in a good order such that the most relevant links should be at the top of the page. In sociology, there has been a long line of research in how to design a voting scheme such that election results reflect true preferences of voters.

The task of learning to rank is closely related to some problems in computer science. It is similar to a combinatorial optimization problem called *Minimum Feedback Arc Set in Tournaments* which has been proved to be NP-hard [Alo06], but admits a Polynomial-Time Approximation Scheme [KMS07]. [ACN08] applies the idea in [KMS07] to design an active ranking algorithm. Learning to rank is also similar to the problem of sorting from noisy information. Many noisy sorting algorithms are also efficient active ranking algorithms [BM08, MG15b]. Many works also study the relation between ranking and other machine learning problems like binary classification [AM07, BBB$^+$08] and multiclass/multilabel classification [DSM03, HFCB08].

In machine learning, people study ranking problems from various kinds of information including pointwise information, pairwise information, and listwise information [Liu09]. In this report, we will focus on learning to rank from pairwise information, where learner only has access to pairwise preferences among objects.

The reasons for considering pairwise comparisons are as follows. First, pairwise comparisons are easier to obtain than the total order. For example, for movie recommendation, it would take a lot of time to let users give a total order, or even a partial order, of a list of movies, but it will not take too much time to let users make some pairwise comparisons. Second, in many cases, pairwise comparisons are more accurate than pointwise information. Considering movie recommendation again, it would be hard for a user to accurately quantize to which extent he/she likes a movie, but it would be easier to decide which one is preferred

*Email: `yansongbai@eng.ucsd.edu`

between two objects. Moreover, many studies show that pairwise comparison results are more reliable in many applications like clickthrough data [Joa02] and crowd sourcing [YJJJ13].

In this report, instead of time complexity, we focus on sample complexity which is defined as how many comparisons are required to find a good total order. When comparisons are noiseless, in order to rank n objects, the optimal sample complexity is $O(n \log n)$ which can be achieved by any comparison-based sorting algorithms that run in $O(n \log n)$ time. This report will address the problem in more complicated settings where comparisons are noisy, or the objects to rank have certain special structure (for instance, a low dimensional embedding).

We will survey and compare both passive ranking and active ranking algorithms. The difference between the two paradigms is that in passive ranking, the learner will receive a set of independent and identically distributed (i.i.d.) pairs and their comparison results as training data at once, and use these data to learn a total order, whereas in active ranking, the learner can *interactively* choose which pair to compare, and this choice can depend on the previous queries and responses. Thus it is reasonable to expect that active ranking will require less comparisons because the interactive process allows the learner to make use of comparison results more effectively.

This report is organized as follows: in Section 2, we will give the formal setting of learning to rank from pairwise comparisons; in Section 3, we will compare our ranking problem with related ranking models; in Section 4, we will survey algorithms and analyses for learning to rank in the general setting where few assumptions are made on the objects to rank or the comparison process; in Section 5, we will survey the learning algorithms for learning to rank with special structures, including bounded noise model, the Bradley-Terry-Luce model, and embedding in a low dimensional Euclidean space; in Section 6, we will briefly survey the efficient algorithms (without theoretical guarantee) for practical applications; finally Section 7 will summarize the report and discuss open problems for future research.

2 Problem definition

2.1 Notations

Let V be a set of n elements $\{v_1, \dots, v_n\}$. The input space $\mathcal{X} = V \times V$ contains all pairs of V. The label space $\mathcal{Y}$ is defined as $\{0, 1\}$ which represents the comparison result of a pair. For any *unordered pair* $(u, v) \in \mathcal{X}$, its label is 1 when u *ranks higher than* v and 0 otherwise. We will also use $u \prec v$ and *ordered pair* $\langle u, v \rangle$ to represent u *ranks higher than* v.

Denote by $\binom{V}{2}$ the set of pairs from V. Denote by $\mathcal{D}$ the underlying distribution over $\mathcal{X} \times \mathcal{Y}$, by $\mathcal{D}_\mathcal{X}$ the marginal distribution over $\mathcal{X}$, and by $\mathcal{D}|_x$ the conditional distribution of the label given a sample x. We assume $\mathcal{D}_\mathcal{X}$ is uniform over $\binom{V}{2}$.

The hypothesis space $\mathcal{H}$ is the set of all permutations of V. For a permutation π, with some abuse of notations, we make following definitions. For any pair $u, v \in V$, define $\pi(u, v) = 1$ if u is before v in π (in other words, u ranks higher than v), and 0 otherwise. Define $\pi(i)$ to be the index of v_i in π, and define $\pi^{-1}(i)$ to be the rank i element (in other words, π^{-1} is the inverse function of π, $\pi(\pi^{-1}(i)) = i$). Define π_i^j to be the ordered sequence $\pi^{-1}(i), \pi^{-1}(i+1), \dots, \pi^{-1}(j)$. For example, for permutation $\pi = \langle 3, 5, 4, 1, 2 \rangle$, by definition, $\pi(5, 2) = 1$, $\pi(2, 5) = 0$, $\pi(3) = 1$, $\pi(4) = 3$, $\pi^{-1}(1) = 3$, $\pi^{-1}(3) = 4$, and $\pi_2^4 = \langle 5, 4, 1 \rangle$.

We define the indicator function $\mathbb{I}[A]$ to be 1 if A is true, and 0 otherwise.

For any permutations π_1, π_2, define $\mathrm{er}_\mathcal{D}(\pi_1) = \mathbb{E}_{\langle u,v \rangle \sim \mathcal{D}} \mathbb{I}[\pi_1(u, v) \neq 1]$, $\Pr(\pi_1 \neq \pi_2) = \Pr_{(u,v) \sim \mathcal{D}_\mathcal{X}} (\pi_1(u, v) \neq \pi_2(u, v))$, and $\nu = \min_{\pi \in \mathcal{H}} \mathrm{er}_\mathcal{D}(\pi)$.

2.2 Passive ranking and Active ranking

In this report, we will survey both *passive ranking* algorithms and *active ranking* algorithms, which are defined as follows:

Passive ranking The input is a set of i.i.d. samples from $\mathcal{D}$, and the output is some permutation $\pi \in \mathcal{H}$ that minimizes the loss function (to be specified later).

Active ranking There is an Oracle that on query $x \in \mathcal{X}$ returns a label $y \in \mathcal{Y}$ drawn from distribution $\mathcal{D}|_x$. The algorithm can interactively query any $x \in \mathcal{X}$ and receive its label from the Oracle. Each query can be dependent with previous queries and Oracle responses. The output should be some permutation $\pi \in \mathcal{H}$ that minimizes the loss function (to be specified later).

Throughout this report, instead of focusing on *computational complexity,* we mostly care *sample complexity* which measures how many comparisons are needed to guarantee a given loss. More formally, for a loss function $L : \mathcal{H} \to [0, 1]$, for any $\epsilon, \delta \in (0, 1)$, for any learning algorithm $\mathcal{A}$, when $\mathcal{A}$ is a passive ranking algorithm, its sample complexity is defined as the minimum natural number $m(\epsilon, \delta, n, \mathcal{A})$ such that with probability at least $1 - \delta$, if $\mathcal{A}$ receives at least $m(\epsilon, \delta, n, \mathcal{A})$ pairwise comparison results drawn i.i.d. from $\mathcal{D}$, then it can output a permutation with $L(\pi) \leq \epsilon$; when $\mathcal{A}$ is an active ranking algorithm, its sample complexity is defined as the minimum natural number $m(\epsilon, \delta, n, \mathcal{A})$ such that with probability at least $1 - \delta$, if $\mathcal{A}$ makes at least $m(\epsilon, \delta, n, \mathcal{A})$ queries for pairwise comparisons, then it can output a permutation with $L(\pi) \leq \epsilon$.

2.3 Different loss functions

There are many different loss functions for the task of learning to rank. For learning to rank in the general setting discussed in Section 4, the loss function we use is $L(\pi) = \mathrm{er}_{\mathcal{D}}(\pi)$, which is a variant of Kendall-τ distance defined as $d_K(\pi_1, \pi_2) = \sum_{(i,j)} \mathbb{I}[\pi_1(i, j) \neq \pi_2(i, j)]$ for any two permutations π_1, π_2. Kendall-τ distance counts how many pairs are ordered differently by the two permutations. There are also other distances between permutations. The Spearman Footrule distance is defined as $d_F(\pi_1, \pi_2) = \sum_i |\pi_1(i) - \pi_2(i)|$, which measures the accumulated displacement of objects. [DG77] proves that $\forall \pi_1, \pi_2$, $d_K(\pi_1, \pi_2) \leq d_F(\pi_1, \pi_2) \leq 2d_K(\pi_1, \pi_2)$, so asymptotically Spearman Footrule distance is equivalent to Kendall-τ distance. Some papers (for example [BM08, WJJ13]) also consider the distance $d(\pi_1, \pi_2) = \max_i |\pi_1(i) - \pi_2(i)|$, which measures uniform displacement instead of accumulated displacement in Spearman Footrule distance.

In Section 5, we will discuss some learning to rank problems where there exists a true total order π^*. For these problems, the loss function we use is $L(\pi) = \Pr(\pi \neq \pi^*)$ which is the probability that π orders a pair differently from how the ground truth permutation will order. By triangle inequality, for any $\pi \in \mathcal{H}$ $L(\pi) \leq \mathrm{er}_{\mathcal{D}}(\pi) + \nu$, but this bound is often quite loose.

3 Related problems

3.1 Learning to rank and Minimum Feedback Arc Set in Tournaments

The task of learning to rank is similar to the Minimum Feedback Arc Set in Tournaments (MFAST) problem in combinatorial optimization. In MFAST, one is given a set of n elements $V = \{v_1, \ldots, v_n\}$, and a matrix $W \in \mathbb{R}^{n*n}$ satisfying $\forall i, j \in \{1, 2, \ldots, n\}$, $b \leq W_{ij} + W_{ji} \leq 1$ for some constant $b \in [0, 1]$. The matrix W can be seen as the pairwise comparison results . The goal for MFAST is to find a permutation π minimizing $\sum_{u,v:\pi(u,v)=1} W_{u,v}$.

For our purpose, the most important difference between MFAST and learning to rank is that in MFAST, the goal is to design a computationally efficient algorithm, and the comparison matrix W is given as input at no cost; whereas in learning to rank, computational complexity is not the primary concern. In learning to rank, one can only get a sequence of samples of comparison results, and wants to minimize the loss with as few comparisons as possible.

MFAST has been proved to be NP-hard by [Alo06], and [KMS07] designs a polynomial-time approximation scheme (PTAS) for it. Although the algorithm in [KMS07] is not query-efficient, [Ail12] proposes an active ranking algorithm based on it.

Algorithm	Error rate	Sample Complexity
Passive ranking, ERM [Ail12]	$\mathrm{er}_{\mathcal{D}}(\pi) \leq \nu + \epsilon$	$\epsilon^{-2} n \log n$
Active ranking [Ail12]	$\mathrm{er}_{\mathcal{D}}(\pi) \leq (1+\epsilon)\nu$	$\epsilon^{-6} n \log^5 n$ in expectation
Active ranking using SRRA [ABE14]	$\mathrm{er}_{\mathcal{D}}(\pi) \leq (1+\epsilon)\nu$	$\epsilon^{-3} n \log^5 n$

Table 1: Results for ranking in the general setting

3.2 Learning to rank and sorting

Every comparison-based sorting algorithm is also an algorithm for active ranking from pairwise comparisons. There are many works discussing sorting algorithms when comparisons are noisy [FPRU90, KK07, BM08]. They assume that there is an underlying true total order, and the pairwise comparisons observed are certain noisy versions of their true relative orders (Note that in Section 2 we do not assume the existence of an underlying true total order). Many algorithms in noisy sorting also work for MFAST[ACN08] and active ranking (we will see an active ranking from sorting in Subsection 5.1).

Another classical model for noisy ranking is the Mallow's model where the algorithm is given a set of i.i.d. samples of permutations instead of pairwise comparisons, and the probability of observing a permutation σ is proportional to $e^{-\beta d_K(\pi,\sigma)}$ where β is some constant and π is the underlying true permutation. This is similar to the setting of list-wise learning to rank in information retrieval. This model is beyond the scope of this report because we are focusing on ranking from pairwise comparisons.

3.3 Other related ranking problems

One interesting theoretical problem is bipartite ranking which involves both ranking and classification. The algorithm is expected to output a total order, but unlike our problem, in bipartite ranking each object is associated with a positive/negative label, and the loss is measured by how many the negative objects are put ahead of the positive samples in the output. Many works study the relation between bipartite ranking and binary classification [Aga05, AM07, BBB+08].

For applications, there has been extensive research on various learning to rank models in information retrieval. The models include the pointwise approach (where the algorithm aims at modeling the score function of each object), the pairwise approach (where the algorithm aims at modeling the pairwise preference), and the listwise approach (where the input can be some random copies of partial/total order of some objects, and the goal is to output a sorted list) [Liu09]. And instead of Kendall-τ distance or Spearman Footrule distance, many works employ measures (for example, Mean Reciprocal Rank, Mean Average Precision, and Discounted Cumulative Gain) that give a higher weight to the objects that rank higher to accommodate the phenomena that users often only care the first few items returned by the search engine.

In addition, in collaborate filtering, many works are on personalized ranking where instead of giving a global order of objects, it is expected to give different rankings for different users based on some user-specific features ([CXL+06, RFGST09]).

Another interesting problem is *label ranking* in multiclass classification problem, and its difference with the classical multiclass classification setting is that here the for each object, the algorithm will receive a (partial) order of labels (for example, a partial order of labels can be "this image is more likely to be a *cat* than a *dog*") [DSM03, HFCB08].

4 Learning to rank in the general setting

In this section, we survey algorithms for learning to rank in a general agnostic setting that does not assume any structures underlying the n objects, and does not assume any ground truth permutation. The results are summarized in Table 1.

4.1 An additive VC bound for passive ranking

Note that each permutation induces a binary classifier on pairs: for a permutation π, for any pair $(u, v) \in [n] \times [n]$, recall that $\pi(u, v) = 1$ if v_u is before v_v in π, and 0 otherwise. [Ail12] proves that the VC dimension of the set of binary classifiers induced by permutations of n objects is $n - 1$. Consequently, we have the following VC bound.

Theorem 1. *[Ail12] If E is a set containing m i.i.d. samples from $\mathcal{D}$, then with probability at least $1 - \delta$, for any permutation π,*

$$\left| \mathbb{E}_{\langle u,v\rangle \sim \mathcal{D}} \mathbb{I}[\pi(u, v) \neq 1] - \frac{1}{m} \sum_{\langle u,v\rangle \in E} \mathbb{I}[\pi(u, v) \neq 1] \right| = O\left(\sqrt{\frac{n \log m + \log \frac{1}{\delta}}{m}} \right)$$

This implies that in order to achieve an *additive* error of ϵ, i.e., $\mathrm{er}_{\mathcal{D}}(\pi) \leq \nu + \epsilon$, an passive learner can simply do Empirical Risk Minimization (ERM) (assume the learner has unlimited computational power) which requires $\epsilon^{-2} n \log n$ comparisons asymptotically. But in many cases, it is required to achieve a *multiplicative* error of ϵ, i.e., $\mathrm{er}_{\mathcal{D}}(\pi) \leq (1 + \epsilon)\nu$. A multiplicative error is more meaningful when ν is very small (for instance, in the realizable case where $\nu = 0$). In the following subsections, we will see that an active learner could achieve a multiplicative error of ϵ with $O\left(n \mathrm{poly}(\epsilon^{-1}, \log \frac{1}{\delta}, \log n)\right)$ labels.

4.2 Active ranking based on the PTAS for MFAST

[Ail12] proposes an active ranking algorithm consisting of two steps: first decompose V, the set of n objects to rank, into groups, and then find a good permutation for each group.

Formally, in the decomposition step, the algorithm tries to find an ϵ-good decomposition, which is defined as follows.

Definition 2. We say $V_1, \ldots, V_k \subseteq V$ is a decomposition if $V_1, \ldots, V_k$ are pairwise disjoint, and $\cup_{i=1}^{k} V_k = V$. Let $\Pi(V)$ be the set of permutations on V. For a decomposition, let $\Pi(V_1, \ldots, V_k) = \{\pi \in \Pi(V) : \pi(u, v) = 1 \text{ for any } 1 \leq i < j \leq k, u \in V_i, v \in V_j\}$. We say $U \subseteq V$ is small if $|U| \leq \log n / \log \log n$, otherwise we say U is big.

A decomposition is *ϵ-good* if both of the following hold:

(1) Local chaos:

$$\min_{\pi \in \Pi(V)} \sum_{i: V_i \text{ is big}} \binom{|V_i|}{2} \mathbb{E}_{\langle u,v\rangle \sim \mathcal{D}} \left[\mathbb{I}[\pi(u, v) \neq 1] | u, v \in V_i \right] \geq \epsilon^2 \sum_{i: V_i \text{ is big}} \binom{|V_i|}{2}$$

,

(2) Approximate optimality

$$\min_{\pi \in \Pi(V_1, \ldots, V_k)} \mathrm{er}_{\mathcal{D}}(\pi) \leq (1 + \epsilon)\nu$$

In other words, a decomposition is ϵ-good if every big group is hard to rank, and if one could rank each group well, it will have low error overall.

If one can construct a ϵ-good decomposition $V_1, \ldots, V_k$, then by standard VC bound and the definition of ϵ-good decomposition, the following lemma holds:

Lemma 3. *[Ail12]Let $B = \{i \in [k] : V_i \text{ is big}\}$, E be random samples of size $O\left(\epsilon^{-6} n \log n\right)$ drawn uniformly at random from $\cup_{i \in B} \binom{V_i}{2}$, $E_i = E \cap \binom{V_i}{2}$ for $i \in B$. For any $\pi \in \Pi(V_1, \ldots, V_k)$, define*

$$\begin{aligned} \hat{C}(\pi) \quad = \quad & \left(|E|^{-1} \sum_{i \in B} \binom{|V_i|}{2} \right) \sum_{i \in B} \sum_{\langle u,v\rangle \in E_i} \mathbb{I}[\pi(u, v) \neq 1 \textit{ and } u, v \in V_i] \\ & + \sum_{i \in B^c} \binom{|V_i|}{2} \mathbb{E}_{\langle u,v\rangle \sim \mathcal{D}} \left[\mathbb{I}[\pi(u, v) \neq 1] | u, v \in V_i \right] \end{aligned}$$

if $\pi^* \in \Pi(V_1, \ldots, V_k)$ is a minimizer of $\hat{C}(\pi)$, then

$$er_{\mathcal{D}}(\pi^*) \leq (1+\epsilon)\nu$$

Lemma 3 implies that a minimizer of $\hat{C}(\pi)$ is also a good ranking. To minimize $\hat{C}(\pi)$, for the first term, it is just a combinatorial optimization problem. Although this is known as Minimum Feedback Arc Set (MFAS) problem which is even computational harder than MFAST [Kar72], since we only care about query complexity, it can be solved without any extra comparisons. For the second term, since $\sum_{i\in B^c}\binom{|V_i|}{2}$ is as small as $O(n\log n/\log\log n)$, one can simply make comparisons of all pairs among them.

What remains is to find a ϵ-good decomposition with few comparisons. The procedure for finding a good decomposition is similar to the PTAS of [KMS07], which first finds a ranking with constant multiplicative error, then repetitively tries to make local improvements. To make local improvements, one needs to evaluate the change of error rate after modifying the ranking. In [KMS07], one could directly recalculate the loss function, but here, we need to find a query-efficient method to get a good estimation of the loss. The algorithm proposed by [Ail12] is presented as Algorithm 1.

The procedure Decompose in Algorithm 1 first finds some ranking with constant multiplicative error, and then calls procedure SampleAndDecompose to recursively construct the decompositions. Before recursion, SampleAndDecompose will first check between line 6 and line 12 if current set V is a small set or already satisfies local chaos, and then try to make local improvement by calling ApproxLocalImprove. Line 23 to line 27 draws samples with different granularity which is used to guarantee a good estimation accuracy for evaluating the effect of a local move in line 28. To keep this good granularity after each move, line 30 will adjust the sample set and make extra samples when necessary.

It can be shown that Algorithm 1 will return an ϵ-good decomposition with probability at least $1-n^{-2}$ and querying $O(\epsilon^{-6}n\log^5 n)$ comparisons in expectation. The running time for this step is $O(n\text{poly}(\log n, \epsilon^{-1}))$. Combining with Lemma 3, we have the following guarantee overall:

Theorem 4. *Let π be the result by first finding an ϵ-good decomposition with Algorithm 1 and then optimizing $\hat{C}(\pi)$. Then π satisfies $er_{\mathcal{D}}(\pi) \leq (1+O(\epsilon))\,\nu$ with probability at least $1-n^{-2}$, and the expect number of comparisons made is $O\big(\epsilon^{-6}n\log^5 n\big)$.*

4.3 Active ranking using Smooth Relative Regret Approximations

[ABE14] proposes a general scheme for active learning called Smooth Relative Regret Approximations (SRRA) and applies it to active ranking. We will first introduce SRRA in a general active learning setting, and then demonstrate how it can be used for active ranking.

4.3.1 Some additional definitions for general active learning

Denote by $\mathcal{X}$ the sample space, and by $\mathcal{Y}=\{0,1\}$ the label space. Denote by $\mathcal{D}$ the underlying distribution over $\mathcal{X}\times\mathcal{Y}$, by $\mathcal{D}|_x$ the conditional distribution of label given a sample x, $\mathcal{D}_x$ the marginal distribution over $\mathcal{X}$. The hypothesis space $\mathcal{H}$ is a subset of $\mathcal{X}^{\mathcal{Y}}$, the set of functions mapping from $\mathcal{X}$ to $\mathcal{Y}$.

Define $\text{er}_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{I}[h(x)\neq y]\right]$, $\nu = \inf_{h\in\mathcal{H}} \text{er}_{\mathcal{D}}(h)$, $d(h_1,h_2) = \Pr_{x\sim D_x}\left(h_1(x)\neq h_2(x)\right)$.

4.3.2 The SRRA algorithm

Define $\text{reg}_h(h') = \text{er}_{\mathcal{D}}(h') - \text{er}_{\mathcal{D}}(h)$ to be *the relative regret function with respect to h.* Let $f:\mathcal{H}\to\mathbb{R}$ be any function, $0\leq\epsilon\leq\frac{1}{5}$, $0\leq\mu\leq 1$, we say f is an ϵ-smooth relative regret approximation (ϵ-SRRA) with respect to a pivot hypothesis h if $\forall h'\in\mathcal{H}$, $|f(h')-\text{reg}_h(h')|\leq \epsilon d(h,h')$. An SRRA is often an empirical estimation of the relative regret function. Note that an SRRA should have a low multiplicative error compared with the true distance instead of additive error, which requires SRRA to be even more precise when h' gets closer to the pivot hypothesis.

If one can construct an ϵ-SRRA, then the following theorem states that Algorithm 2 will learn a classifier with good generalization error.

Algorithm 1 Finding an ϵ-good Decomposition

```
1: procedure Decompose($V$, $\epsilon$)
2:     $\pi \leftarrow$ an expected $O(1)$-approximate solution to MFAST using $O(|V| \log |V|)$ queries on expectation (e.g. the algorithm in [ACN08])
3:     return SampleAndDecompose($V$, $|V|$, $\epsilon$, $\pi$)
4: end procedure
5: procedure SampleAndDecompose($V$, $n$, $\epsilon$, $\pi$)
6:     if $|V| \leq \log n / \log\log n$ then
7:         return trivial partition $\{V\}$
8:     end if
9:     $E \leftarrow$ uniformly draw at random $O\left(\epsilon^{-4} \log n\right)$ samples from $\binom{V}{2}$ and for their comparisons
10:    if $|E|^{-1} \sum_{\langle u,v\rangle \in E} \mathbb{I}[\pi(u,v) \neq 1] \geq \Omega(\epsilon^2)$ then
11:        return trivial partition $\{V\}$
12:    end if
13:    $\pi' \leftarrow$ ApproxLocalImprove($V$, ,$n$, $\epsilon$, $\pi$)
14:    $k \leftarrow$ draw an integer uniformly at random from $[|V|/3, 2|V|/3]$
15:    decomposeL $\leftarrow$ SampleAndDecompose($\{\pi'(i) : i = 1, \ldots, k\}$, $n$, $\epsilon$, ${\pi'}_1^k$)
16:    decomposeR $\leftarrow$ SampleAndDecompose($\{\pi'(i) : i = k+1, \ldots, |V|\}$, $n$, $\epsilon$,${\pi'}_{k+1}^{|V|}$)
17:    return concatenation of decomposeL and decomposeR
18: end procedure
19: procedure ApproxLocalImprove($V$, $n$, $\epsilon$, $\pi$)
20:    if $|V| \leq \epsilon^{-3} \log^3 n$ then
21:        return $\pi$
22:    end if
23:    for $i = 1, \ldots, |V|$ do
24:        for $j = \log \frac{\epsilon |V|}{\log n}, \ldots, \log |V|$ do
25:            $E_{i,j} \leftarrow$ draw $\Theta(\epsilon^{-2} \log^2 n)$ integers from $[\max\{1, i-2^j\}, \min\{|V|, i+2^j\}]$ uniformly at random, and query for the comparisons between $v_i$ and each of them
26:        end for
27:    end for
28:    while $\exists i, j \in [|V|]$ s.t. (let $l = \log |i-j|$, $\tilde{\pi}$ = switch $i$-th element and $j$-th element in $\pi$)
           $$\sum_{\langle u,v\rangle \in E_{i,l}} \mathbb{I}[\tilde{\pi}(u,v) \neq 1] \leq \sum_{\langle u,v\rangle \in E_{i,l}} \mathbb{I}[\pi(u,v) \neq 1] - \frac{\epsilon |E_{i,l}|}{\log n}$$
       do
29:        $\pi \leftarrow \tilde{\pi}$
30:        refresh samples $E_{i,j}$ for all $i, j$ with respect to the new permutation $\pi$
31:    end while
32:    return $\pi$
33: end procedure
```

Algorithm 2 SRRA Algorithm

```
1: Input: $\epsilon$, the number of iterations $T$
2: $h_0 \leftarrow$ an initial hypothesis
3: for $k = 1, 2, \ldots, T$ do
4:     Derive an $\epsilon$-SRRA $f$ with respect to $h_{k-1}$
5:     $h_k \leftarrow \arg\min_{h \in \mathcal{H}} f(h)$
6: end for
7: Output: $h_T$
```

Lemma 5. *[ABE14] Let h be the output of Algorithm 2 with parameter ϵ, T. Then $er_{\mathcal{D}} = (1 + O(\epsilon))\,\nu + O(\epsilon^T) er_{\mathcal{D}}(h_0) + O(\epsilon\mu)$.*

4.3.3 Active ranking with SRRA Algorithm

Algorithm 2 is a generic scheme for active learning with theoretical guarantee on the generalization error. Labels are used to derive ϵ-SRRAs. In [ABE14], the authors propose a general algorithm to get ϵ-SRRAs. Their construction involves disagreement coefficient [Han07] which is a key parameter to characterize active learning problems [BBL09, BDL09, BHLZ10, BL13]. Intuitively, the harder to distinguish between two hypotheses, the more labels need to be queried so that the approximated regret f will be accurate enough up to a multiplicative error. This method is computationally inefficient in general, but the same intuition could be applied to derive an efficient sampling method to construct SRRAs for active ranking, which is presented as Algorithm 3.

Algorithm 3 SRRA Construction

```
1: Input: $\epsilon$, $n$, a pivot permutation $\pi$
2: Output: An $\epsilon$-SRRA with respect to $\pi$
3: $p \leftarrow O\left(\epsilon^{-3}\log^3 n\right)$
4: for $u = 1, 2, \ldots, n$ do
5:     for $i = 1, 2, \ldots, \log n$ do
6:         $I_{u,i} \leftarrow \left\{v : (2^i - 1)p < |\pi(u) - \pi(v)| < 2^{i+1}p\right\}$
7:         Draw $p$ samples uniformly at random, and get $v_{u,i,1}, \ldots, v_{u,i,p}$
8:         Ask the Oracle to compare $u$ with $v_{u,i,1}, \ldots, v_{u,i,p}$, and get $p$ labels $y_{u,i,1}, \ldots, y_{u,i,p}$
9:     end for
10: end for
11: Define $\hat{\text{er}}$ as
        $$\hat{\text{er}}(\sigma) = \sum_{u=1}^{n}\sum_{i=0}^{\log n}\sum_{t=1}^{p}\frac{|I_{u,i}|}{\binom{n}{2}p}\mathbb{I}\left[\sigma(u, v_{u,i,t}) \neq y_{u,i,t}\right]$$
12: Define $f$ as $f(\sigma) = \hat{\text{er}}(\sigma) - \hat{\text{er}}(\pi)$
13: Return $f$
```

The intuition for Algorithm 3 is the same with the general sampling method mentioned above. For a given permutation π, for any element u, it will make more comparisons between u and nearby elements of u in π such that f has a low multiplicative approximation error. It is also interesting to point out that this sampling scheme is similar to the one in ApproxLocalImprove in Algorithm 1. It can be shown that Algorithm 3 outputs an ϵ-SRRA, so we have the following error bound for active ranking with SRRA algorithm.

Theorem 6. *[ABE14] Let π be the output of Algorithm 2 with $T = \log n$ and use Algorithm 3 to construct ϵ-SRRAs. Then at most $O\big(\epsilon^{-3} n \log^5 n\big)$ comparisons are made, and with probability at least $1 - n^{-2}$, π satisfies $er_{\mathcal{D}}(\pi) \leq (1 + O(\epsilon))\,\nu$.*

Remark 7. Although Algorithm 3 is computational efficient, the active ranking algorithm is not efficient overall without additional assumptions since one still needs to optimize over the SRRA in Algorithm 2.

4.4 Discussion

In order to achieve a good multiplicative error $\text{er}_{\mathcal{D}}(\pi) \leq (1 + O(\epsilon))\,\nu$, the sample complexity for Algorithm 3 is $\text{O}\big(\epsilon^{-3} n \log^5 n\big)$, which is better than the $\text{O}\big(\epsilon^{-6} n \log^5 n\big)$ for Algorithm 1. For the passive ranking, to achieve an additive error $\text{er}_{\mathcal{D}}(\pi) \leq \nu + \epsilon$, it requires $O\left(\epsilon^{-2} n \log n\right)$ samples, which seems to be smaller, but note that an additive error guarantee is much weaker than that of a multiplicative error. To see that, in order to achieve $\text{er}_{\mathcal{D}}(\pi) = \Theta(\nu)$, the passive learner requires $O\left(\nu^{-2} n \log n\right)$ samples, while the active learner

Assumption	Algorithm	Error rate	Sample Complexity
None	Passive, ERM [Ail12]	$\text{er}_{\mathcal{D}}(\pi) \le \nu + \epsilon$	$\epsilon^{-2} n \log n$
None	Active ranking using SRRA [ABE14]	$\text{er}_{\mathcal{D}}(\pi) \le (1+\epsilon)\nu$	$\epsilon^{-3} n \log^5 n$
Bounded noise	Passive, Borda count [WJJ13]	$\Pr(\pi \ne \pi^*) \le \frac{1}{\sqrt{n}}$	n^2
Bounded noise	Active, noisy sorting [BM08]	$\Pr(\pi \ne \pi^*) \le \frac{1}{n}$	$n \log^2 n$
BTL with additional assumptions	Active, quick sort [MG15b]	$\Pr(\pi \ne \pi^*) \le \frac{1}{n}$	$n \log n$
Embedding, no noise	Passive [JN11a]	Exactly recover	$\Omega\left(n^2\right)$ on average
Embedding, no noise	Active, disagreement-based [JN11a]	Exactly recover	$d \log n$ on average

Table 2: Results for ranking with additional structures. For comparison reasons, in the first two rows we also list results under the general setting in Section 4. The loss function for algorithms in bounded noise and the BTL model is different with the general setting. The relationship between these two loss functions is discussed in Subsection 5.1.4. We do not include the results in [RA14] for BTL models because its results are just consistency results.

requires $O\left(n \log^5 n\right)$ samples which is independent of ν. In the extreme case where $\nu = 0$, the active learner still requires $O\left(n \log^5 n\right)$ samples, but for passive ranking $\Omega(n^2)$ samples are actually necessary, because to achieve zero error the algorithm must know every consecutive pairs in the optimal permutation, which requires $\Omega(n^2)$ draws if the samples are drawn i.i.d.. In this sense, active ranking is better than passive ranking. It would be interesting to see if one could improve the sample complexity for active ranking.

5 Learning to rank with additional structures

In this section, we will make some additional assumptions on structures underlying the n objects to rank. First, we will consider ranking with bounded noise, which assumes an optimal order and that the comparison results are flipped with some probability bounded away from $\frac{1}{2}$. Next, we will see another model that also assumes an optimal order. The comparison results in this model can be noisier but follow a probabilistic model. Last, we will discuss the setting that assumes an embedding in Euclidean space. The results are summarized in Table 2.

5.1 Ranking with bounded noise

5.1.1 The model

In this subsection, we assume that there is a true underlying order π^*. The noise is bounded in the sense that there is a constant $\gamma \in (0, \frac{1}{2})$ such that for any two objects v_i, v_j, $\Pr(v_i \prec v_j | \pi^*(i,j) = 1) \ge \frac{1}{2} + \gamma$.

Since we assume the existence of a ground truth π^*, we will use $\Pr(\sigma \ne \pi^*) = \Pr_{(u,v)\sim\mathcal{D}_{\mathcal{X}}}\left(\sigma(u,v) \ne \pi^*(u,v)\right)$ to measure prediction error in this subsection. Recall that by triangle inequality for any σ, $\Pr(\sigma \ne \pi^*) \le \text{er}_{\mathcal{D}}(\sigma) + \nu$, and that under the bounded noise condition, $\nu \le \frac{1}{2} - \gamma$.

5.1.2 Balanced Rank Estimation algorithm

[WJJ13] gives an analysis of a very simple counting algorithm called Balanced Rank Estimation which is also known as Borda Count algorithm, shown as Algorithm 4.

Theorem 8. *[WJJ13] Let the σ be the output of Algorithm 4, and assume $\{\langle u_i, v_i\rangle\}_{i=1}^{M}$ are i.i.d drawn from $\mathcal{D}$* without replacement. *If $m \ge \Omega\left(\frac{n}{\eta^2\gamma^2}\right)$, and n is large enough, then $\Pr(\sigma \ne \pi^*) \le \eta$. If further $m \ge \Omega(n \log n)$, then with probability at least $1 - n^{1 - c a_n \gamma^2 \eta^2}$, $\max_j |\sigma(j) - \pi^*(j)| \le \eta n$, where $\{a_n\}$ is a sequence of numbers that converge to 1.*

Algorithm 4 Balanced Rank Estimation

```
1: Input: $n$, $M$ training samples $\langle u_i, v_i\rangle_{i=1}^M$
2: Output: the ordering of the objects
3: for $k = 1, 2, \dots, n$ do
4:     $c_k \leftarrow \sum_{i=1}^M \mathbb{I}[u_i = k]$
5: end for
6: Let $\sigma$ be the permutation according to the increacing order of $c_k$
7: Return $\sigma$
```

Note that this theorem requires $\{\langle u_i, v_i\rangle\}_{i=1}^M$ are drawn *without replacement,* so $\eta \geq \Omega(\frac{1}{\sqrt{n}})$.

[RA14] also analyses this algorithm that assumes $\{\langle u_i, v_i\rangle\}_{i=1}^M$ are drawn with replacement. But unlike the low error rate sample complexity bound in [WJJ13] , the analysis in [RA14] is on the consistency of the algorithm that shows, roughly speaking, if $m \gtrsim n^4 \log n$ and further $\mathcal{D}|_x$ satisfies some low noise condition, then with high probability $\Pr(\sigma \neq \pi^*) = 0$.

5.1.3 An active ranking algorithm from sorting

[BM08] discusses noisy sorting which can also be seen as active ranking from pairwise comparisons.

If repetitions are allowed in querying, then it is easy to see that by applying any standard sorting algorithm running in $O(n \log n)$ time and repeating each comparison for $O(\log n)$ times, then with high probability one can recover π^* using $O(n \log^2 n)$ queries.

It is more sophisticated and interesting to consider the case where repetitions are not allowed. [BM08] designs a randomized algorithm shown as Algorithm 5.

Algorithm 5 Noisy Sorting

```
1: Input: $V$, $\gamma$
2: Output: the ordering of the objects
3: $S_0 \leftarrow \emptyset$, $\sigma_0 \leftarrow$ an empty permutation
4: for $k = 1, 2, \dots, n$ do
5:     $a_k \leftarrow$ choose an object from $V \backslash S_{k-1}$ uniformly at random
6:     $S_k \leftarrow S_{k-1} \cup \{a_k\}$
7:     Split $\sigma_{k-1}$ into blocks $B_1, \dots, B_{l(k)}$ such that each block is of size $\Theta(\log n)$
8:     Use a noisy binary search procedure to find a $t$ such that any object in $B_1, \dots, B_{t-1}$ is smaller than
       $a_k$ and any object in $B_{t+1}, \dots, B_{l(k)}$ is larger than $a_k$ with high probability
9:     $\hat{\sigma_k} \leftarrow B_1, \dots, B_{t-1}, B_t, \{a_k\}, B_{t+1}, \dots, B_{l(k)}$
10:    Use a dynamic programming to find a permutation $\sigma_k$ based on $\hat{\sigma_k}$ with low error rate
11: end for
12: Return $\sigma_n$
```

Algorithm 5 uses a framework of insertion sort. For each object to insert, it first roughly estimates the position of this new object by a variant of noisy binary search in [KK07], and then uses a dynamic programming at line 10 which is guaranteed to find a permutation that maximizes $s(\sigma) = \sum_{i,j:\sigma(i,j)=0} \mathbb{I}(v_i \prec v_j)$ given the assumption that $\max_i |\hat{\sigma_k}(i) - \pi^*(i)| = O(\log n)$.

[BM08] shows that with high probability, the permutation that maximizes $s(\sigma)$ is close to π^*, the binary search process at line 10 will guarantee $\max_i |\hat{\sigma_k}(i) - \pi^*(i)| = O(\log n)$, and the dynamic programming at line 10 will only require $O(\log n)$ queries each time. Therefore, Algorithms 5 has following guarantee:

Theorem 9. *[BM08] Let σ be the output of Algorithms 5 and β be a positive real number. Then with probability at lest $1 - n^{-\beta}$, Algorithms 5 makes at most $O(C(\beta, \gamma) n \log n)$ comparisons, and*

$$\max_i |\sigma(i) - \pi^*(i)| \leq C_1(\beta) \log n \ , \ \Pr(\sigma \neq \pi^*) \leq \frac{C_2(\beta)}{n}$$

, where $C(\beta,\gamma), C_1(\beta), C_2(\beta)$ is independent with n.

Remark 10. One drawback for this algorithm is that it cannot achieve a lower error by increasing sample size.

5.1.4 Discussion

Comparing the guarantee for passive ranking and active ranking, Theorem 8 does not give sample complexity to achieve $\Pr(\sigma \neq \pi^*) \leq O\left(\frac{1}{n}\right)$ or $\max_i |\hat{\sigma_k}(i) - \pi^*(i)| = O(\log n)$, while Theorem 9 shows $O(n \log n)$ queries are enough for active ranking to achieve $\Pr(\sigma \neq \pi^*) = O\left(\frac{1}{n}\right)$.

It is not clear how the algorithms for the general setting in Section 4 perform under this bounded noise setting. It is tentative to use the triangle inequality $\Pr(\sigma \neq \pi^*) \leq \mathrm{er}_{\mathcal{D}}(\sigma) + \nu$ to relate the $\mathrm{er}_{\mathcal{D}}(\sigma)$ with $\Pr(\sigma \neq \pi^*)$, but to achieve $\Pr(\sigma \neq \pi^*) = O(\frac{1}{n})$, one has to set $\nu = O\left(\frac{1}{n}\right)$ which contradicts with the fact that $\nu = \frac{1}{2} - \gamma$ where γ is a constant.

5.2 Ranking with Bradley-Terry-Luce model

Bradley-Terry-Luce (BTL) model is a popular probabilistic model that models how likely one object gets ranked higher than another. It assigns a score to each object, and the higher the score is, the more likely the object ranks higher. When two objects have similar scores, it will be very hard to compare them.

More formally, each object v_i $(i = 1, \ldots, n)$ is associated with a score $w_i \in (0, 1]$ satisfying $\sum_{i=1}^n w_i = 1$. For any two objects v_i, v_j, $\Pr(v_i \prec v_j) = \frac{w_j}{w_i + w_j}$.

To state the results, we introduce following notations.

Notation 11. Define π^* to be the permutation that ranks the objects in increasing order of w_i. Define $\mu_m = \min_{u,v \in V} \Pr_{(u,v)\sim\mathcal{D}_\mathcal{X}}((u,v))$ to be the minimum of the probability of observing a certain pair. Define $P \in \mathbb{R}^{n\times n}$ to be the preference matrix such that $P_{ij} = \Pr(v_i \prec v_j)$. Define $\hat{P} \in \mathbb{R}^{n\times n}$ to be the empirical comparison matrix such that

$$\hat{P}_{ij} = \begin{cases} \frac{\sum_{\langle u,v\rangle \in E} \mathbb{I}[\langle u,v\rangle = \langle i,j\rangle]}{\sum_{(u,v)\in E} \mathbb{I}[(u,v) = (i,j)]} & \sum_{(u,v)\in E} \mathbb{I}\left[(u,v) = (i,j)\right] > 0 \\ 0 & \text{otherwise} \end{cases}$$

Again, we will use $\Pr_{(u,v)\sim\mathcal{D}_\mathcal{X}}(\sigma(u,v) \neq \pi^*(u,v))$ to measure prediction error in this subsection due to the existence of a ground truth π^*.

5.2.1 Maximum likelihood algorithm

Maximum likelihood algorithm for BTL model is straightforward. It consists of two steps. First, it makes a maximum likelihood estimation of $w_1, \ldots, w_n$:

$$\hat{\boldsymbol{w}} = \arg\max_{w} \prod_{\langle u,v\rangle \in E} \Pr(u \prec v | \boldsymbol{w})$$

Then it ranks the objects in increasing order of $\hat{w}_i$.

[RA14] provides a consistency result of this algorithm (in fact, it shows that Theorem 12 holds for a more general model called Low-Noise Condition):

Theorem 12. *[RA14] Assume $\mu_m > 0$, $\exists i, j$ s.t. $\Pr(v_i \prec v_j) \neq \frac{1}{2}$, and let σ be the order given by the above maximum likelihood algorithm. If the sample size $|E| \geq O\left(\frac{1}{\mu_m^2} \log \frac{n}{\delta}\right)$, then with probability at least $1 - \delta$, $\Pr_{(u,v)\sim\mathcal{D}_\mathcal{X}}(\sigma(u,v) \neq \pi^*(u,v)) = 0$.*

5.2.2 Rank Centrality algorithm

Another popular passive ranking algorithm is the Rank Centrality algorithm [NOS12, RA14] based on a random walk interpretation of BTL model.

We define the transition matrix Q for the Markov chain induced by a preference matrix P as

$$Q_{i,j} = \begin{cases} P_{ij}/n & \text{if } i \neq j \\ 1 - \sum_{k \neq i} P_{kj}/n & \text{if } i = j \end{cases}$$

. Recall that Q is time-reversible if and only if Q is irreducible and aperiodic, and its stationary distribution π satisfies $\pi_i Q_{ij} = \pi_j Q_{ji}$. [RA14] shows that the Markov chain induced by a preference matrix P is time-reversible if and only if P corresponds to some BTL model.

The intuition for Rank Centrality algorithm is as follows. Imagine a random walk on this Markov chain. One is more likely to walk from v_i to v_j if v_j ranks higher than v_i. Thus, the "stronger" an object is, the random walk will visit the object more frequently. Note that if the induced Markov chain of P satisfies the time-reversible condition, then for the stationary distribution π, $\forall i, j$, $w_j \geq w_i \iff P_{ij} > P_{ji} \implies \pi_j \geq \pi_i$. Thus, ranking the objects in the increasing order of stationary probabilities will yield the optimal order of BTL model. In the learning task, one has no access to the preference matrix, and it can only use the empirical preference matrix.

The rank centrality algorithm is demonstrated as Algorithm 6.

Algorithm 6 Rank Centrality

```
1: Input: the empirical comparison matrix $\hat{P}$, $n$
2: Output: the ordering of the objects
3: Construct the transition matrix $\hat{Q}$ for the Markov chain induced by $\hat{P}$
4: Calculate the stationary distribution $\hat{\pi}$ of $\hat{Q}$
5: Let $\sigma$ be the permutation according to the increacing order of $\hat{\pi}$
6: Return $\sigma$
```

[RA14] provides a consistency result of this algorithm:

Theorem 13. *[RA14] Assume $\mu_m > 0$, and let σ be the order given by Algorithm 6. If the sample size $|E| \geq O\left(\frac{n}{\mu_m^2} \log \frac{n}{\delta}\right)$, then with probability at least $1 - \delta$, $\Pr_{(u,v)\sim\mathcal{D}_\mathcal{X}}(\sigma(u,v) \neq \pi^*(u,v)) = 0$.*

Remark 14. Note that $\frac{1}{\mu_m^2}$ is at least n^4, the bounds in Theorem 13 and Theorem 12 require a huge amount of samples. But these two theorems focus on the consistency of the algorithms: how many samples it needs to obtain the optimal ranking. It would be interesting to see how many samples it needs to obtain a ranking that has low error rate compared with the optimal ranking.

Remark 15. [NOS12] gives a bound that requires less samples but with a stronger assumption. Their sample generating process as follows: first select some pairs uniformly at random, then compare each of these pairs for a fixed number of times. They show that under this generative assumption, if $|E| \geq O\left(n \log^3 n\right)$, then $\|\hat{\pi} - \pi\|_2 = o((\log n)^{-1/2})$. Note that this only guarantees the estimation accuracy of the stationary distribution. They demonstrate experimental results to show that a weighted version of prediction error goes down at the same speed of a maximum likelihood estimator, but they do not provide any theoretical guarantee on the error rate.

5.2.3 A quicksort-style active ranking algorithm

There is only a few works on active ranking for the BTL model. The only work known to the author that has theoretical guarantee is [MG15b].

[MG15b] makes an additional assumption about the BTL model: it assumes that the model parameters $w_1, \ldots, w_n$ are generated according to some Poisson process, and $\mathcal{D}_\mathcal{X}$ is uniform. The generating process is as follows: first, draw n random variables $\xi_1, \ldots, \xi_n$ i.i.d. from the exponential distribution with parameter λ, then let $w_i = \exp(\sum_{j=1}^{i} \xi_i)/C$ where C is a normalization term to make $\sum_i w_i = 1$.

They show that under this strong assumption, the classical randomized quicksort algorithm will guarantee that with probability at least $1-\frac{1}{n}$, it will output a permutation σ such that $\Pr_{(u,v)\sim\mathcal{D}_\mathcal{X}}(\sigma(u,v) \neq \pi^*(u,v)) \leq \frac{\lambda^3}{n}$.

5.2.4 Extensions

There are many generalizations of the BTL model. For example, [RA14] discusses the Low-Noise model. Its assumption is that $P_{ij} > P_{ji} \implies \sum_k P_{kj} \geq \sum_k P_{ki}$ (in other words, imagine n teams compete in a tournament, the stronger team should win more games than others). And they show that the BTL model follows this assumption, and that the maximum likelihood algorithm for BTL model actually works under this general assumption as well. They also consider a even more general model called generalized low-noise condition, and propose a SVM style algorithm that is consistent under this assumption.

Another popular extension is the Plackett-Luce (PL) model, where instead of observing the comparison of a pair at each time, the learner will get a permutation of a subset of the objects and the permutation is generated according to a probabilistic model similar to the BTL model [HOX14, MG15a].

5.2.5 Discussion

As far as we know, the maximum likelihood algorithm only has a consistency result, and the Rank Centrality algorithm only has a consistency result in general and a parameter estimation error result with some additional assumptions. It would be interesting to see if one could derive a sample complexity bound to guarantee a low prediction error ($\Pr_{(u,v)\sim\mathcal{D}_\mathcal{X}}(\sigma(u,v) \neq \pi^*(u,v))$ or $\mathrm{er}_D(\sigma)$) for passive ranking algorithms.

Besides, active ranking has not been intensively studied in this BTL model. The only algorithm ([MG15b]) known to the author has a strong assumption on the model parameters. It would be also interesting to explore active ranking methods for BTL models.

5.3 Ranking with underlying embedding

5.3.1 The model

Jamieson and Nowak assume that the n elements to be sorted are embedded in a d dimensional Euclidean space ($d < n$) and the comparisons are induced according to the distances to a fixed reference point [JN11a, JN11b]. Furthermore, they assume that the responses of the Oracle is noiseless, so the goal is to exactly recover the true permutation. In this noiseless setting, by simply applying sorting algorithms one could exactly recover the true permutation with $O(n \log n)$ time. Here, they want to find an algorithm which exploits the low dimensional embedding and requires a small number of comparisons that grows sublinear in n.

More formally, they assume $v_i \in \mathbb{R}^d$ for any $i = 1, \ldots, n$ (known to the learner), and there is a reference point $\theta \in \mathbb{R}^d$ (unknown to the learner) which induces a permutation σ such that $\sigma(v_i, v_j) = 1$ if and only if $\|\theta - v_i\| \leq \|\theta - v_j\|$. They also assume the ground truth permutation σ is generated uniformly at random from all possible permutations that admit a d dimensional embedding. Let $M_n(\sigma)$ be the number of comparisons required by an algorithm to exactly identify σ. They consider the *average query complexity* which is defined as $\mathbb{E}_\sigma M_n(\sigma)$.

5.3.2 Main results

They first prove that there is an embedding of n objects such that any learning algorithm must make $\Omega(n)$ comparisons. This justifies the use of average-case analysis instead of worst-case analysis as we did in previous sections.

By counting the total number of permutations that admit a d dimensional embedding, they derive the following lower bound.

Theorem 16. *[JN11a] For any permutation that admits a d dimensional embedding, any algorithm will require at least $\Theta(d \log_2 n)$ comparisons to exactly identify it.*

Also by counting the number of permitted permutations, they prove the following theorem to show the inefficiency of passive ranking.

Theorem 17. *[JN11a] If $o(n^2)$ pairwise comparisons are made uniformly at random without replacement from all possible $\binom{n}{2}$ pairs, then with probability at least $\frac{1}{2}$, there are at least two permutations that admits some d dimensional embedding and consistent with the comparisons made.*

They design a disagreement-based active ranking algorithm shown as Algorithm 7. In brief, the algorithm will maintain a set of possible positions of the reference point, and will make comparison of a pair only if there are two possible reference points in the set that induce different pairwise order of the pair. In other words, this algorithm will only make comparisons when the relative order of a pair cannot be determined from the comparisons already made. To check this, note that for each pair (v_i, v_j), v_i ranks higher than v_j if and only if $\|\theta - v_i\| \leq \|\theta - v_j\|$, which is equivalent to $(v_2 - v_1)^T \theta + v_1^T v_1 - v_2^T v_2 \leq 0$. Therefore, each comparison induces a half-space, and the comparisons made so far define a (maybe unbounded) polytope in $\mathbb{R}^d$ which is the set of possible positions of θ. The order of a pair is undetermined if and only if the half-space induced by this pair intersects the polytope. If the induced half-space and the polytope do not intersect, the order can be inferred by checking which side they polytope lies in the half-space.

Algorithm 7 Ranking with embedding

```
1: Input: $n$ objects $v_1, \ldots, v_n \in \mathbb{R}^d$
2: Randomly shuffle $v_1, \ldots, v_n$ and get $U_1, \ldots, U_n$
3: for $i = 1, 2, \ldots, n$ do
4:     for $j = 1, 2, \ldots, i-1$ do
5:         if the relative order between $U_i$ and $U_j$ cannot be determined then
6:             Request to compare $U_i$ and $U_j$
7:         else
8:             Infer the relative order between $U_i$ and $U_j$
9:         end if
10:    end for
11: end for
12: Return the ranking inferred from pairwise orders
```

Theorem 18. *[JN11a] Let M_n be the number of pairwise comparisons requested by Algorithm 7, and $P(n, i)$ be the probability that the relative order between U_i and U_j cannot been determined at line 5 of Algorithm 7. Then $P(n, i) = O(\frac{d}{i^2})$ and thus $\mathbb{E}[M_n] = O(d \log n)$.*

Therefore, active ranking requires much less comparisons than passive ranking (whose sample complexity lower bound is $\Omega(n^2)$).

When the Oracle's responses are noisy (i.e., the responses will flip between 0 and 1), if the flipping probability is bounded away from $\frac{1}{2}$ and each query can be repeated to obtain multiple independent samples, then one can simply repeat each query for multiple times and use the majority vote. This only adds a logarithm term to the query complexity. When the noise is consistent (i.e., repetitive querying is not allowed), [JN11a] designs an algorithm for this case, but it does not have any theoretical guarantee on the query complexity.

6 Algorithms in applications

Most algorithms mentioned above are not computationally efficient. On the application side, many efficient passive and active learning to rank methods have been proposed. Most of them lack theoretical guarantees, but report good empirical results.

There is a popular line of research starting from the seminal work of [Joa02] that uses SVM to learn to rank. In [Joa02], each object v_i is associated with a feature vector $\phi(v_i)$, and it assumes a linear model that there is a weight vector w such that $\forall i, j\ v_i \prec v_j \iff w^T\phi(v_i) \leq w^T\phi(v_j)$. The algorithm is to minimize the SVM style objective function:

$$\begin{aligned}&\min L(w,\xi) = \frac{1}{2}w^T w + C\sum \xi_{i,j}\\ &s.t. \forall i,j\ v_i \prec v_j : w^T\phi(v_j) - w^T\phi(v_j) \geq 1 - \xi_{i,j}\\ &\quad \forall i,j\ \xi_{i,j} \geq 0\end{aligned}$$

Following this, some works discuss active sampling. For instance [Yu05] will query the pair that is closest to the decision boundary, and [DC09] will query the pair that has the largest "expected" hinge loss.

Some algorithms assume a generative model. For instance, [RJ07] uses a Bayesian framework that maintains $\Pr(M|D)$ where M is the vector of "relevant" scores for each objects, and D is the comparison results. They assume $\Pr(D|M)$ follows the BTL model, and will actively make queries to update $\Pr(M|D)$ by choosing the pair that maximizes some expected loss with respect to current $\Pr(M|D)$ or maximizes the decrease of some expected overall loss.

Another class of methods is the score-based approach. For instance, [ZCSZ07] assumes that each object has a feature vector, and that there is a score function defined over the feature vector such that the larger the score is, the higher the object ranks. It then designs a regression framework to learn a ranking through learning the score function.

7 Conclusion

In this report, we give a survey on passive and active algorithms for learning to rank from pairwise comparisons. We focus on the algorithms with theoretical guarantees in the general agnostic setting, as well as ranking with some special structures including the bounded noise model, the BTL model, and the model that admits a low dimensional Euclidean space embedding. In most settings, active ranking requires less comparisons than passive ranking.

There are still many open problems in ranking from pairwise comparisons, including:

- In the general setting, active ranking algorithms are query efficient in the sense that it only requires about $n \log n$ queries to achieve a low multiplicative error, but are not computationally efficient. The PTAS designed by [KMS07] is computationally efficient but requires the access to the comparisons of all $\binom{n}{2}$ pairs. Can we design an algorithm that is both computationally efficient and query efficient?
- In the general setting, active ranking algorithms require at least $O\left(\epsilon^{-3} n \log^5 n\right)$ comparisons. Can we design an active ranking algorithm with better dependency on ϵ and $\log n$? Or are there any lower bounds with respect to ϵ and $\log n$?
- Most active ranking algorithms discussed in this report require the distribution $\mathcal{D}_{\mathcal{X}}$ to be uniform. Can we relax this assumption?
- For the algorithms in the bounded noise model and the BTL model, how many samples are required to achieve a low $\mathbb{E}_{\langle u,v\rangle\sim\mathcal{D}}\mathbb{I}[\sigma(u,v) \neq 1]$ instead of just $\Pr_{(u,v)\sim\mathcal{D}_{\mathcal{X}}}(\sigma(u,v) \neq \pi^*(u,v))$?
- For the BTL model, can we design an active ranking method with weaker assumptions than [MG15b]?

References

[ABE14] Nir Ailon, Ron Begleiter, and Esther Ezra. Active learning using smooth relative regret approximations with applications. *The Journal of Machine Learning Research*, 15(1):885–920, 2014.

[ACN08] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):23, 2008.

[Aga05] Shivani Agarwal. A study of the bipartite ranking problem in machine learning. 2005.

[Ail12] Nir Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13:137–164, 2012.

[Ail13] Nir Ailon. Learning and optimizing with preferences. In *Algorithmic Learning Theory*, pages 13–21. Springer, 2013.

[Alo06] Noga Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006.

[AM07] Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. *arXiv preprint arXiv:0710.2889*, 2007.

[BBB$^+$08] Maria-Florina Balcan, Nikhil Bansal, Alina Beygelzimer, Don Coppersmith, John Langford, and Gregory B Sorkin. Robust reductions from ranking to classification. *Machine learning*, 72(1-2):139–153, 2008.

[BBL09] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. *J. Comput. Syst. Sci.*, 75(1):78–89, 2009.

[BDL09] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *ICML*, 2009.

[BHLZ10] A. Beygelzimer, D. Hsu, J. Langford, and T. Zhang. Agnostic active learning without constraints. In *NIPS*, 2010.

[BL13] M.-F. Balcan and P. M. Long. Active and passive learning of linear separators under log-concave distributions. In *COLT*, 2013.

[BM08] Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 268–276. Society for Industrial and Applied Mathematics, 2008.

[CXL$^+$06] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM, 2006.

[DC09] Pinar Donmez and Jaime G Carbonell. Active sampling for rank learning via optimizing the area under the roc curve. In *Advances in Information Retrieval*, pages 78–89. Springer, 2009.

[DG77] Persi Diaconis and Ronald L Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268, 1977.

[DSM03] Ofer Dekel, Yoram Singer, and Christopher D Manning. Log-linear models for label ranking. In *Advances in neural information processing systems*, page None, 2003.

[FPRU90] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Computing with unreliable information. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 128–137. ACM, 1990.

[Han07] Steve Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th international conference on Machine learning*, pages 353–360. ACM, 2007.

[HFCB08] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916, 2008.

[HOX14] Bruce Hajek, Sewoong Oh, and Jiaming Xu. Minimax-optimal inference from partial rankings. In *Advances in Neural Information Processing Systems*, pages 1475–1483, 2014.

[JN11a] Kevin G Jamieson and Robert Nowak. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems*, pages 2240–2248, 2011.

[JN11b] Kevin G Jamieson and Robert D Nowak. Active ranking in practice: General ranking functions with sample complexity bounds. In *Annual Conference on Neural Information Processing Systems workshop*. Citeseer, 2011.

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[Kar72] Richard M Karp. *Reducibility among combinatorial problems.* Springer, 1972.

[KK07] Richard M Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 881–890. Society for Industrial and Applied Mathematics, 2007.

[KMS07] Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 95–103. ACM, 2007.

[Liu09] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[MG15a] Lucas Maystre and Matthias Grossglauser. Fast and accurate inference of plackett–luce models. In *Advances in Neural Information Processing Systems*, pages 172–180, 2015.

[MG15b] Lucas Maystre and Matthias Grossglauser. Robust active ranking from sparse noisy comparisons. *arXiv preprint arXiv:1502.05556*, 2015.

[NOS12] Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pair-wise comparisons. In *Advances in Neural Information Processing Systems*, pages 2474–2482, 2012.

[RA14] Arun Rajkumar and Shivani Agarwal. A statistical convergence perspective of algorithms for rank aggregation from pairwise data. In *Proceedings of the 31st International Conference on Machine Learning*, pages 118–126, 2014.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[RJ07] Filip Radlinski and Thorsten Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 570–579. ACM, 2007.

[WJJ13] Fabian Wauthier, Michael Jordan, and Nebojsa Jojic. Efficient ranking from pairwise comparisons. In *Proceedings of the 30th International Conference on Machine Learning*, pages 109–117, 2013.

[YJJJ13] Jinfeng Yi, Rong Jin, Shaili Jain, and Anil Jain. Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[Yu05] Hwanjo Yu. Svm selective sampling for ranking with application to data retrieval. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 354–363. ACM, 2005.

[ZCSZ07] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294. ACM, 2007.