Active Learning Ranking from Pairwise Preferences with Almost Optimal Query Complexity

Nir Ailon*
Technion, Haifa, Israel nailon@cs.technion.ac.il

Abstract

Given a set V of n elements we wish to linearly order them using pairwise preference labels which may be non-transitive (due to irrationality or arbitrary noise). The goal is to linearly order the elements while disagreeing with as few pairwise preference labels as possible. Our performance is measured by two parameters: The number of disagreements (loss) and the query complexity (number of pairwise preference labels). Our algorithm adaptively queries at most $O(n\,\mathrm{poly}(\log n, \varepsilon^{-1}))$ preference labels for a regret of ε times the optimal loss. This is strictly better, and often significantly better than what non-adaptive sampling could achieve. Our main result helps settle an open problem posed by learning-to-rank (from pairwise information) theoreticians and practitioners: What is a provably correct way to sample preference labels?

1 Introduction

We study the problem of learning to rank from pairwise preferences, and solve an open problem that has led to development of many heuristics but no provable results. The input is a set V of n elements from some universe, and we wish to linearly order them given pairwise preference labels, given as response to *which is preferred, u or v?* for pairs $u, v \in V$. The goal is to linearly order the elements from the most preferred to the least preferred, while disagreeing with as few pairwise preference labels as possible. Our performance is measured by two parameters: The loss (number of disagreements) and query complexity (number of preference responses we need). This is a learning problem, with a finite sample space of $\binom{n}{2}$ possibilities only (hence a transductive learning problem). The loss minimization problem given the entire $n \times n$ preference matrix is a well known NP-hard problem called MFAST (minimum feedback arc-set in tournaments) [5]. Recently, Kenyon and Schudy [23] have devised a PTAS for it, namely, a polynomial (in n) -time algorithm computing a solution with loss at most $(1+\varepsilon)$ the optimal, for and $\varepsilon > 0$ (the degree of the polynomial there may depend on ε). In our case each edge from the input graph is given for a unit cost, hence we seek query efficiency. Our algorithm samples preference labels non-uniformly and adaptively, hence we obtain an *active* learning algorithm. Our output is not a solution to MFAST, but rather a reduction of the original learning problem to a simpler one decomposed into small instances in which the optimal loss is high, consequently, uniform sampling of preferences can be shown to be sufficiently good.

Our Setting vs. The Usual "Learning to Rank" Problem. Our setting defers from much of the *learning to rank* (LTR) literature. Usually, the labels used in LTR problems are responses to individual elements, and not to pairs of elements. A typical example is the $1..5$ scale rating for restaurants, or $0, 1$ rating (irrelevant/relevant) for candidate documents retrieved for a query (known as the *binary ranking* problem). The preference graph induced from these labels is transitive, hence no combinatorial problems arise due to nontransitivity. We do not discuss this version of LTR. Some LTR literature *does* consider the pairwise preference label approach, and there is much justification to it (see [11, 22] and reference therein). Other works (e.g. [26]) discuss pairwise or higher order

*Supported by a Marie Curie International Reintegration Grant PIRG07-GA-2010-268403

(*listwise*) approaches, but a close inspection reveals that they do not use pairwise (or listwise) labels, only pairwise (or listwise) loss functions.

Using Kenyon and Schudy's PTAS as a starting point. As mentioned above, our main algorithm is derived from the PTAS of [23], but with a significant difference. We use their algorithm to obtain a certain decomposition of the input. A key change to their algorithm, which is not query efficient, involves careful sampling followed by iterated sample refreshing steps.

Our work can be studied in various contexts, aside from LTR. **Machine Learning Reductions:** Our main algorithm reduces a given instance to smaller subproblems decomposing it. We mention other work in this vein: [6, 3, 9]. **Active Learning:** An important field of statistical learning theory and practice ([8, 21, 15, 14, 24, 17, 13, 20, 16, 13]). In the most general setting, one wishes to improve on standard statistical learning theoretical complexity bounds by actively choosing instances for labels. Many heuristics have been developed, while algorithms with provable bounds (especially in the agnostic case) are known for few problems, often toys. General bounds are difficult to use: [8] provides general purpose active learning bounds which are quite difficult to use in actual specific problems; The A2 algorithm [7], analyzed in [21] using the disagreement coefficient is not useful here. It can be shown that the disagreement coefficient here is trivial (omitted due to lack of space). **Noisy Sorting:** There is much literature in theoretical computer science on sorting noisy data. [10] work in a Bayesian setting; In [19], the input preference graph is transitive, and labels are nondeterministic. In other work, elements from the set of alternatives are assumed to have a latent value. In this work the input is worst case and not Bayesian, query responses are deterministic and elements do not necessarily have a latent value.

Paper Organization: Section 2 presents basic definitions and lemmata, and in particular defines what a good decomposition is and how it can be used in learning permutations from pairwise preferences. Section 3 presents our main active learning algorithm which is, in fact, an algorithm for producing a good decomposition query efficiently. The main result is presented in Theorem 3.1. Section 4 discusses future work and followup work appearing in the full version of this paper.

2 Notation and Basic Lemmata

Let V denote a finite set of size n that we wish to rank.[1] We assume an unknown preference function W on pairs of elements in V, which is unknown to us. For any pair $u, v \in V$, $W(u, v)$ is 1 if u is deemed preferred over v, and 0 otherwise. We enforce $W(u, v) + W(v, u) = 1$ (no abstention) hence, (V, W) is a tournament. We assume that W is *agnostic*: it is not necessarily transitive and may contain errors and inconsistencies. For convenience, for any two real numbers a, b we will let $[a, b]$ denote the interval $\{x : a \leq x \leq b\}$ if $a \leq b$ and $\{x : b \leq x \leq a\}$ otherwise.

We wish to predict W using a hypothesis h from concept class $\mathcal{H} = \Pi(V)$, where $\Pi(V)$ is the set of permutations π over V viewed equivalently as binary functions over $V \times V$ satisfying, for all $u, v, w \in V$, $\pi(u, v) = 1 - \pi(v, u)$ and $\pi(u, w) = 1$ whenever $\pi(u, v) = \pi(v, w) = 1$. For $\pi \in \Pi(V)$ we also use notation: $\pi(u, v) = 1$ if and only if $u \prec_\pi v$, namely, if u precedes v in π. Abusing notation, we also view permutations as injective functions from $[n]$ to V, so that the element $\pi(1) \in V$ is in the first, most preferred position and $\pi(n)$ is the least preferred one. We also define the function ρ_π inverse to π as the unique function satisfying $\pi(\rho_\pi(v)) = v$ for all $v \in V$. Hence, $u \prec_\pi v$ is equivalent to $\rho_\pi(u) < \rho_\pi(v)$.) As in standard ERM, we define a risk function $C_{u,v}$ penalizing the error of π with respect to the pair u, v, namely, $C_{u,v}(\pi, V, W) = \mathbf{1}_{\pi(u,v) \neq W(u,v)}$. The total loss, $C(h, V, W)$ is defined as $C_{u,v}$ summed over all unordered $u, v \in V$. Our goal is to devise an active learning algorithm for the purpose of minimizing this loss.

In this paper we show an improved, almost optimal statistical learning theoretical bound using recent important breakthroughs in combinatorial optimization of a related problem called *minimum feedback arc-set in tournaments* (MFAST). The relation between this NP-Hard problem and our learning problem has been noted before in (eg [12]), when these breakthroughs were yet to be known.

MFAST is more precisely defined as follows: V and W are given in entirety (we pay no price for reading W), and we seek $\pi \in \Pi(V)$ minimizing the MFAST cost $C(\pi, V, W)$. A PTAS has been

[1] In a more general setting we are given a sequence $V^1, V^2, \ldots$ of sets, but there is enough structure and interest in the single set case, which we focus on in this work.

discovered for this NP-Hard very recently in groundbreaking work by Kenyon and Schudy [23]. This PTAS is not useful however for the purpose of *learning to rank from pairwise preferences* because it is not query efficient. It may require to read all quadratically many entries in W. In this work we fix this drawback, and use the PTAS to obtain a certain useful decomposition.

Definition 2.1. *Given a set V of size n, an ordered decomposition is a list of pairwise disjoint subsets $V_1, \ldots, V_k \subseteq V$ such that $\cup_{i=1}^{k} V_i = V$. We let $W|_{V_i}$ denote the restriction of W to $V_i \times V_i$ for $i = 1, \ldots, k$. For a permutation $\pi \in \Pi(v)$ we let $\pi|_{V_i}$ denote its restriction to the elements of V_i (hence, $\pi|_{V_i} \in \Pi(V_i)$). We say that $\pi \in \Pi(V)$ respects $V_1, \ldots, V_k$ if for all $u \in V_i, v \in V_j, i < j$, $u \prec_\pi v$. We denote the set of permutations $\pi \in \Pi(V)$ respecting the decomposition $V_1, \ldots, V_k$ by $\Pi(V_1, \ldots, V_k)$. We say that a subset U of V is* small in V *if $|U| \leq \log n / \log\log n$, otherwise we say that U* is big in V. *A decomposition $V_1, \ldots, V_k$ is ε-*good *with respect to W if:*[2]

$$\text{Local Chaos:} \quad \min_{\pi \in \Pi(V)} \sum_{i:V_i \text{ big in } V} C(\pi_{|V_i}, V_i, W_{|V_i}) \geq \varepsilon^2 \sum_{i:V_i \text{ big in } V} \binom{n_i}{2}. \tag{2.1}$$

$$\text{Approximate Optimality:} \quad \min_{\sigma \in \Pi(V_1,\ldots,V_k)} C(\sigma, V, W) \leq (1+\varepsilon) \min_{\pi \in \Pi(V)} C(\pi, V, W)\ . \tag{2.2}$$

We will show how to use an ε-good decomposition, and how to obtain one query-efficiently.

Basic (suboptimal) results from statistical learning theory: Viewing pairs of V-elements as data points, the loss $C(\pi, V, W)$ is, up to normalization, an expected cost over a random draw of a data point. A sample E of unordered pairs gives rise to a *partial cost*, C_E defined as: $C_E(\pi, V, W) = \binom{n}{2}|E|^{-1} \sum_{\substack{(u,v)\in E \\ u \prec_\pi v}} W(v,u)$. (We assume throughout that E is chosen with repetitions and is hence a multiset; the accounting of parallel edges is clear.) $C_E(\cdot,\cdot,\cdot)$ is an *empirical unbiased estimator* of $C(\pi, V, W)$ if $E \subseteq \binom{V}{2}$ is chosen uniformly at random among all (multi)subsets of a given size. The basic question in statistical learning theory is, how good is the minimizer π of C_E, in terms of C? The notion of VC dimension [25] gives us a nontrivial (albeit suboptimal - see below) bound.

Lemma 2.2. *The VC dimension of the set of permutations on V, viewed as binary classifiers on pairs of elements, is $n-1$.*

It is easy to show that the VC dimension is at most $O(n \log n)$, which is the logarithm of the number of permutations. See [4] for a linear bound. The implications are:

Proposition 2.3. *If E is chosen uniformly at random (with repetitions) as a sample of m elements from $\binom{V}{2}$, where $m > n$, then with probability at least $1-\delta$ over the sample, all permutations π satisfy: $|C_E(\pi, V, W) - C(\pi, V, W)| = n^2 O\left(\sqrt{\frac{n \log m + \log(1/\delta)}{m}}\right)$.*

Hence, if we want to minimize $C(\pi, V, W)$ over π to within an additive error of μn^2 with probability at least $1-\delta$, it suffices to choose a sample E of $m = O(\mu^{-2}(n \log n + \log \delta^{-1}))$ elements from $\binom{V}{2}$ uniformly at random (with repetitions), and optimize $C_E(\pi, V, W)$ instead.[3] Assume $\delta \geq e^{-n}$, so that we get a more manageable sample bound of $O(\mu^{-2} n \log n)$. **Is this bound at all interesting?** For two permutations π, σ, the Kendall-Tau metric $d_\tau(\pi, \sigma)$ is defined as $d_\tau(\pi, \sigma) = \sum_{u \neq v} \mathbf{1}[(u \prec_\pi v) \wedge (v \prec_\sigma u)]$. The Spearman Footrule metric $d_{\text{foot}}(\pi, \sigma)$ is defined as $d_{\text{foot}}(\pi, \sigma) = \sum_u |\rho_\pi(u) - \rho_\sigma(u)|$. The following is well known [18]:

$$d_\tau(\pi, \sigma) \leq d_{\text{foot}}(\pi, \sigma) \leq 2 d_\tau(\pi, \sigma)\ . \tag{2.3}$$

Clearly $C(\cdot, V, \cdot)$ extends $d_\tau(\cdot,\cdot)$ to distances between permutations and binary tournaments, with the triangle inequality $d_\tau(\pi, \sigma) \leq C(\pi, V, W) + C(\sigma, V, W)$ satisfied for all W and $\pi, \sigma \in \Pi(V)$.

Assume we use Proposition 2.3 to find $\pi \in \Pi(V)$ with an additive regret of $O(\mu n^2)$ with respect to an optimal solution π^* for some $\mu > 0$. The triangle inequality implies $d_\tau(\pi, \pi^*) = \Omega(\mu n^2)$. By (2.3), hence, $d_{\text{foot}}(\pi, \pi^*) = \Omega(\mu n^2)$. By definition of d_{foot}, this means that the averege element $v \in V$ is translated $\Omega(\mu n)$ positions away from its position in π^*. In some applications (e.g. IR), one may

[2] We will just say ε-good if W is clear from the context.

[3] $\binom{V}{2}$ denotes the set of unordered pairs of distinct elements in V.

want elements to be at most a constant γ positions off. This translates to a sought regret of $O(\gamma n)$ for constant γ, and using our notation, to $\mu = \gamma/n$. Proposition 2.3 cannot guarantee less than a quadratic sample size for such a regret, which is tantamount to querying all of W. **We can do better:** For any $\varepsilon > 0$ we achieve an additive regret of $O(\varepsilon C(\pi^*, V, W))$ using $O(\mathrm{poly}(\log n, \varepsilon^{-1}))$ W-queries, for arbitrarily small optimal loss $C(\pi^*, V, W)$. This is not achievable using Proposition 2.3. One may argue that the VC bound may be too pessimistic, and other arguments may work for the uniform sample case. A simple extremal case (omitted from this abstract) shows that this is false.

Proposition 2.4. *Let $V_1, \ldots, V_k$ be an ordered decomposition of V. Let $\mathcal{B}$ denote the set of indices $i \in [k]$ such that V_i is big in V. Assume E is chosen uniformly at random (with repetitions) as a sample of m elements from $\bigcup_{i\in\mathcal{B}} \binom{V_i}{2}$, where $m > n$. For each $i = 1, \ldots, k$, let $E_i = E \cap \binom{V_i}{2}$. Define $C_E(\pi, \{V_1, \ldots, V_k\}, W)$ to be*

$C_E(\pi, \{V_1, \ldots, V_k\}, W) = \left(\sum_{i\in\mathcal{B}} \binom{n_i}{2}\right) |E|^{-1} \sum_{i\in\mathcal{B}} \binom{n_i}{2}^{-1} |E_i| C_{E_i}(\pi_{|V_i}, V_i, W_{|V_i}) $ *. (The normalization is defined so that the expression is an unbiased estimator of $\sum_{i\in\mathcal{B}} C(\pi_{|V_i}, V_i, W_{|V_i})$. If $|E_i| = 0$ for some i, formally define $\binom{n_i}{2}^{-1} |E_i| C_{E_i}(\pi_{|V_i}, V_i, W_{|V_i}) = 0$.) Then with probability at least $1 - e^{-n}$ over the sample, all permutations $\pi \in \Pi(V)$ satisfy:*

$$\left|C_E(\pi, \{V_1, \ldots, V_k\}, W) - \sum_{i\in\mathcal{B}} C(\pi|_{V_i}, V_i, W|_{V_i})\right| = \sum_{i\in\mathcal{B}} \binom{n_i}{2} O\left(\sqrt{\frac{n \log m + \log(1/\delta)}{m}}\right).$$

The proof (omitted from this abstract) uses simple VC dimension arithmetic. **Now, why is ε-goodness good?**

Lemma 2.5. *Fix $\varepsilon > 0$ and assume we have an ε-good partition (Definition 2.1) $V_1, \ldots, V_k$ of V. Let $\mathcal{B}$ denote the set of $i \in [k]$ such that V_i is big in V, and let $\bar{\mathcal{B}} = [k] \setminus \mathcal{B}$. Let $n_i = |V_i|$ for $i = 1, \ldots, n$, and let E denote a random sample of $O(\varepsilon^{-6} n \log n)$ elements from $\bigcup_{i\in\mathcal{B}} \binom{V_i}{2}$, each element chosen uniformly at random with repetitions. Let E_i denote $E \cap \binom{V_i}{2}$. Let $C_E(\pi, \{V_1, \ldots, V_k\}, W)$ be defined as in Proposition 2.4. For any $\pi \in \Pi(V_1, \ldots, V_k)$ define:*

$$\tilde{C}(\pi) := C_E(\pi, \{V_1, \ldots, V_k\}, W) + \sum_{i\in\bar{\mathcal{B}}} C(\pi_{|V_i}, V_i, W_{|V_i}) + \sum_{1\le i<j\le k} \sum_{(u,v)\in V_i\times V_j} \mathbf{1}_{v\prec_\pi u} \,. \quad (2.4)$$

Then the following event occurs with probability at least $1 - e^{-n}$: For any minimizer σ^* of $\tilde{C}(\cdot)$ over $\Pi(V_1, \ldots, V_k)$: $C(\sigma^*, V, W) \le (1 + 2\varepsilon) \min_{\pi\in\Pi(V)} C(\pi, V, W)$.

(Proof omitted from abstract.) The consequence: Given an ε-good decomposition $V_1, \ldots, V_k$, optimizing $\tilde{C}(\sigma)$ over $\sigma \in \Pi(V_1, \ldots, V_k)$, would give a solution with *relative regret* of 2ε w.r.t. the optimum. The first and last terms in the RHS of (2.4) require no more than $O(\varepsilon^{-6} n \log n)$ W-queries to compute (by definition of E, and given the decomposition). The middle term runs over small V_i's, and can be computed from $O(n \log n / \log\log n)$ W-queries. If we now assume that a good decomposition can be efficiently computed using $O(n\,\mathrm{polylog}(n, \varepsilon^{-1}))$ W-queries (as we indeed show), then we would beat the VC bound whenever the optimal loss is at most $O(n^{2-\nu})$ for some $\nu > 0$.

3 A Query Efficient Algorithm for ε-Good Decompositions

Theorem 3.1. *Given a set V of size n, a preference oracle W and an error tolerance parameter $0 < \varepsilon < 1$, there exists a $\mathrm{poly}(n, \varepsilon^{-1})$-time algorithm returning, with constant probabiliy, an ε-good partition of V, querying at most $O(\varepsilon^{-6} n \log^5 n)$ locations in W on expectation.*

Before describing the algorithm and its analysis, we need some definitions:

Definition 3.2. *Let π denote a permutation over V. Let $v \in V$ and $i \in [n]$. We define $\pi_{v\to i}$ to be the permutation obtained by moving the rank of v to i in π, and leaving the rest of the elements in the same order.*[4]

Definition 3.3. *Fix $\pi \in \Pi(V)$, $v \in V$ and $i \in [n]$. We define* $\mathrm{TestMove}(\pi, V, W, v, i) := C(\pi, V, W) - C(\pi_{v\to i}, V, W)$ *. Equivalently, if $i \ge \rho_\pi(v)$ then* $\mathrm{TestMove}(\pi, V, W, v, i) :=$

[4]For example, if $V = \{x, y, z\}$ and $(\pi(1), \pi(2), \pi(3)) = (x, y, z)$, then $(\pi_{x\to 3}(1), \pi_{x\to 3}(2), \pi_{x\to 3}(3)) = (y, z, x)$.

$\sum_{u:\rho_\pi(u)\in[\rho_\pi(v)+1,i]}(W_{uv}-W_{vu})$ *. A similar expression can be written for* $i<\rho_\pi(v)$*. For a multiset* $E\subseteq\binom{V}{2}$*, define* $\mathrm{TestMove}_E(\pi,V,W,v,i)$*, for* $i\geq\rho_\pi(v)$*, as* $\mathrm{TestMove}_E(\pi,V,W,v,i):=\frac{|i-\rho_\pi(v)|}{|\tilde{E}|}\sum_{u:(u,v)\in\tilde{E}}(W(u,v)-W(v,u))$*. where the multiset* $\tilde{E}$ *is defined as* $\{(u,v)\in E:\rho_\pi(u)\in[\rho_\pi(v)+1,i]\}$*. Similarly, for* $i<\rho_\pi(v)$ *we define* $\mathrm{TestMove}_E(\pi,V,W,v,i):=\frac{|i-\rho_\pi(v)|}{|\tilde{E}|}\sum_{u:(u,v)\in\tilde{E}}(W(v,u)-W(u,v))$*. where* $\tilde{E}$ *is now* $\{(u,v)\in E:\rho_\pi(u)\in[i,\rho_\pi(v)-1]\}$*.*

Lemma 3.4. *Fix* $\pi\in\Pi(V)$*,* $v\in V$*,* $i\in[n]$ *and an integer* N*. Let* $E\subseteq\binom{V}{2}$ *be a random (multi)-set of size* N *with elements* $(v,u_1),\ldots,(v,u_N)$*, drawn so that for each* $j\in[N]$ *the element* u_j *is chosen uniformly at random from among the elements lying between* v *(exclusive) and position* i *(inclusive) in* π*. Then* $\mathbf{E}[\mathrm{TestMove}_E(\pi,V,W,v,i)]=\mathrm{TestMove}(\pi,V,W,v,i)$*. Additionally, for any* $\delta>0$*, except with probability of failure* δ*,*

$$|\,\mathrm{TestMove}_E(\pi,V,W,v,i)-\mathrm{TestMove}(\pi,V,W,v,i)|=O\left(|i-\rho_\pi(v)|\sqrt{\tfrac{\log\delta^{-1}}{N}}\right).$$

The lemma is easily proven using Hoeffding tail bounds, using the fact that $|W(u,v)|\leq 1$ for all u,v. Our decomposition algorithm $\mathrm{SampleAndRank}$ is detailed in Algorithm 1, with subroutines in Algorithms 2 and 3. It is a query efficient improvement of the PTAS in [23] with the following difference: here we are not interested in an approximation algorithm for MFAST, but just in an ε-good decomposition. Whenever we reach a small block (line 3) or a big block with a probably approximately sufficiently high cost (line 8) in our recursion of Algorithm 2), we simply output it as a block in our partition. Denote the resulting outputted partition by $V_1,\ldots,V_k$. Denote by $\hat{\pi}$ the minimizer of $C(\cdot,V,W)$ over $\Pi(V_1,\ldots,V_k)$. We need to show that $C(\hat{\pi},V,W)\leq(1+\varepsilon)\min_{\pi\in\Pi(V)}C(\pi,V,W)$, thus establishing (2.2). The analysis closely follows [23]. Due to space limitations, we focus on the differences, and specifically on Procedure $\mathrm{ApproxLocalImprove}$ (Algorithm 3), replacing a greedy local improvement step in [23] which is *not* query efficient.

$\mathrm{SampleAndRank}$ (Algorithm 1) takes the following arguments: The set V, the preference matrix W and an accuracy argument ε. It is implicitly understood that the argument W passed to $\mathrm{SampleAndRank}$ is given as a query oracle, incurring a unit cost upon each access. The first *warm start* step in $\mathrm{SampleAndRank}$ computes an expected constant factor approximation π to MFAST on V,W using QuickSort [2]. The query complexity of this step is $O(n\log n)$ *on expectation* (see [3]). Before continuing, we make the following assumption, which holds with constant probability using Markov probability bounds.

Assumption 3.5. *The cost* $C(\pi,V,W)$ *of* π *computed in line 2 of* $\mathrm{SampleAndRank}$ *is* $O(1)$ *times that of the optimal* π^**, and the query cost incurred in the computation is* $O(n\log n)$*.*

Next, a recursive procedure $\mathrm{SampleAndDecompose}$ is called, running a divide-and-conquer algorithm. Before branching, it executes the following: Lines 5 to 9 identify local chaos (2.1) (with high probability). Line 10 calls $\mathrm{ApproxLocalImprove}$ (Algorithm 3), responsible for performing query-efficient approximate greedy steps, as we now explain.

Approximate local improvement steps. $\mathrm{ApproxLocalImprove}$ takes a set V of size N, W, a permutation π on V, two numbers C_0, ε and an integer n.[5] The number n is always the size of the input in the root call to $\mathrm{SampleAndDecompose}$, passed down in the recursion, and used for the purpose of controlling success probabilities. The goal of is to repeatedly identify w.h.p. single vertex moves that considerably decrease the cost. The procedure starts by creating a *sample ensemble* $\mathcal{S}=\{E_{v,i}:v\in V,i\in[B,L]\}$, where $B=\log\lfloor\Theta(\varepsilon N/\log n)\rfloor$ and $L=\lceil\log N\rceil$. The size of each $E_{v,i}\in\mathcal{S}$ is $\Theta(\varepsilon^{-2}\log^2 n)$, and each element $(v,x)\in E_{v,i}$ was added (with possible multiplicity) by uniformly at random selecting, with repetitions, an element $x\in V$ positioned at distance at most 2^i from the position of v in π. Let $\mathcal{D}_\pi$ denote the distribution space from which $\mathcal{S}$ was drawn, and let $\Pr_{X\sim\mathcal{D}_\pi}[X=\mathcal{S}]$ denote the probability of obtaining a given sample ensemble $\mathcal{S}$. $\mathcal{S}$ will enable us to approximate the improvement in cost obtained by moving a single element u to position j.

Definition 3.6. *Fix* $u\in V$ *and* $j\in[n]$*, and assume* $\log|j-\rho_\pi(u)|\geq B$*. Let* $\ell=\lceil\log|j-\rho_\pi(u)|\rceil$*. We say that* $\mathcal{S}$ *is successful at* u,j *if* $|\{x:(u,x)\in E_{u,\ell}\}\cap\{x:\rho_\pi(x)\in[\rho_\pi(u),j]\}|=\Omega(\varepsilon^{-2}\log^2 n)$ *.*

[5] Notation abuse: V here is a subset of the original input.

Success of $\mathcal{S}$ at u, j means that sufficiently many samples $x \in V$ such that $\rho_\pi(x)$ is between $\rho_\pi(u)$ and j are represented in $E_{u,\ell}$. Conditioned on $\mathcal{S}$ being successful at u, j, note that the denominator from the definition of $\mathrm{TestMove}_E$ does not vanish, and we can thereby define:

Definition 3.7. *$\mathcal{S}$ is a* good approximation *at u, j if (defining ℓ as in Definition 3.6):*

$\left|\mathrm{TestMove}_{E_{u,\ell}}(\pi, V, W, u, j) - \mathrm{TestMove}(\pi, V, W, u, j)\right| \leq \frac{1}{2}\varepsilon|j - \rho_\pi(u)|/\log n$ *. $\mathcal{S}$ is a good approximation if it is succesful and a good approximation at all $u \in V$, $j \in [n]$ satisfying $\lceil \log |j - \rho_\pi(u)| \rceil \in [B, L]$.*

Using Chernoff to ensure success and Hoeffding to ensure good approximation, union bounding:

Lemma 3.8. *Except with probability $1 - O(n^{-4})$, $\mathcal{S}$ is a good approximation.*

Algorithm 1 $\mathrm{SampleAndRank}(V, W, \varepsilon)$

1: $n \leftarrow |V|$
2: $\pi \leftarrow$ Expected $O(1)$-approx solution to MFAST using $O(n \log n)$ W-queries on expectation using QuickSort [2]
3: **return** $\mathrm{SampleAndDecompose}(V, W, \varepsilon, n, \pi)$

Algorithm 2 $\mathrm{SampleAndDecompose}(V, W, \varepsilon, n, \pi)$

1: $N \leftarrow |V|$
2: **if** $N \leq \log n / \log \log n$ **then**
3: **return** trivial partition $\{V\}$
4: **end if**
5: $E \leftarrow$ random subset of $O(\varepsilon^{-4} \log n)$ elements from $\binom{V}{2}$ (with repetitions)
6: $C \leftarrow C_E(\pi, V, W)$ (C is an additive $O(\varepsilon^2 N^2)$ approximation of C w.p. $\geq 1 - n^{-4}$)
7: **if** $C = \Omega(\varepsilon^2 N^2)$ **then**
8: **return** trivial partition $\{V\}$
9: **end if**
10: $\pi_1 \leftarrow \mathrm{ApproxLocalImprove}(V, W, \pi, \varepsilon, n)$
11: $k \leftarrow$ random integer in the range $[N/3, 2N/3]$
12: $V_L \leftarrow \{v \in V : \rho_\pi(v) \leq k\}$, $\pi_L \leftarrow$ restriction of π_1 to V_L
13: $V_R \leftarrow V \setminus V_L$, $\pi_R \leftarrow$ restriction of π_1 to V_R
14: **return** concatenation of $\mathrm{SampleAndDecompose}(V_L, W, \varepsilon, n, \pi_L)$, $\mathrm{SampleAndDecompose}(V_R, W, \varepsilon, n, \pi_R)$

Mutating the Pair Sample To Reflect a Single Element Move. Line 16 in $\mathrm{ApproxLocalImprove}$ requires elaboration. In lines 15-18 we sought (using $\mathcal{S}$) an element u and position j, such that moving u to j (giving rise to $\pi_{u\to j}$) would considerably improve the cost w.h.p. If such an element u existed, we executed the exchange $\pi \leftarrow \pi_{u\to j}$. Unfortunately the sample ensemble $\mathcal{S}$ becomes *stale*: even if $\mathcal{S}$ was a good approximation, it is no longer necessarily so w.r.t. the new value of π. We refresh it in line 16 by applying a transformation $\varphi_{u\to j}$ on $\mathcal{S}$, resulting in a new sample ensemble $\varphi_{u\to j}(\mathcal{S})$ approximately distributed by $\mathcal{D}_{\pi_{u\to j}}$. More precisely, φ (defined below) is such that

$$\varphi_{u\to j}(\mathcal{D}_\pi) = D_{\pi_{u\to j}}\ , \tag{3.1}$$

where the left hand side denotes the distribution obtained by drawing from $\mathcal{D}_\pi$ and applying $\varphi_{u\to j}$ to the result. We now define $\varphi_{u\to j}$. Denoting $\varphi_{u\to j}(\mathcal{S}) = \mathcal{S}' = \{E'_{v,i} : v \in V, i \in [B, L]\}$, we need to define each $E'_{v,i}$.

Definition 3.9. *$E_{v,i}$ is* interesting *in the context of π and $\pi_{u\to j}$ if the two sets T_1, T_2 defined as $T_1 = \{x \in V : |\rho_\pi(x) - \rho_\pi(v)| \leq 2^i\}, T_2 = \{x \in V : |\rho_{\pi_{u\to j}}(x) - \rho_{\pi_{u\to j}}(v)| \leq 2^i\}$ differ.*

We set $E'_{v,i} = E_{v,i}$ for all v, i for which $E_{v,i}$ is *not* interesting. Fix one interesting choice v, i. Let T_1, T_2 be as in Defintion 3.9. It can be easily shown that each of T_1 and T_2 contains $O(1)$ elements that are not contained in the other, and it can be assumed (using a simple clipping argument - omitted) that this number is exactly 1, hence $|T_1| = |T_2|$. let $X_1 = T_1 \setminus T_2$, and $X_2 = T_2 \setminus T_1$. Fix any injection $\alpha : X_1 \to X_2$, and extend $\alpha : T_1 \to T_2$ so that $\alpha(x) = x$ for all $x \in T_1 \cap T_2$. Finally,

Algorithm 3 ApproxLocalImprove$(V, W, \pi, \varepsilon, n)$ (*N*ote: π used as both input and output)

1: $N \leftarrow |V|$, $B \leftarrow \lceil \log(\Theta(\varepsilon N / \log n)) \rceil$, $L \leftarrow \lceil \log N \rceil$
2: **if** $N = O(\varepsilon^{-3} \log^3 n)$ **then**
3: **return**
4: **end if**
5: **for** $v \in V$ **do**
6: $r \leftarrow \rho_\pi(v)$
7: **for** $i = B \dots L$ **do**
8: $E_{v,i} \leftarrow \emptyset$
9: **for** $m = 1..\Theta(\varepsilon^{-2} \log^2 n)$ **do**
10: $j \leftarrow$ integer uniformly at random chosen from $[\max\{1, r - 2^i\}, \min\{n, r + 2^i\}]$
11: $E_{v,i} \leftarrow E_{v,i} \cup \{(v, \pi(j))\}$
12: **end for**
13: **end for**
14: **end for**
15: **while** $\exists u \in V$ and $j \in [n]$ s.t. *(setting $\ell := \lceil \log |j - \rho_\pi(u)| \rceil$):*
$\ell \in [B, L]$ and $\text{TestMove}_{E_{u,\ell}}(\pi, V, W, u, j) > \varepsilon |j - \rho_\pi(u)| / \log n$ **do**
16: For $v \in V$, $i \in [B, L]$ refresh $E_{v,i}$ w.r.t. the move $u \to j$ using $\varphi_{u \to j}$ (Section 3)
17: $\pi \leftarrow \pi_{u \to j}$
18: **end while**

define $E'_{v,i} = \{(v, \alpha(x)) : (v, x) \in E_{v,i}\}$. For $v = u$ we create $E'_{v,i}$ from scratch by repeating the loop in line 7 for that v. It is easy to see that (3.1) holds. By Lemma 3.8, the total variation distance between ($\mathcal{D}_\pi|$ good approximation) and $\mathcal{D}_{\pi_{u \to j}}$ is $O(n^{-4})$. Using a simple chain rule argument:

Lemma 3.10. *Fix π^0 on V of size N, and fix $u_1, \dots, u_k \in V$ and $j_1, \dots, j_k \in [n]$. Draw $\mathcal{S}^0$ from $\mathcal{D}_{\pi^0}$, and define $\mathcal{S}^1 = \varphi_{u_1 \to j_1}(\mathcal{S}^0), \mathcal{S}^2 = \varphi_{u_2 \to j_2}(\mathcal{S}^1), \cdots, \mathcal{S}^k = \varphi_{u_k \to j_k}(\mathcal{S}^{k-1})$, $\pi^1 = \pi^0_{u_1 \to j_1}, \pi^2 = \pi^1_{u_2 \to j_2}, \cdots, \pi^k = \pi^{k-1}_{u_k \to j_k}$. Consider the random variable S^k conditioned on $\mathcal{S}^0, \mathcal{S}^1, \dots, \mathcal{S}^{k-1}$ being good approximations for $\pi_0, \dots, \pi^{k-1}$, respectively. Then the total variation distance between the distribution of $\mathcal{S}^k$ and the distribution $(\mathcal{D}_{\pi^k}|\pi^k)$ (corresponding to the process of obtaning π^k and drawing from D_{π^k} "from scratch") is at most $O(kn^{-4})$.*

The difference between $\mathcal{S}$ and $\mathcal{S}'$, defined as $\text{dist}(\mathcal{S}, \mathcal{S}') := \left| \bigcup_{v,i} E_{v,i} \Delta E'_{v,i} \right|$ bounds the query complexity of computing mutations. The proof of the following has been omitted from this abstract.

Lemma 3.11. *Assume $\mathcal{S} \sim \mathcal{D}_\pi$ for some π, and $\mathcal{S}' = \varphi_{u \to j}$. Then* $\mathbf{E}[\text{dist}(\mathcal{S}, \mathcal{S}')] = O(\varepsilon^{-3} \log^3 n)$.

Analysis of SampleAndDecompose**.** Various *high probability* events must occur in order for the algorithm guarantees to hold. Let $\mathcal{E}_1$ denote the event that the first $\Theta(n^4)$ sample ensembles $\mathcal{S}_1, \mathcal{S}_2, \dots$ ApproxLocalImprove, either in lines 5 and 14, or via mutations, are good approximations By Lemmas 3.8 and 3.10, using a union bound, with constant probability (say, 0.99) this happens. Let $\mathcal{E}_2$ denote the event that the cost approximations obtained in line 5 of SampleAndDecompose are successful at all recursive calls. By Hoeffding tail bounds, this happens with probability $1 - O(n^{-4})$ for each call, there are $O(n \log n)$ calls, hence we can lower bound the probability of success of all executions by 0.99. Concluding, the following holds with probability at least 0.97:

Assumption 3.12. *Events $\mathcal{E}_1$ and $\mathcal{E}_2$ hold true.*

We condition what follows on this assumption.[6] Let π^* denote the optimal permutation for the root call to SampleAndDecompose with V, W, ε. The permutation π is, by Assumption 3.5, a constant factor approximation for π^*. By the triangle inequality, $d_\tau(\pi, \pi^*) \leq C(\pi, V, W) + C(\pi^*, V, W)$, hence, $E[d_\tau(\pi, \pi^*)] = O(C(\pi^*, V, W))$. From this, using (2.3), $E[d_{\text{foot}}(\pi, \pi^*)] = O(C(\pi^*, V, W))$. Now consider the recursion tree $\mathcal{T}$ of SampleAndDecompose. Denote $\mathcal{I}$ the set of internal nodes, and by $\mathcal{L}$ the set of leaves (i.e. executions exiting from line 8). For a call SampleAndDecompose corresponding to a node X, denote the input arguments by $(V_X, W, \varepsilon, n, \pi_X)$. Let $L[X], R[X]$ denote the left and right children of X respectively. Let k_X

[6] This may bias some expectation upper bounds derived earlier and in what follows. This bias can multiply the estimates by at most $1/0.97$, which can be absorbed in our O-notations.

denote the integer k in 11 in the context of $X \in \mathcal{I}$. Hence, by our definitions, $V_{L[X]}, V_{R[X]}, \pi_{L[X]}$ and $\pi_{R[X]}$ are precisely V_L, V_R, π_L, π_R from lines 12-13 in the context of node X. Take, as in line 1, $N_X = |V_X|$. Let π^*_X denote the optimal MFAST solution for instance $(V_X, W_{|V_X})$. By $\mathcal{E}_1$ we conclude that the cost of $\pi_{X\,u\to j}$ is always an actual improvement compared to π_X (for the current value of π_X, u and j in iteration), and the improvement in cost is of magnitude at least $\Omega(\varepsilon|\rho_{\pi_X}(u) - j|/\log n)$, which is $\Omega(\varepsilon^2 N_X/\log^2 n)$ due to the use of B defined in line 1.[7] But then the number of iterations of the while loop in line 15 of $\mathrm{ApproxLocalImprove}$ is $O(\varepsilon^{-2}C(\pi_X, V_X, W_{|V_X}) \log^2 n/N_X)$ (Otherwise the true cost of the running solution would go below 0.) Since $C(\pi_X, V_X, W_{|V_X}) \leq \binom{N_X}{2}$, the number of iterations is hence at most $O(\varepsilon^{-2} N_X \log^2 n)$. By Lemma 3.11 the expected query complexity incurred by the call to $\mathrm{ApproxLocalImprove}$ is therefore $O(\varepsilon^{-5} N_X \log^5 n)$. Summing over the recursion tree, the total query complexity incurred by calls to $\mathrm{ApproxLocalImprove}$ is, on expectation, at most $O(\varepsilon^{-5} n \log^6 n)$. Now consider the moment at which the while loop of $\mathrm{ApproxLocalImprove}$ terminates. Let π_{1X} denote the permutation obtained at that point, returned to $\mathrm{SampleAndDecompose}$ in line 10. We classify the elements $v \in V_X$ to two families: V_X^{short} denotes all $u \in V_X$ s.t. $|\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)| = O(\varepsilon N_X/\log n)$, and V_X^{long} denotes $V_X \setminus V_X^{\mathrm{short}}$. We know by assumption, that the last sample ensemble $\mathcal{S}$ used in $\mathrm{ApproxLocalImprove}$ was a good approximation, hence for all $u \in V_X^{\mathrm{long}}$: (*) $\mathrm{TestMove}(\pi_{1X}, V_X, W_{|V_X}, u, \rho_{\pi^*_X}(u)) = O(\varepsilon|\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)|/\log n)$. Following [23], we say for $u \in V_X$ that u *crosses* k_X if $[\rho_{\pi_{1X}}(u), \rho_{\pi^*_X}(u)]$ contains k_X. Let V_X^{cross} denote the (random) set of elements $u \in V_X$ that cross k_X. We define a key quantity as in [23]: $T_X := \sum_{u \in V_X^{\mathrm{cross}}} \mathrm{TestMove}(\pi_{1X}, V_X, W_{|V_X}, u, \rho_{\pi^*_X}(u))$. Following (*), the elements $u \in V_X^{\mathrm{long}}$ can contribute at most $O\left(\varepsilon \sum_{u \in V_X^{\mathrm{long}}} |\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)|/\log n\right)$ to T_X. This latter bound is, by definition, $O(\varepsilon d_{\mathrm{foot}}(\pi_{1X}, \pi^*_X)/\log n)$ which is, using (2.3) at most $O(\varepsilon d_\tau(\pi_{1X}, \pi^*_X)/\log n)$. By the triangle inequality and the definition of π^*_X, the last expression is $O(\varepsilon C(\pi_{1X}, V_X, W_{|V_X})/\log n)$. How much can elements in V_X^{short} contribute to T_X? The probability of each such element to cross k is $O(|\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)|/N_X)$. Hence, the total expected contribution is $O\left(\sum_{u \in V_X^{\mathrm{short}}} |\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)|^2/N_X\right)$. Under the constraints $\sum_{u \in V_X^{\mathrm{short}}} |\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)| \leq d_{\mathrm{foot}}(\pi_{1X}, \pi^*_X)$ and $|\rho_{\pi_{1X}}(u) - \rho_{\pi^*_X}(u)| = O(\varepsilon N_X/\log n)$, this is $O(d_{\mathrm{foot}}(\pi_{1X}, \pi^*_X)\varepsilon N_X/(N_X \log n)) = O(d_{\mathrm{foot}}(\pi_{1X}, \pi^*_X)\varepsilon/\log n)$. Again using (2.3) and the triangle inequality, the bound becomes $O(\varepsilon C(\pi_{1X}, V_X, W_{|V_X})/\log n)$. Combining for V^{long} and V^{short}, we conclude: (**) $E_{k_X}[T_X] = O(\varepsilon C(\pi^*_X, V_X, W_{|V_X})/\log n)$, (the expectation is over the choice of k_X.) The bound (**) is the main improvement over [23], and should be compared with Lemma 3.2 there, stating (in our notation) $T_X = O(\varepsilon C^* N_X/(4n \log n))$. The latter bound is more restrictive than ours in certain cases, and obtaining it relies on a procedure that cannot be performed without having access W in its entirety. (**) however can be achieved using efficient querying of W, as we have shown. The remaineder of the arguments leading to proof of Theorem 3.1 closely follow those in Section 4 of [23]. The details have been omitted from this abstract.

4 Future Work

We presented a statistical learning theoretical active learning result for pairwise ranking. The main vehicle was a query (and time) efficient decomposition procedure, reducing the problem to smaller ones in which the optimal loss is high and uniform sampling suffices. The main drawback of our result is the inability to use it in order to search in a limited subspace of permutations. A typical example of such a subspace is the case in which each element $v \in V$ has a corresponding feature vector in a real vector space, and we only seek permutations induced by linear score functions. In followup work, Ailon, Begleiter and Ezra [1] show a novel technique achieving a slightly better query complexity than here with a simpler proof, while also admitting search in restricted spaces.

Acknowledgements The author gratefully acknowledges the help of Warren Schudy with derivation of some of the bounds in this work. Special thanks to Ron Begleiter for helpful comments. Apologizes for omitting references to much relevant work that could not fit in this version's bibliography.

[7] This also bounds the number of times a sample ensemble is created by $O(n^4)$, as required by $\mathcal{E}_1$.

References

[1] Nir Ailon, Ron Begleiter, and Esther Ezra, *A new active learning scheme with applications to learning to rank from pairwise preferences*, arxiv.org/abs/1110.2136 (2011).

[2] Nir Ailon, Moses Charikar, and Alantha Newman, *Aggregating inconsistent information: Ranking and clustering*, J. ACM **55** (2008), no. 5.

[3] Nir Ailon and Mehryar Mohri, *Preference based learning to rank*, vol. 80, 2010, pp. 189–212.

[4] Nir Ailon and Kira Radinsky, *Ranking from pairs and triplets: Information quality, evaluation methods and query complexity*, WSDM, 2011.

[5] Noga Alon, *Ranking tournaments*, SIAM J. Discret. Math. **20** (2006), no. 1, 137–142.

[6] M. F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. B. Sorkin, *Robust reductions from ranking to classification*, Machine Learning **72** (2008), no. 1-2, 139–153.

[7] Maria-Florina Balcan, Alina Beygelzimer, and John Langford, *Agnostic active learning*, J. Comput. Syst. Sci. **75** (2009), no. 1, 78–89.

[8] Maria-Florina Balcan, Steve Hanneke, and Jennifer Vaughan, *The true sample complexity of active learning*, Machine Learning **80** (2010), 111–139.

[9] A. Beygelzimer, J. Langford, and P. Ravikumar, *Error-correcting tournaments*, ALT, 2009, pp. 247–262.

[10] M. Braverman and E. Mossel, *Noisy sorting without resampling*, SODA: Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms, 2008, pp. 268–276.

[11] B. Carterette, P. N. Bennett, D. Maxwell Chickering, and S. T. Dumais, *Here or there: Preference judgments for relevance*, ECIR, 2008.

[12] William W. Cohen, Robert E. Schapire, and Yoram Singer, *Learning to order things*, NIPS '97, 1998, pp. 451–457.

[13] D. Cohn, L. Atlas, and R. Ladner, *Improving generalization with active learning*, Machine Learning **15** (1994), no. 2, 201–221.

[14] A. Culotta and A. McCallum, *Reducing labeling effort for structured prediction tasks*, AAAI: Proceedings of the 20th national conference on Artificial intelligence, 2005, pp. 746–751.

[15] S. Dasgupta, *Coarse sample complexity bounds for active learning*, Advances in Neural Information Processing Systems 18, 2005, pp. 235–242.

[16] S. Dasgupta, A. Tauman Kalai, and C. Monteleoni, *Analysis of perceptron-based active learning*, Journal of Machine Learning Research **10** (2009), 281–299.

[17] Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni, *A general agnostic active learning algorithm*, NIPS, 2007.

[18] Persi Diaconis and R. L. Graham, *Spearman's footrule as a measure of disarray*, Journal of the Royal Statistical Society. Series B (Methodological) **39** (1977), no. 2, pp. 262–268.

[19] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, *Computing with unreliable information*, STOC: Proceedings of the 22nd annual ACM symposium on Theory of computing, 1990, pp. 128–137.

[20] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby, *Selective sampling using the query by committee algorithm*, Mach. Learn. **28** (1997), no. 2-3, 133–168.

[21] Steve Hanneke, *A bound on the label complexity of agnostic active learning*, ICML, 2007, pp. 353–360.

[22] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker, *Label ranking by learning pairwise preferences*, Artif. Intell. **172** (2008), no. 16-17, 1897–1916.

[23] Claire Kenyon-Mathieu and Warren Schudy, *How to rank with few errors*, STOC, 2007, pp. 95–103.

[24] Dan Roth and Kevin Small, *Margin-based active learning for structured output spaces*, 2006.

[25] V. N. Vapnik and A. Ya. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Prob. and its Applications **16** (1971), no. 2, 264–280.

[26] F. Xia, T-Y Liu, J. Wang, W. Zhang, and H. Li, *Listwise approach to learning to rank: theory and algorithm*, ICML '08, 2008, pp. 1192–1199.