Direct Learning to Rank and Rerank

Cynthia Rudin
Duke University

Yining Wang
Carnegie Mellon University

Abstract

Learning-to-rank techniques have proven to be extremely useful for prioritization problems, where we rank items in order of their estimated probabilities, and dedicate our limited resources to the top-ranked items. This work exposes a serious problem with the state of learning-to-rank algorithms, which is that they are based on convex proxies that lead to poor approximations. We then discuss the possibility of "exact" reranking algorithms based on mathematical programming. We prove that a relaxed version of the "exact" problem has the same optimal solution, and provide an empirical analysis.

1 Introduction

We are often faced with prioritization problems – how can we rank aircraft in order of vulnerability to failure? How can we rank patients in order of priority for treatment? When we have limited resources and need to make decisions on how to allocate them, these ranking problems become important. The quality of a ranked list is often evaluated in terms of *rank statistics*. The area under the receiver operator characteristic curve (AUC), which counts pairwise comparisons, is a rank statistic, but it does not focus on the top of a ranked list, and is not a good evaluation measure if we care about prioritization problems. For prioritization problems, we would use rank statistics that focus on the top of the ranked list, such as a weighted area under the curve that focuses on the left part of the curve. Then, since we evaluate our models using these rank statistics, we should aim to optimize them out-of-sample by optimizing them in-sample. The learning-to-rank field (also called supervised ranking) is built from this fundamental idea. Learning-to-rank is a natural fit for many prioritization problems. If we are able to improve the quality of a prioritization policy by even a small amount, it can have an important practical impact. Learning-to-rank can be used to prioritize mechanical equipment for repair (e.g., airplanes, as considered by Oza et al, 2009), it could be useful for prioritizing maintenance on the power grid (Rudin et al, 2012, 2010), it could be used for ranking medical workers in order of likelihood that they accessed medical records inappropriately (as considered by Menon et al, 2013), prioritizing safety inspections or lead paint inspections in dwellings (Potash et al, 2015), ranking companies in order of likeliness of committing tax violations (see Kong and Saar-Tsechansky, 2013), or ranking water pipes in order of vulnerability (as considered by Li et al, 2013), and in almost any domain where one measures the quality of results by rank statistics. Learning-to-rank algorithms have been used also in sentiment analysis (Kessler and Nicolov, 2009), natural language processing (Ji et al, 2006; Collins and Koo, 2005), image retrieval (Jain and Varma, 2011; Kang et al, 2011), and reverse-engineering product quality rating systems (Chang et al, 2012).

This work exposes a serious problem with the state of learning-to-rank algorithms, which is that they are based on convex proxies for rank statistics, and when these convex proxies are used, computation is faster but the quality of the solution can be poor.

We then discuss the possibility of more direct optimization of rank statistics for predictive learning-to-rank problems. In particular, we consider a strategy of ranking with a simple ranker (logistic regression for instance) which is computationally efficient, and then reranking only the candidates near the top of the ranked list with an "exact" method. The exact method does not have the shortcoming that we discussed earlier for convex proxies.

For most ranking applications, we care only about the top of the ranked list; thus, as long as we rerank enough items with the exact method, the re-ranked list is (for practical purposes) just as useful as a full ranked list would be (if we could compute it with the

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

exact method, which would be computationally prohibitive).

The best known theoretical guarantee on ranking methods is obtained by directly optimizing the rank statistic of interest (as shown by theoretical bounds of Clemençon and Vayatis, 2008; Rudin and Schapire, 2009, for instance) hence our choice of methodology – mixed-integer programming (MIP) – for reranking in this work. Our general formulation can optimize any member of a large class rank statistics using a single mixed-integer *linear* program. Specifically, we can handle (a generalization of) the large class of *conditional linear rank statistics*, which includes the Wilcoxon-Mann Whitney U statistic, or equivalently the Area Under the ROC Curve, the Winner-Take-All statistic, the Discounted Cumulative Gain used in information retrieval (Järvelin and Kekäläinen, 2000), and the Mean Reciprocal Rank.

Exact learning-to-rank computations need to be performed carefully; we should not refrain from solving hard problems, but certain problems are harder than others. We provide two MIP formulations aimed at the same ranking problems. The first one works no matter what the properties of the data are. The second formulation is much faster, and is theoretically shown to produce the *same* quality of result as the first formulation when there are no duplicated observations. Note that if the observations are chosen from a continuous distribution then duplicated observations do not occur, with probability one.

One challenge in the exact learning-to-rank formulation is the way of handling ties in score. As it turns out, the original definition of conditional linear rank statistics can be used for the purpose of evaluation but not optimization. We show that a small change to the definition can be used for optimization.

This paper differs from our earlier technical report and non-archival conference paper (Chang et al, 2011, 2010), which were focused on solving full problems to optimality, and did not consider reranking or regularization; our exposition for the formulations closely follows this past work. The technique was used by Chang et al (2012) for the purpose of reverse engineering product rankings from rating companies that do not reveal their secret rating formula.

2 Learning-to-Rank and Learning-To-Rerank

The training data are labeled observations $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with observations $\mathbf{x}_i \in X \subset \mathcal{R}^d$ and labels $y_i \in \{0, 1\}$ for all i. The observations labeled "1" are called "positive observations," and the observations labeled "0" are "negative observations." There are n_+ positive observations and n_- negative observations, with index sets $S_+ = \{i : y_i = 1\}$ and $S_- = \{k : y_k = 0\}$. A ranking algorithm uses the training data to produce a scoring function $f : X \to \mathcal{R}$ that assigns each observation a real-valued score. Ideally, for a set of test observations drawn from the same (unknown) distribution as the training data, f should rank the observations in order of $P(y = 1|x)$, and we measure the quality of the solution using "rank statistics," or functions of the observations relative to each other. Note that bipartite ranking and binary classification are fundamentally different, and there are many works that explain the differences (e.g., Ertekin and Rudin, 2011). Briefly, classification algorithms consider a statistic of the observations relative to a decision boundary (n comparisons) whereas ranking algorithms consider observations relative to each other (on the order of n^2 comparisons for pairwise rank statistics).

Since the evaluation of test observations uses a chosen rank statistic, the same rank statistic (or a convexified version of it) is optimized on the training set to produce f. Regularization is added to help with generalization. Thus, a ranking algorithm looks like:

$$\min_{f \in \mathcal{F}} \ \text{RankStatistic}(f, \{\mathbf{x}_i, y_i\}_i) + C \cdot \text{Regularizer}(f).$$

For efficiency we are considering *reranking* methods, which have two ranking steps. In the first ranking step, a base algorithm is run over the training set, and a scoring function f_{initial} is produced and observations are rank-ordered by the score. A threshold is chosen, and all observations with scores above the threshold are reranked by another ranking algorithm which produces another scoring function f. To evaluate the quality of the solution on the test set, each test observation is evaluated first by f_{initial}. For the observations with scores above the threshold, they are reranked according to f. The full ranking of test observations is produced by appending the test observations scored by f to the test observations scored only by f_{initial}.

Rank Statistics. For the purpose of this section, the rank is currently defined so that the top of the list has the highest ranks, and all ranks are unique so that each observation is assigned to one rank. The rank of an observation is the number of observations with scores at or beneath it: $\text{Rank}(f(\mathbf{x}_i)) = \sum_{t=1}^n \mathbf{1}_{[f(\mathbf{x}_t) \leq f(\mathbf{x}_i)]}$. Thus, ranks can range from 1 at the bottom to n at the top. A **conditional linear rank statistic** (CLRS) created from scoring function $f : X \to \mathcal{R}$ is of the

form

$$\text{CLRS}(f) = \sum_{i=1}^{n} \mathbf{1}_{[y_i=1]} \phi(\text{Rank}(f(\mathbf{x}_i)).$$

Here ϕ is a non-decreasing function producing only non-negative values. We define $a_\ell := \phi(\ell)$, the contribution to the score if the observation with rank ℓ has label +1. By properties of ϕ, we know $0 \leq a_1 \leq a_2 \leq \cdots \leq a_n$. Then

$$\text{CLRS}(f) = \sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} \mathbf{1}_{[\text{Rank}(f(\mathbf{x}_i))=\ell]} \cdot a_\ell. \quad (1)$$

This class captures a broad collection of rank statistics, including the following well-known rank statistics:

- $a_\ell = \ell$: Wilcoxon Rank Sum (WRS) statistic, which is an affine function of the Area Under the Receiver Operator Characteristic Curve (AUC) when there are no ties in rank (that is, f such that $f(\mathbf{x}_i) \neq f(\mathbf{x}_k)\ \forall i \neq k$).

  $$\text{WRS}(f) \quad = \quad n_+ n_- \cdot \text{AUC}(f) + \frac{n_+(n_+ + 1)}{2}.$$

  The AUC is the fraction of correctly ranked positive-negative pairs: $\text{AUC}(f) = \frac{1}{n_+ n_-} \sum_{i \in S_+} \sum_{k \in S_-} \mathbf{1}_{[f(\mathbf{x}_k) < f(\mathbf{x}_i)]}$. The AUC, when multiplied by constant $n_+ n_-$, is the Mann-Whitney U statistic.
- $a_\ell = \ell \cdot \mathbf{1}_{[\ell \geq \theta]}$ for predetermined threshold θ: Related to the local AUC or partial AUC, which looks at the area under the leftmost part of the ROC curve only. The leftmost part of the ROC curve is the top portion of the ranked list.
- $a_\ell = \mathbf{1}_{[\ell=n]}$: Winner Takes All (WTA), which is 1 when the top observation in the list is positively-labeled.
- $a_\ell = \frac{1}{n-\ell+1}$: Mean Reciprocal Rank (MRR) (as used by Burges et al, 2006).
- $a_\ell = \frac{1}{\log_2(n-\ell+2)}$: Discounted Cumulative Gain (DCG), which is used as a quality measure in information retrieval (Järvelin and Kekäläinen, 2000).
- $a_\ell = \frac{1}{\log_2(n-\ell+2)} \cdot \mathbf{1}_{[\ell \geq N]}$: DCG@N, which cuts off the DCG after the top N.
- $a_\ell = \ell^p$ for some $p > 0$: Similar to the P-Norm Push, which uses ℓ_p norms to focus on the top of the list, the same way as an ℓ_p norm focuses on the largest elements of a vector (Rudin, 2009a).

Rank statistics have been studied in several theoretical papers (e.g., Wang et al, 2013), and there are many works that discuss how to approximately and rapidly solve ranking and reranking problems on the large scale (Freund et al, 2003; Tsochantaridis et al, 2005; Joachims, 2002, 2006; Cossock and Zhang, 2006; Burges et al, 2006; Chakrabarti et al, 2008; Xu et al, 2008; Le et al, 2010; Qin et al, 2013; Ferri et al, 2002; Qin et al, 2013; Ataman et al, 2006; Collins and Koo, 2005; Ji et al, 2006; Rudin, 2009a; Rudin and Schapire, 2009; Ertekin and Rudin, 2011). These works all use heuristic loss functions or other approximations in order to produce solutions more rapidly, possibly at the expense of the quality of the solution, as we will discuss.

3 Why Learning-To-Rank Methods Can Fail

Current methods for learning-to-rank optimize convex proxies for the rank statistics we provided above. For instance, RankBoost (Freund et al, 2003) uses the exponential loss function as an upper bound for the 0-1 loss within the misranking error, $\mathbf{1}_{[z \leq 0]} \leq e^{-z}$, and minimizes $\sum_{i \in S_+} \sum_{k \in S_-} e^{-(f(\mathbf{x}_i) - f(\mathbf{x}_k))}$ as a proxy for maximizing the AUC. There are algorithms that use various other loss functions.

We prove that the exponential loss and other common loss functions may yield poor results for some rank statistics. In particular, the main result in this section is as follows:

Theorem: *There is a simple one-dimensional dataset for which there exist two ranked lists (called Solution 1 and Solution 2) that are completely reversed from each other (the top of one list is the bottom of the other and vice versa) such that the WRS (the AUC), partial AUC@100, DCG, MRR and hinge loss prefer solution 1, whereas the DCG@100, partialAUC@10 and exponential loss all prefer Solution 2.*

The proof is by construction. Along the single dimension x, the dataset has 10 negatives near x=3, then 3000 positives near x=1, then 3000 negatives near x=0, and 80 positives near x=−10. This theorem means that using the exponential loss to approximate the AUC, as RankBoost does, could give the completely opposite result than desired. It also means that using the hinge loss to approximate the partial DCG or partial AUC could yield completely the wrong result. Further, the fact that the exponential loss and hinge loss behave differently also suggests that convex losses can behave quite differently than the underlying rank statistics that they are meant to approximate.

If we were directly to optimize the rank statistic of interest (as we do in this paper), the problem discussed above would vanish.

3.1 Most Learning-To-Rank Methods Have The Problem Discussed Above

The class of CLRS includes a very wide range of rank statistics, some of which concentrate on the top of the list (e.g., DCG) and some that do not (e.g.,WRS), and it is not clear which conditional linear rank statistics (if any) from the CLRS are close to the convexified loss functions of the ranking algorithms. RankBoost is not the only algorithm where problems can occur, and they can also occur for support vector machine ranking algorithms (e.g., Joachims, 2002; Herbrich et al, 2000) and algorithms like RankProp and RankNet (Caruana et al, 1996; Burges et al, 2005). The methods of Ataman et al (2006), Brooks (2010), and Tan et al (2013) have used linear relaxations or greedy methods for learning to rank, rather than exact reranking, which will have similar issues; if one optimizes the wrong rank statistic, one may not achieve the correct answer. Logistic regression is commonly used for ranking. Logistic regression minimizes: $\sum_{i=1}^{n} \ln\left(1 + e^{-y_i f(\mathbf{x}_i)}\right)$. This loss function does not closely resemble AUC. On the other hand, it is surprising how common it is to use logistic regression to produce a predictive model, and yet evaluate the quality of the model using AUC.

The fundamental premise of learning-to-rank is that better test performance can be achieved by optimizing the performance measure (a rank statistic) on the training set. This means that one should choose to optimize differently for each rank statistic. However, in practice when the same convex substitute is used to approximate a variety of rank statistics, it directly undermines this fundamental premise, and could compromise the quality of the solution. If convexified rank statistics are a reasonable substitute for rank statistics, we would expect to see that (i) the rank statistics are reasonably approximated by their convexified versions, (ii) if we consider several convex proxies for the same rank statistic (in this case AUC), then they should all behave very similarly to each other, and similarly to the true (non-convexified) AUC. However, as we discussed, neither of these are true.

3.2 Ties Are Problematic, Thus Use ResolvedRank and Subrank

Dealing with ties in rank is critical when directly optimizing rank statistics. If a tie in rank between a positive and negative is considered as correct, then an optimal learning algorithm would produce the trivial scoring function $f(x) = constant\ \forall x$; this solution would unfortunately attain the highest possible score when optimizing any pairwise rank statistic. (This problem happens with the definitions of Clemençon and Vayatis, 2008).

We need to encourage our ranking algorithm not to produce ties in score, and thus in rank. To do this, we pessimistically consider a tie between and positive and a negative as a misrank. We will use two definitions of rank within the CLRS – *ResolvedRanks* and *Subranks*. For ResolvedRanks, when negatives are tied with positives, we force the negatives to be higher ranked. For Subranks, we do not force this, but when we optimize the CLRS, we will prove that ties are resolved this way anyway. The assignment of ResolvedRanks and Subranks are not unique, there can be multiple ways to assign ResolvedRanks or Subranks for a set of observations.

We define the ***Subrank*** *by the following formula:*

$$\text{Subrank}(f(\mathbf{x}_i)) = \sum_{k=1}^{n} \mathbf{1}_{[f(\mathbf{x}_k) < f(\mathbf{x}_i)]}, \quad \forall i = 1, \ldots, n.$$

The Subrank of observation i is the number of observations that score strictly below it. Subranks range from 0 to $n-1$ and the CLRS becomes:

$$\text{CLRS}_{\text{Subrank}}(f) = \sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} \mathbf{1}_{[\text{Subrank}(f(\mathbf{x}_i))=\ell-1]} \cdot a_\ell.$$

Observations with equal score have tied Subranks. *ResolvedRanks* are defined as follows, where tied ranks are resolved pessimistically.

ResolvedRanks *obey: (i) The ResolvedRank of an observation is greater than or equal to its Subrank. (ii) If a positive observation and a negative observation have the same score, then the negative observation is given a higher ResolvedRank. (iii) Each possible ResolvedRank, 0 through* $n-1$*, is assigned to exactly one observation.*

The SubRanks and ResolvedRanks are equal to each other when there are no ties in score. We then have the CLRS with ResolvedRanks as:

$$\begin{aligned} &\text{CLRS}_{\text{ResolvedRank}}(f) \\ &\quad = \sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} \mathbf{1}_{[\text{ResolvedRank}(f(\mathbf{x}_i))=\ell-1]} \cdot a_\ell. \end{aligned}$$

The ResolvedRanks are the quantity of interest, as optimizing them will provide a scoring function with minimal misranks and minimal ties between positives and negatives.

4 Maximize the Regularized CLRS with ResolvedRanks

In the supplement we produce two formulations – one for optimizing the regularized CLRS with ResolvedRanks, and the other for optimizing the regularized

CLRS with Subranks. Here we describe only the ResolvedRanks formulation, though the Subrank formulation would be used for the reranking step.

We would like to optimize the general CLRS, for any choices of the a_ℓ's, where we want to penalize ties in rank between positives and negatives, and we would also like a full ranking of observations. Thus, we will directly optimize

$$\text{CLRS}_{\text{ResolvedRank}}(f) + C \cdot \text{Regularizer}(f)$$

for our reranking algorithm. Our hypothesis space is the space of linear scoring functions $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$, where $\mathbf{w} \in \mathcal{R}^d$.

$$\max_{\mathbf{w}\in\mathcal{R}^d} \text{CLRS}_{\text{ResolvedRank}}(\mathbf{w}) - C\|\mathbf{w}\|_0$$
$$= \max_{\mathbf{w}\in\mathcal{R}^d} \sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} \mathbf{1}_{[\text{ResolvedRank}(\mathbf{w}^T\mathbf{x}_i))=\ell-1]} \cdot a_\ell - C\|\mathbf{w}\|_0.$$

Nonlinearities can be incorporated as usual by including additional variables, such as indicator variables or nonlinear functions of the original variables. We optimize over choices for vector $\mathbf{w}$. Building up to the formulation, we will create the binary variable $t_{i\ell}$ so that it is 1 for $\ell \leq \text{ResolvedRank}(f(\mathbf{x}_i)) + 1$ and 0 otherwise. That is, if observation i has ResolvedRank equal to 5, then $t_{i1}, ..., t_{i6}$ are all 1 and $t_{i7}, ..., t_{in}$ are 0. Then $\sum_{\ell=1}^{n}(a_\ell - a_{\ell-1})t_{i\ell}$ is a telescoping sum for $\ell \leq$ResolvedRank$(f(\mathbf{x}_i))+1$. When we define $a_0 = 0$, the sum reduces to $a_{\text{ResolvedRank}(f(\mathbf{x}_i))+1}$, or: $\sum_{\ell=1}^{n} \mathbf{1}_{[\text{ResolvedRank}(f(\mathbf{x}_i))=\ell-1]} \cdot a_\ell$. We multiply by y_i and sum to produce the $\text{CLRS}_{\text{ResolvedRank}}$. Doing this, $\text{CLRS}_{\text{ResolvedRank}}$ becomes:

$$\sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} (a_\ell - a_{\ell-1})t_{i\ell} \quad \text{where } a_0 = 0.$$

By definition $t_{i1} = 1$ for all i, so we can simplify the $\text{CLRS}_{\text{ResolvedRank}}$ function above to:

$$\sum_{i\in S_+} \left(\sum_{\ell=2}^{n} (a_\ell - a_{\ell-1})t_{i\ell} + a_1 \right)$$
$$= |S_+|a_1 + \sum_{i\in S_+} \sum_{\ell=2}^{n} (a_\ell - a_{\ell-1})t_{i\ell}.$$

Note that the differences $a_\ell - a_{\ell-1}$ are all nonnegative. Only when they are strictly positive is a contribution made to the $\text{CLRS}_{\text{ResolvedRank}}$ function. Thus, we introduce notation $\widetilde{a}_\ell = a_\ell - a_{\ell-1}$ and $S_{\text{r}} = \{\ell \geq 2 : \widetilde{a} > 0\}$. The $\text{CLRS}_{\text{ResolvedRank}}$ becomes:

$$|S_+|a_1 + \sum_{i\in S_+} \sum_{\ell\in S_{\text{r}}} \widetilde{a}_\ell t_{i\ell}. \tag{2}$$

We will maximize this, which means that the $t_{i\ell}$'s will be set to 1 when possible, because the $\widetilde{a}_\ell$'s in the sum are all positive. When we maximize, the constant $|S_+|a_1$ term can be omitted. We define integer variables $r_i \in [0, n-1]$ to represent the ResolvedRanks of the observations. Variables r_i and $t_{i\ell}$ are related in that $t_{i\ell}$ can only be 1 when $\ell \leq r_i + 1$, implying $t_{i\ell} \leq \frac{r_i}{\ell-1}$. We use linear scoring functions, so the score of instance $\mathbf{x}_i$ is $\mathbf{w}^T\mathbf{x}_i$. Variables z_{ik} are indicators of whether the score of observation i is above the score of observation k. Thus we want to have $z_{ik} = 1$ if $\mathbf{w}^T\mathbf{x}_i > \mathbf{w}^T\mathbf{x}_k$ and $z_{ik} = 0$ otherwise. Beyond this we want to ensure no ties in score, so we want all scores to be at least ε apart. Thus we have derived our first ranking algorithm, which maximizes the regularized CLRS using ResolvedRanks.

$$\underset{\mathbf{w},\gamma_j,z_{ik},t_{i\ell},r_i\forall i,k,\ell,j}{\text{argmax}} \quad \sum_{i\in S_+} \sum_{\ell\in S_{\text{r}}} \widetilde{a}_l t_{i\ell} - C\sum_j \gamma_j \quad \text{s.t(3)}$$

$$z_{ik} \leq \mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_k) + 1 - \varepsilon, \quad \forall i,k = 1,\ldots,n, \tag{4}$$
$$z_{ik} \geq \mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_k), \quad \forall i,k = 1,\ldots,n, \tag{5}$$
$$\gamma_j \geq w_j \tag{6}$$
$$\gamma_j \geq -w_j \tag{7}$$
$$r_i - r_k \geq 1 + n(z_{ik} - 1), \quad \forall i,k = 1,\ldots,n, \tag{8}$$
$$r_k - r_i \geq 1 - nz_{ik}, \quad \forall i \in S_+, k \in S_-, \tag{9}$$
$$r_k - r_i \geq 1 - nz_{ik}, \quad \forall i,k \in S_+, i < k, \tag{10}$$
$$r_k - r_i \geq 1 - nz_{ik}, \quad \forall i,k \in S_-, i < k, \tag{11}$$
$$t_{i\ell} \leq \frac{r_i}{\ell - 1}, \quad \forall i \in S_+, \ell \in S_r, \tag{12}$$
$$-1 \leq w_j \leq 1, \quad \forall j = 1,\ldots,d,$$
$$0 \leq r_i \leq n-1, \quad \forall i = 1,\ldots,n, \tag{13}$$
$$z_{ik} \in \{0,1\}\ \forall i,k = 1,\ldots,n,$$
$$t_{i\ell} \in \{0,1\}\ \forall i \in S_+, \ell \in S_r, \tag{14}$$
$$\gamma_j \in \{0,1\} \forall j \in \{1,...d\}. \tag{15}$$

To ensure that solutions with ranks that are close together are not feasible, Constraint (4) forces $z_{ik} = 0$ if $\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_k < \varepsilon$, and Constraint (5) forces $z_{ik} = 1$ if $\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_k > 0$. Thus, a solution where any two observations have a score difference above 0 and less than ε is not feasible. (Note that these constraints alone do not prevent a score difference of exactly 0; for that we need the constraints that follow.) Constraints (6) and (7) define the γ_j's to be indicators of nonzero coefficients w_j. Constraints (8)-(11) are the "tie resolution" equations. Constraint (8) says that for any pair $(\mathbf{x}_i, \mathbf{x}_k)$, if the score of i is larger than that of k so that $z_{ik} = 1$, then $r_i \geq r_k + 1$. That handles the assignment of ranks when there are no ties, so now we need only to resolve ties in the score. We have Constraint (9) that applies to positive-negative pairs: when the pair is tied, this constraint forces the negative observation to have higher rank. Similarly, Constraints (10) and (11) apply to positive-positive pairs and negative-negative pairs respectively, and state that ties are broken lexi-

cographically, that is, according to their index in the dataset. We discussed Constraint (12) earlier, which provides the definition of $t_{i\ell}$ so that $t_{i\ell} = 1$ whenever $\ell \leq r_i + 1$. Also we force the w_j's to be between -1 and 1 so its values do not go to infinity and so that the ε values are meaningful, in that they can be considered relative to the maximum possible range of w_j. The novelty of this formulation is that it is *linear* in all variables, which is a challenge for a problem with this level of complexity.

The formulation for the Subrank optimization problem $\max_{\mathbf{w}\in\mathcal{R}^d} \text{CLRS}_{\text{Subrank}}(\mathbf{w}) - C\|\mathbf{w}\|_0$ is provided in the supplement. Maximizing the Subrank problem is much easier, since we do not want to force a unique assignment of ranks. This means the "tie resolution" equations are no longer present. We can directly assign a Subrank for observation i by $r_i = \sum_{k=1}^{n} z_{ik}$ because it is exactly the count of observations ranked beneath observation i; that way the r_i variables do not even need to appear in the formulation.

5 Why Subranks Are Often Sufficient.

We would ultimately like to get away with solving the Subrank problem rather than the ResolvedRank problem. As we will show, this is generally possible.

The ResolvedRank formulation above has $2d + n^2 + n_+|S_r| + n$ variables, which is the total number of w, γ, z, t, and r variables. The Subrank formulation on the other hand has only $2d + n_+ n + n_+|S_r|$ variables, since we only have w, γ, z, and t. Denote the objectives as follows, where we have $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$.

$$G_{\text{RR}}(f) = \sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} \mathbf{1}_{[\text{ResolvedRank}(f(\mathbf{x}_i))=\ell-1]} \cdot a_\ell - C\|\mathbf{w}\|_0$$

$$G_{\text{Sub}}(f) = \sum_{i=1}^{n} y_i \sum_{\ell=1}^{n} \mathbf{1}_{[\text{Subrank}(f(\mathbf{x}_i))=\ell-1]} \cdot a_\ell - C\|\mathbf{w}\|_0.$$

The following lemma contains a basic relationship:

Lemma 1 *$G_{\text{Sub}}(f) \leq G_{\text{RR}}(f)$ for all f. Further, $G_{\text{Sub}}(f) = G_{\text{RR}}(f)$ for all f with no ties.*

We will show that any maximizer of G_{Sub} also maximizes G_{RR}. This is true under a general condition, which is that there are no exactly duplicated observations. In the first part of the proof, we consider whether there are maximizers of G_{RR} that have no ties in score, that is, solutions $\mathbf{w}$ where $f(\mathbf{x}_i) \neq f(\mathbf{x}_k)$ for any two observations i and k. Assuming such solutions exist, we then show in Theorem 1 that any maximizer of G_{Sub} is also a maximizer of G_{RR}.

Theorem 1 *Assume that the set $\text{argmax}_f G_{\text{RR}}(f)$ contains at least one function $\overline{f}$ having no ties in score. Then any f^* such that $f^* \in \text{argmax}_f G_{\text{Sub}}(f)$ also obeys $f^* \in \text{argmax}_f G_{\text{RR}}(f)$.*

Thus, if there are no ties, a maximizer of G_{Sub} is also a maximizer of G_{RR}. One of the main steps in the proof of this theorem is:

Lemma 2 *If we are given $\overline{f} \in \text{argmax}_f G_{\text{RR}}(f)$ that yields a scoring function $\overline{f}(\mathbf{x}) = \overline{\mathbf{w}}^T\mathbf{x}$ with ties, it is possible to construct a perturbed scoring function $\widehat{f}$ that: (i) preserves all pairwise orderings, $\overline{f}(\mathbf{x}_i) > \overline{f}(\mathbf{x}_k) \Rightarrow \widehat{f}(\mathbf{x}_i) > \widehat{f}(\mathbf{x}_k)$, (ii) has no ties, $\widehat{f}(\mathbf{x}_i) \neq \widehat{f}(\mathbf{x}_k)$ for all i, k. (iii) has $\|\overline{\mathbf{w}}\|_0 = \|\widehat{\mathbf{w}}\|_0$. This result holds whenever no observations are duplicates of each other, $\mathbf{x}_i \neq \mathbf{x}_k \ \forall i, k$.*

Here is the main result about the Subrank and ResolvedRank. It says that if there are no duplicated observations, then there are no ties, and thus a maximizer of G_{Sub} is also a maximizer of G_{RR}.

Theorem 2 *Given $f^* \in \text{argmax}_f G_{\text{Sub}}(f)$, then $f^* \in \text{argmax}_f G_{\text{RR}}(f)$. This holds when there are no duplicated observations, $x_i \neq x_k \ \forall i, k$ where $i \neq k$.*

The result in Theorem 2 shows why optimizing G_{Sub} is sufficient to obtain the maximizer of G_{RR} to use in the reranking algorithm. These theoretical results show why the Subrank formulation is sufficient to solve the ResolvedRank formulation. The proofs of each theorem requires several lemmas proven over several pages, and are thus relegated to the supplement.

6 Empirical Discussion of Learning-To-Rank.

Table 1 illustrates that there are many datasets where reranking using the Subrank formulation can substantially improve the quality of the solution. The datasets we used were: ROC Flexibility (which was designed to show differences in rank statistics, available from Rudin, 2009b); Abalone19 (growth of sea creatures, Alcalá-Fdez et al, 2011); UIS from the UMass AIDS Research Unit, which contains patient information including drug use history, and the label represents whether the patient remained drug free for 12 months afterwards (Hosmer et al, 2013); Travel, which is from a transportation survey between 3 Australian cities (Hosmer et al, 2013); NHANES, which considers prediction of obesity (Hosmer et al, 2013); and Gaussians, which is similar to the dataset discussed earlier about why ranking methods can fail. We present comparative results with several baseline ranking meth-

ods, namely Logistic Regression (LR), Support Vector Machines (SVM), RankBoost (RB), and the P-Norm Push for $p = 2$ and for the Subrank MIP formulations at 4 different levels of the cutoff K for reranking. For the SVM, we tried regularization parameters 10^{-1}, 10^{-2}, ..., 10^{-6} and reported the best. We chose datasets with the right level of imbalance so that not all of the top observations belong to a single class; this ensures the rank statistics are meaningful at the top of the list. For the MIP-based methods, we used logistic regression as the base ranker, and the reranker was learned from the top K. We varied K between 50, 100, 150, and we also used the full list. An exception is made for the Abalone19 data set, for which K varies between 250, 500 and 750 instead because Abalone19 is a highly imbalanced data set. We stopped the computation after 2 hours for each trial (1 hour for the ROC flexibility dataset), which gives a higher chance for the lower-K rerankers to solve to optimality. Most of the K=50 experiments for the ROC flexibility dataset solved to optimality within 5 minutes. The reported means and standard deviations were computed over 10 randomly chosen training and test splits. We chose to evaluate according to the DCG measure as it is used heavily in information retrieval applications. Table 1 shows the results of our experiments, where we highlighted the best algorithm for each dataset on both training and test in bold, and used italics to represent test set results that are not statistically significantly worse than the best algorithm according to a matched pairs t-test. The smaller K models performed consistently well on these data, achieving the best test performance on all of these datasets. On some datasets, we see a $\sim$10% average performance improvement from reranking; on the Travel dataset, the K=50 reranking model had superior results over all of the baselines uniformly across all 10 trials.

Thus our first observation is: *Observation 1: There are some datasets where reranking can substantially improve the quality of the solution.*

The supplement has several other experiments, illustrating the following observations.

Observation 2: There is a tradeoff between computation and quality of solution. Specifically, if the number of elements to rerank (denoted by K) is too small, the solution will not generalize as well, and if the number of elements K is too large, we will not be able to sufficiently solve the reranking problem within the allotted time, and the solution again could suffer.

Observation 3: There are some datasets for which the variance of the result is larger than the differences in the rank statistics themselves. These are cases where better relative training values do not necessarily lead to better relative test values. In these cases we do not think it is worthwhile to use ranking algorithms at all, let alone reranking algorithms. For these datasets, logistic regression may suffice. The cases where reranking/ranking makes a difference are cases where the variance of the training and test values are small enough that we can reliably distinguish between the different rank statistics.

Observation 4: As long as the margin parameter ε is sufficiently small without being too small so that the solver will not recognize it, the quality of the solution is maintained. The regularization parameter C also can have an influence on the quality of the solution, and it is useful not to over-regularize or under-regularize. Our experiments show that if ε is too large, the solver will not be able to force all of the inequalities to be strictly satisfied with margin ε. This could force many good solutions to be considered infeasible and this may ruin the quality of the solution. It could also cause problems with convergence of the optimization problem. When ε is smaller, it increases the size of the feasible solution space, so the problem is easier to solve. On the other hand, if ε is too small, the solver will have trouble recognizing the inequality and may have numerical problems.

Observation 5: Proving optimality takes longer than finding a reasonable solution. Usually a good solution is found within a few minutes, which is an appropriate timescale for maintenance applications.

7 Conclusion

We discussed why current methods for ranking can fail, and presented new approaches to mitigate these serious issues. We proved an analytical reduction from the problem we want to solve (the ResolvedRank formulation) to a much more tractable problem (the Subrank formulation). We created mixed integer *linear* programs that are solvable for many practical problems (e.g., maintenance prioritization), and showed promising empirical results for reranking.

References

Alcalá-Fdez J, Fernandez A, Luengo J, J Derrac SG, Sánchez L, Herrera F (2011) KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing 17(2–3):255–287

Ataman K, Street WN, Zhang Y (2006) Learning to rank by maximizing AUC with linear programming. In: Proc. International Joint Conference on Neural Networks (IEEE IJCNN)

Brooks JP (2010) Support vector machines with the ramp loss and the hard margin loss. Operations Research 59(2):467 – 479

Table 1: Datasets for which reranking can make a difference

Dataset		Baseline methods				MIP-based methods			
		LR	SVM	RB	P-norm Push	$K = 50$	$K = 100$	$K = 150$	Full List
ROC	train	31.21 ± 1.65	30.94 ± 1.57	29.00 ± 1.39	31.33 ± 31.43	$\mathbf{31.96 \pm 1.32}$	31.84 ± 1.36	31.65 ± 1.12	28.43 ± 1.80
	test	31.35 ± 1.48	31.10 ± 1.63	29.57 ± 1.43	31.43 ± 1.61	$\mathbf{32.16 \pm 1.31}$	*32.09 ± 1.31*	*31.74 ± 1.65*	28.96 ± 2.40
Abalone19 [a]	train	3.63 ± 0.43	3.41 ± 0.47	3.40 ± 0.65	3.44 ± 0.47	$\mathbf{4.89 \pm 0.58}$	4.45 ± 0.50	4.13 ± 0.53	2.54 ± 0.35
	test	2.96 ± 0.42	3.02 ± 0.53	2.66 ± 0.49	3.03 ± 0.50	$\mathbf{3.08 \pm 0.49}$	*2.89 ± 0.35*	*2.76 ± 0.38*	2.42 ± 0.48
UIS	train	18.86 ± 1.32	18.46 ± 1.38	19.44 ± 1.44	18.78 ± 1.40	$\mathbf{20.45 \pm 1.23}$	19.76 ± 1.27	19.26 ± 1.05	18.84 ± 1.44
	test	17.88 ± 1.11	17.81 ± 1.21	17.70 ± 1.40	*17.89 ± 1.13*	*18.00 ± 1.31*	$\mathbf{18.64 \pm 1.51}$	*17.79 ± 1.73*	*17.89 ± 0.67*
Travel	train	28.16 ± 1.60	27.59 ± 1.61	26.57 ± 1.60	28.09 ± 1.62	$\mathbf{28.30 \pm 1.63}$	28.24 ± 1.56	27.12 ± 1.45	26.94 ± 1.36
	test	27.32 ± 1.70	26.81 ± 1.76	24.95 ± 1.63	27.24 ± 1.66	$\mathbf{27.61 \pm 1.70}$	*27.39 ± 1.60*	26.00 ± 2.26	26.31 ± 1.83
NHANES	train	14.69 ± 1.63	13.83 ± 1.87	13.75 ± 2.01	14.46 ± 1.57	$\mathbf{15.48 \pm 1.69}$	15.02 ± 1.93	14.73 ± 1.59	13.87 ± 1.25
	test	*13.06 ± 1.74*	*12.98 ± 1.79*	12.10 ± 1.75	*13.18 ± 1.82*	$\mathbf{13.26 \pm 1.50}$	*12.71 ± 1.61*	*12.94 ± 1.55*	*13.09 ± 1.82*
Pima	train	35.50 ± 1.66	35.30 ± 1.61	$\mathbf{35.80 \pm 1.44}$	35.64 ± 1.67	35.75 ± 1.67	35.43 ± 1.69	34.85 ± 1.96	34.77 ± 2.03
	test	*34.18 ± 1.81*	*34.03 ± 1.83*	33.83 ± 1.65	*34.24 ± 1.82*	$\mathbf{34.44 \pm 1.76}$	33.56 ± 1.87	33.72 ± 2.21	*33.65 ± 2.01*
Gaussians	train	69.25 ± 2.70	69.28 ± 2.70	71.31 ± 2.15	69.24 ± 2.70	71.71 ± 2.22	$\mathbf{71.76 \pm 2.18}$	71.58 ± 2.29	64.70 ± 2.53
	test	64.69 ± 2.45	64.73 ± 2.45	67.13 ± 2.06	64.65 ± 2.43	$\mathbf{68.03 \pm 2.27}$	*67.91 ± 2.27*	*67.79 ± 2.30*	59.89 ± 2.26

[a] We use $K = 250$, $K = 500$ and $K = 750$ for this data set because it is highly imbalanced.

Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: Proc. 22nd International Conference on Machine Learning (ICML)

Burges CJ, Ragno R, Le QV (2006) Learning to rank with nonsmooth cost functions. In: Proc. Advances in Neural Information Processing Systems (NIPS), pp 395–402

Caruana R, Baluja S, Mitchell T (1996) Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In: Advances in Neural Information Processing Systems (NIPS), vol 8, pp 959–965

Chakrabarti S, Khanna R, Sawant U, Bhattacharyya C (2008) Structured learning for non-smooth ranking losses. In: Proc. 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), pp 88–96

Chang A, Rudin C, Bertsimas D (2010) A discrete optimization approach to supervised ranking. In: Proc. INFORMS 5th Annual Workshop on Data Mining and Health Informatics

Chang A, Rudin C, Bersimas D (2011) Integer optimization methods for supervised ranking. Operations Research Center Working Paper Series OR 388-11, MIT

Chang A, Rudin C, Cavaretta M, Thomas R, Chou G (2012) How to reverse-engineer quality rankings. Machine Learning 88:369–398

Clemençon S, Vayatis N (2008) Empirical performance maximization for linear rank statistics. Proc Advances in Neural Information Processing Systems (NIPS) 21

Collins M, Koo T (2005) Discriminative reranking for natural language parsing. Journal of Association for Computational Linguistics 31(1):25–70

Cossock D, Zhang T (2006) Subset ranking using regression. In: Conference on Learning Theory (COLT), pp 605–619

Ertekin Ş, Rudin C (2011) On equivalence relationships between classification and ranking algorithms. Journal of Machine Learning Research 12:2905–2929

Ferri C, Flach P, Hernández-Orallo J (2002) Learning decision trees using the area under the ROC curve. In: Proc. 19th International Conference on Machine Learning (ICML), Morgan Kaufmann, pp 139–146

Freund Y, Iyer R, Schapire RE, Singer Y (2003) An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research 4:933–969

Herbrich R, Graepel T, Obermayer K (2000) Large margin rank boundaries for ordinal regression. Advances in Large Margin Classifiers

Hosmer DW, Lemeshow S, Sturdivant RX (2013) Applied Logistic Regression: Third Edition. John Wiley & Sons Inc.

Jain V, Varma M (2011) Learning to re-rank: Query-dependent image re-ranking using click data. In: Proc. 20th International Conference on World Wide Web (WWW), pp 277–286

Järvelin K, Kekäläinen J (2000) IR evaluation methods for retrieving highly relevant documents. In: Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 41–48

Ji H, Rudin C, Grishman R (2006) Re-ranking algorithms for name tagging. In: Proc. HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing, pp 49–56

Joachims T (2002) Optimizing search engines using clickthrough data. In: Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)

Joachims T (2006) Training linear svms in linear time. In: Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 217–226

Kang C, Wang X, Chen J, Liao C, Chang Y, Tseng B, Zheng Z (2011) Learning to re-rank web search results with multiple pairwise features. In: Proc. Fourth International Conference on Web Search and Web Data Mining (WSDM)

Kessler JS, Nicolov N (2009) Targeting sentiment expressions through supervised ranking of linguistic configurations. In: Proceedings of the Third International Conference on Weblogs and Social Media (ICWSM)

Kong D, Saar-Tsechansky M (2013) Collaborative information acquisition for data-driven decisions. Machine Learning, Special Issue on ML for Science and Society

Le QV, Smola A, Chapelle O, Teo CH (2010) Optimization of ranking measures. Journal of Machine Learning Research pp 1–48

Li Z, Zhang MB, Wang Y, Chen F, Whiffin V, Taib R, Vicky W, Wang Y (2013) Water pipe condition assessment: A hierarchical beta process approach for sparse incident data. Machine Learning, Special Issue on ML for Science and Society

Menon AK, Jiang X, Kim J, Vaidya J, Ohno-Machado L (2013) Detecting inappropriate access to electronic health records using collaborative filtering. Machine Learning, Special Issue on ML for Science and Society

Oza N, Castle JP, Stutz J (2009) Classification of aeronautics system health and safety documents. IEEE Transactions on Systems, Man and Cybernetics, Part C 39:1–11

Potash E, Brew J, Loewi A, Majumdar S, Reece A, Walsh J, Rozier E, Jorgenson E, Mansour R, Ghani R (2015) Predictive modeling for public health: Preventing childhood lead poisoning. In: Proc. 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)

Qin T, Liu T, Li H (2013) A general approximation framework for direct optimization of information retrieval measures. Information Retrieval 13(4):375–397

Rudin C (2009a) The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list. Journal of Machine Learning Research 10:2233–2271

Rudin C (2009b) ROC Flexibility Data. `https://users.cs.duke.edu/~cynthia/code/ROCFlexibilityData.html`

Rudin C, Schapire RE (2009) Margin-based ranking and an equivalence between AdaBoost and RankBoost. Journal of Machine Learning Research 10:2193–2232

Rudin C, Passonneau R, Radeva A, Dutta H, Ierome S, Isaac D (2010) A process for predicting manhole events in Manhattan. Machine Learning 80:1–31

Rudin C, Waltz D, Anderson RN, Boulanger A, Salleb-Aouissi A, Chow M, Dutta H, Gross P, Huang B, Ierome S, Isaac D, Kressner A, Passonneau RJ, Radeva A, Wu L (2012) Machine learning for the New York City power grid. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(2):328–345

Tan M, Xia T, Guo L, Wang S (2013) Direct optimization of ranking measures for learning to rank models. In: Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 856–864

Tsochantaridis I, Joachims T, Hofmann T, Altun Y, Singer Y (2005) Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6:1453–1484

Wang Y, Wang L, Li Y, He D, Liu TY (2013) A theoretical analysis of ndcg type ranking measures. In: Proc. 26th Annual Conference on Learning Theory (COLT), PMLR, vol 30, pp 25–54

Xu J, Liu TY, Lu M, Li H, Ma WY (2008) Directly optimizing evaluation measures in learning to rank. In: Proc. 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval