*Proceedings of the 5th INFORMS Workshop on Data Mining and Health Informatics (DM-HI 2010)*
*D. Sundaramoorthi, M. Lavieri, H. Zhao, eds.*

# A DISCRETE OPTIMIZATION APPROACH TO SUPERVISED RANKING

**Dimitris Bertsimas** **Allison Chang** **Cynthia Rudin**

**Operations Research Center**
**Massachusetts Institute of Technology**
**Cambridge, MA**
**aachang@mit.edu**

## Abstract

In many ranking tasks in machine learning, the goal is to construct a scoring function $f: X \rightarrow R$, with $X$ in $R^d$, that can be used to rank a set of labeled examples $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in X$ and $y_i \in \{0,1\}$, that are randomly drawn from an unknown distribution on $X \times \{0,1\}$. We present a mixed integer programming (MIP) method to generate this scoring function. In particular, the scoring function is chosen to be a linear combination of features, i.e., $f(x_i) = w^T x_i$, such that the coefficients $w$ are optimal with respect to a ranking quality measure (area under the curve, discounted cumulative gain, $P$-Norm Push, etc.) on a training set. Other methods for ranking approximate the ranking quality measure by a convex function, whereas our MIP approach is exact. As a result, the proposed approach provides a benchmark against which other methods can be judged. We use datasets from various applications, and show that this novel method performs well on both training and test data, compared with traditional machine learning techniques.

**Keywords**: Discrete optimization, supervised ranking, rank statistics, machine learning.

## 1 Introduction

Supervised ranking tasks arise in many domains, such as information retrieval [8], recommender systems [2], natural language processing [9], bioinformatics [1], and industrial prioritization [11], among others. For many applications, improving the quality of a ranked list by even a small amount can have significant consequences. For instance, the accurate prioritization, or ranking, of manholes on the electricity grid in Manhattan, in order of vulnerability to dangerous events, helps prevent serious events such as fires and explosions [11]. In the area of drug screening, where developing a new drug costs over $1 billion, the ability to correctly rank the top of a list of millions of compounds according to the chance of clinical success can produce enormous savings, in terms of both time and money [1]. For Netflix, the accurate ranking of movies can translate into increased user satisfaction and millions of dollars in increased profit [2].

This paper focuses on a new mixed integer programming (MIP) technique for ranking tasks in machine learning. The primary advantage of using MIP for ranking is that it allows for direct optimization of the true objective function rather than approximating with a heuristic choice of loss functions. This means that the cost function that we optimize with MIP is also the measure that we use to evaluate ranking quality, while other methods use heuristic loss functions that differ from the evaluation measure. All other current state-of-the-art machine learning methods make such a heuristic choice in order to obtain solutions faster. However, we will show that this heuristic choice is often made at the expense of finding better solutions. There are many applications, such as those mentioned above, for which the extra computation time needed to compute a better solution is worthwhile.

Since our methods compute exact, rather than approximate, solutions, they serve as a benchmark by which other algorithms should be compared. Moreover, standard technology for solving MIP problems can quantify a feasible solution's closeness to optimality in case the problem is too large to solve, in contrast to heuristic methods that cannot measure closeness to optimality. We demonstrate experimentally that our exact methods yield better solutions than other approaches. In addition, we show that the linear relaxations of the MIP formulations may be used to generate good approximate solutions to the ranking problems as well.

In the classical statistics literature, summary statistics of receiver operating characteristic (ROC) curves are used to measure the performance of a test in distinguishing between two populations [4]. In supervised ranking, statistics of ROC curves, that is rank statistics, are used for a different purpose, namely to measure the performance of a model on a sample. Supervised ranking algorithms optimize precisely these rank statistics over the sample to generate a model. Common rank statistics include the area under the ROC curve (AUC) [4] and the discounted cumulative gain (DCG) measure used in information retrieval [8]. In this paper, we present two MIP formulations for supervised ranking: one that optimizes the AUC and another that optimizes a general class of rank statistics that includes the DCG. Our MIP methods achieve higher ranking quality than current state-of-the-art algorithms for a wide range of supervised ranking problems.

MIP methods are not commonly used to solve machine learning problems, partly due to a perception starting from the early 1970s that they are computationally intractable for most real-world problems [3]. However, major advances within the last decade in computing power and algorithms for solving MIP problems have made larger scale computations possible, and modern solvers will continue to implement methodologies that take advantage of breakthrough developments in both computer architectures and the mathematical theory behind MIP.

In Section 2, we describe our notation, specifically for supervised bipartite ranking tasks, and also explain the MIP formulations. In Section 3, we show computational results comparing the performance of MIP to that of several other methods. Finally, we conclude with a summary and list of future steps in Section 4.

## 2 Supervised Bipartite Ranking and MIP

In this section, we introduce the framework of supervised bipartite ranking and propose a new way of capturing and optimizing a general class of rank statistics. We also explain our means of handling ties, which is significant in the ability of our methods to optimize rank statistics.

### 2.1 Notation

The data consist of labeled examples $\{(x_i, y_i)\}_{i=1}^n$, with $x_i \in X$ in $R^d$ and $y_i \in \{0,1\}$. Examples labeled "1" are "positive," and examples labeled "0" are "negative." There are $n_+$ positive and $n_-$ negative examples. We define index sets $S_+ = \{i: y_i = 1\}$ and $S_- = \{k: y_k = 0\}$. To rank the examples, we construct a scoring function $f: X \to R$ to assign real-valued scores $\{f(x_i)\}_{i=1}^n$. We define rank and relative rank as functions of $f$ by the following formulas respectively:

$$\text{rank}_f(x_i) = \sum_{k=1}^{n} \mathbf{1}_{[f(x_k)<f(x_i)]} \quad \forall i = 1, \dots, n, \tag{1}$$

$$\text{relrank}_f(x_i) = \sum_{k\in S_-} \mathbf{1}_{[f(x_k)<f(x_i)]} \quad \forall i \in S_+. \tag{2}$$

That is, the rank of any example is the number of examples scored strictly below it, and the relative rank of a positive example is its rank relative to only the negative examples, i.e., the number of negative examples scored strictly below it. A *misrank* occurs when a negative is scored at least as high as a positive. We will use linear scoring functions $f(x_i) = w^T x_i$, where $w \in R^d$. Then the supervised bipartite ranking problem is to generate a function $f$ such that the coefficients $w_1, \dots, w_d$ are optimal with respect to a specified ranking quality measure.

### 2.2 Rank Statistics

The most popular quality metric for ranked lists is the AUC, given by (counting ties as misranks):

$$\text{AUC}(f) = \frac{1}{n_+ n_-} \sum_{i\in S_+} \text{relrank}_f(x_i) = \frac{1}{n_+ n_-} \sum_{i\in S_+} \sum_{k\in S_-} \mathbf{1}_{[f(x_k)<f(x_i)]}.$$

Two related statistics are the *misranking error* and *Wilcoxon-Mann-Whitney* (*WMW*) *U* statistic:

$$\text{ERR}(f) = 1 - \text{AUC}(f), \quad \text{WMW}(f) = n_+ n_- \cdot \text{AUC}(f).$$

Here we define a general class of linear rank statistics:

**Definition 1. (Rank Risk Functional)** Let $a_1 \le a_2 \le \dots \le a_n$ be non-negative constants. A rank risk functional (RRF) is of the form:

$$\text{RRF}(f) = \sum_{i=1}^{n} y_i \sum_{l=1}^{n} \mathbf{1}_{[\text{rank}_f(x_i)=l-1]} \cdot a_l.$$

This class is an extension of the class of conditional linear rank statistics [5]. The RRF equation coincides with the usual definition of these rank statistics when there are no ties. Special members of this class include the following (see [5] and [10]):

- $a_l = l$: Wilcoxon Rank Sum (WRS) – differs from WMW by a constant if there are no ties.
- $a_l = l \cdot \mathbf{1}_{[l\ge t]}$ for some fixed threshold *t*: partial WMW – concentrates at the top of the list.
- $a_l = \mathbf{1}_{[l=n]}$: Winner Takes All (WTA) – concerned only with the top example being positive.
- $a_l = \frac{1}{n-l+1}$: Mean Reciprocal Rank (MRR).
- $a_l = \frac{1}{\log_2(n-l+2)}$: DCG – popular in information retrieval.
- $a_l = l^p$ for some $p > 0$: *P*-norm Push – concentrates at the top of the list.

### 2.3 Treatment of Ties

While the treatment of ties in rank is not critical in more classical applications of statistics such as hypothesis testing, it is of central importance in our approach because we want not only to compute rank statistics, but also to *optimize* them. This is precisely why we require a strict inequality in Equations (1) and (2). Consider, for instance, the AUC. Suppose that the relative rank were defined with an inequality instead of strict inequality in Equation (2). If the scoring function were given by $w = 0$, namely $f(x_i) = w^T x_i = 0$ for all $i$, so that all examples were tied with the trivial score of zero, then the AUC would be 1, i.e., the highest possible value. But clearly $w = 0$ is not optimal in any reasonable sense. The strict inequalities in our definitions of rank and relative rank completely prevent this problem. In our formulation, increasing the number of ties lowers ranking quality, which allows us to avoid trivial solutions when optimizing rank statistics.

### 2.4 MIP Formulations

**Formulation 1 (Maximize AUC)**: Let $v_i = w^T x_i$ be the score for example $x_i$. For each pair $(x_i, x_k)$ such that $i \in S_+$ and $k \in S_-$, the binary variable $z_{ik}$ is 1 if $v_i > v_k$, and 0 otherwise. The user specifies a margin $\varepsilon$, so that $z_{ik} = 1$ if $v_i - v_k \geq \varepsilon$. The value of $\varepsilon$ controls a tradeoff between speed of convergence and generalization; smaller values of $\varepsilon$ typically yield shorter runtimes, but the resulting solutions may not perform as well on new data.

$$
\begin{array}{lll}
P_{\text{AUC}}(\varepsilon):\ \max\limits_{w,v,z} & \displaystyle\sum_{i\in S_+}\sum_{k\in S_-} z_{ik} & \\
\text{s.t.} & z_{ik} \leq v_i - v_k + 1 - \varepsilon & \forall i \in S_+, k \in S_- \\
 & v_i = w^T x_i & \forall i \in S_+ \\
 & v_k = w^T x_k & \forall k \in S_- \\
 & -1 \leq w_j \leq 1 & \forall j = 1, \dots, d \\
 & z_{ik} \in \{0,1\} & \forall i \in S_+, k \in S_-.
\end{array}
$$

**Formulation 2 (Maximize RRF):** Let $v_i$ represent the score of example $x_i$. Now we consider all pairs of examples, so for all $i$ and $k$, $z_{ik} = 1$ if $v_i > v_k$ and $\text{rank}_f(x_i) = \sum_{k=1}^n z_{ik}$. The binary variables $t_{il}$ are 1 if $\text{rank}_f(x_i) \geq l - 1$, which implies $t_{il} - t_{i,l+1} = 1$ if $\text{rank}_f(x_i) = l - 1$. Thus the objective is to maximize the following over all possible choices of $t_{il}$'s:

$$\sum_{i=1}^n y_i \sum_{l=1}^n a_l\big(t_{il} - t_{i,l+1}\big),\ \ t_{i,n+1} = 0\ \ \forall i,\ \text{or equivalently}\ \ \sum_{i=1}^n y_i \sum_{l=1}^n (a_l - a_{l-1}) t_{il},\ \ a_0 = 0.$$

Since we have $t_{i1} = 1$ for all $i$, the cost function is:

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

Note that because $a_l - a_{l-1} \geq 0$, the method will set $t_{il} = 1$ whenever possible to maximize the objective. Let $S_2 = \{l \geq 2 : a_l - a_{l-1} > 0\}$. The MIP formulation is:

$$\begin{array}{llll} P_{\mathrm{RRF}}(\varepsilon): & \max\limits_{w,v,z,t} & \sum\limits_{i\in S_+}\sum\limits_{l\in S_2}(a_l - a_{l-1})t_{il} & \\ & \text{s.t.} & v_i = w^T x_i & \forall i = 1,\dots,n \\ & & z_{ik} \le v_i - v_k + 1 - \varepsilon & \forall i \in S_+, k = 1,\dots,n \\ & & t_{il} \le \dfrac{1}{l-1}\sum\limits_{k=1}^{n} z_{ik} & \forall i \in S_+, l \in S_2 \\ & & -1 \le w_j \le 1 & \forall j = 1,\dots,d \\ & & t_{il}, z_{ik} \in \{0,1\} & \forall i \in S_+, l \in S_2, k \in 1,\dots,n \end{array}$$

In our experiments, we will also use the associated linear relaxations of the MIP, in which the binary variables are allowed to take continuous values in [0,1]. In this case, the cost is no longer exactly the AUC or RRF, and the solution $w$ is an approximate solution to the ranking problem.

## 3 Experiments

We solve $P_{\mathrm{AUC}}$ using six datasets – FourClass and SVMGuide1 are from the LIBSVM collection, and all others are from the UCI Machine Learning Repository, except for ROC Flexibility, which is artificially created so that the ROC curves of the individual features differ substantially from each other. For each dataset, we randomly divide the data into training and test sets, and compare the performance of RankBoost (RB) [6], logistic regression (LR), a support vector machine (SVM)-style ranking algorithm, the linear relaxation (LP), and the MIP. This experiment is repeated ten times, using $\varepsilon = 10^{-4}$ to solve all LPs and $\varepsilon = 10^{-6}$ to solve all MIPs except for the ROC Flexibility data, for which we use $\varepsilon = 10^{-5}$ to solve the MIPs.

Table 1 shows the mean training and test AUC over ten instances for each dataset and algorithm; bold indicates the value is not significantly smaller than the highest value in its row at the 0.05 significance level, according to a matched pairs *t*-test. The MIP achieves the statistically highest mean AUC for all training and test sets. Table 2 shows for each dataset the number of times out of ten that each method performs best on the test data; bold indicates the highest count in each row. The counts in the MIP column clearly dominate the counts of the other methods.

Table 1: Problem dimensions and mean AUC (%) on training (top) and test (bottom) sets

| **Dataset** | $\boldsymbol{n_{train}}$ | $\boldsymbol{n_{test}}$ | $\boldsymbol{d}$ | **RB** | **LR** | **SVM** | **LP** | **MIP** |
|---|---|---|---|---|---|---|---|---|
| Liver Disorders | 172 | 173 | 6 | 73.6621 | 74.0374 | 74.2802 | 74.2816 | **75.3802** |
| | | | | **70.6463** | **70.8567** | **70.9705** | **70.9691** | **71.0257** |
| ROC Flexibility | 250 | 250 | 5 | 71.0959 | 72.0945 | 71.1840 | 70.6327 | **80.6163** |
| | | | | 65.9028 | 67.4607 | 67.7844 | 67.1797 | **81.7706** |
| FourClass | 431 | 431 | 2 | 83.0278 | 82.9907 | 83.1853 | 83.1857 | **83.2230** |
| | | | | 82.8790 | 82.8050 | 83.0438 | **83.0492** | **82.9861** |
| SVMGuide1 | 700 | 6389 | 4 | 99.1929 | 99.2255 | 99.2330 | 99.2327 | **99.2384** |
| | | | | **99.0520** | **99.0563** | 99.0649 | **99.0651** | **99.0642** |
| Abalone | 1000 | 3177 | 10 | 91.3733 | 91.4723 | 91.5078 | 91.5067 | **91.5135** |
| | | | | 90.5580 | 90.6139 | **90.6415** | **90.6411** | **90.6419** |
| Magic | 1000 | 18020 | 10 | 84.0105 | 84.0280 | 84.4880 | 84.4903 | **84.4943** |
| | | | | 83.5273 | 83.5984 | **83.9349** | **83.9365** | **83.9370** |

Table 2: Number of times each method performs best on test data

| **Dataset** | **RB** | **LR** | **SVM** | **LP** | **MIP** |
|---|---|---|---|---|---|
| Liver Disorders | 1 | 1 | 3 | 3 | **5** |
| ROC Flexibility | 0 | 0 | 0 | 0 | **10** |
| FourClass | 1 | 1 | 0 | 3 | **5** |
| SVMGuide1 | 1 | 3 | 0 | 2 | **4** |
| Abalone | 1 | 1 | 2 | 2 | **4** |
| Magic | 0 | 0 | 2 | **4** | **4** |

Not all of the MIPs solve to provable optimality within the specified time limit – up to ten hours for the larger datasets – so given more time, it is possible that the MIP solutions could be even better. Tables 1 and 2 both demonstrate that the MIP has a distinct advantage over approximate methods. They also confirm that the LP can be used to generate good solutions.

## 4 Conclusion

We have introduced new powerful mixed integer programming algorithms for a large class of supervised ranking algorithms. In current work, we are devising comprehensive experiments to test our methods using the Gurobi [7] solver. Preliminary evidence suggests that these techniques outperform state-of-the-art techniques on the AUC problem, and we will continue to obtain more computational results for the RRF formulation.

## References

1. Agarwal, S., Dugar, D., and Sengupta, S., 2010, “Ranking Chemical Structures for Drug Discovery: A New Machine Learning Approach,” J. Chem. Inf. Model., 50(5), 716-731.
2. Bennett, J., Lanning, S., 2007, “The Netflix Prize,” Proceedings of KDD Cup.
3. Bertsimas, D., Shioda, R., 2007, “Classification and Regression via Integer Optimization”, Operations Research, 55(2), 252-271.
4. Bradley, A. P., 1997, “The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms,” Pattern Recognition, 30(7), 1145-1159.
5. Clemençon, S., Vayatis, N., 2008, “Empirical Performance Maximization for Linear Rank Statistics”, Advances in Neural Information Processing Systems, 21.
6. Freund, Y., Iyer, R., Schapire, R., Singer, Y., 2003, “An Efficient Boosting Algorithm for Combining Preferences”, JMLR, 4, 933-969.
7. Gurobi Optimization, Inc., 2010, “Gurobi Optimizer Reference Manual, Version 3.0”, http://www.gurobi.com.
8. Järvelin, K., Kekäläinen, J., 2000, “IR Evaluation Methods for Retrieving Highly Relevant Documents”, SIGIR ’00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 41-48.
9. Ji, H., Rudin, C., Grishman, R., 2006, “Re-Ranking Algorithms for Name Tagging”, Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing, 49-56.
10. Rudin, C., 2009, “The P-Norm Push: A Simple Convex Ranking Algorithm that Concentrates at the Top of the List”, JMLR, 10, 2233-2271.
11. 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.