Learning to Rank by Maximizing AUC with Linear Programming

Kaan Ataman, W. Nick Street *Member, IEEE* and Yi Zhang

***Abstract*— Area Under the ROC Curve (AUC) is often used to evaluate ranking performance in binary classification problems. Several researchers have approached AUC optimization by approximating the equivalent Wicoxon-Mann-Whitney (WMW) statistic. We present a linear programming approach similar to 1-norm Support Vector Machines (SVMs) for instance ranking by an approximation to the WMW statistic. Our formulation can be applied to nonlinear problems by using a kernel function. Our ranking algorithm outperforms SVMs in both AUC and classification performance when using RBF kernels, but curiously not with polynomial kernels. We experiment with variations of chunking to handle the quadratic growth of the number of constraints in our formulation.**

I. Introduction

A. Motivation

Many real world classification problems may require an ordering instead of a simple classification. For example in direct mail marketing, the marketer may want to know which potential customers to target for sending new catalogs so as to maximize the revenue. If the number of catalogs the marketer is sending is limited then it is not enough to know who is likely to buy a product after receiving the catalog. In this case it would be more useful to distinguish top n-percent of potential customers who yield the highest likelihood of buying a product. By doing so, the marketer can more intelligently target a subset of her customer base increasing the expected profit. A slightly different example is collaborative book recommendations. To intelligently recommend a book, an algorithm must distinguish the similarities of book preferences of people who have read and submitted a $score$ for the book. Combining these scores it is possible to return a recommendation list to the user. Ideally this list would be ranked in such a way that the book on the top of the list is expected to appeal the most to the user. It is not hard to see that there is a slight difference between those two ranking problems. In the marketing problem the ranking is based on binary input (purchase, non-purchase) whereas in the book recommendation problem the ranking is based on a collection of partial rankings. In this paper we consider the problem of ranking based on binary input.

B. ROC Curves and Area Under the ROC Curve(AUC)

Ranking is a popular machine learning problem that has been addressed extensively in the literature [4, 5, 21]. The most commonly used performance measure to evaluate a classifier's ability to rank instances in binary classification problems is the area under the ROC curve.

ROC curves are widely used for visualization and comparison of performance of binary classifiers. ROC curves were originally used in signal detection theory. They were introduced to the machine learning community by Spackman in 1989, who showed that ROC curves can be used for evaluation and comparison of classification algorithms [20]. The main reason ROC curves became the mainstream performance evaluation tool is the fact that the produced curve is independent of class distribution or underlying misclassification costs [15]. ROC curves plot true positive rate vs. false positive rate by varying the threshold, which is usually the probability of membership to a class, distance to a decision surface or simply a score produced by a decision function. In the ROC space, the upper left corner represents perfect classification while a diagonal line represents random classification. A point in ROC space that lies to the upper left of another point represents a better classifier. Some classifiers such as neural networks or naive Bayes naturally provide those probabilities while other classifiers such as decision trees do not. It is still possible to produce ROC curves for any type of classifier using minor adjustments [7].

Area under the ROC Curve (AUC) is a single scalar value for classifier comparison [2, 9]. Statistically speaking, the AUC of a classifier is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Since AUC is a probability, its value varies between 0 and 1, where 1 represents all positives being ranked higher than all negatives. Larger AUC values indicate better classifier performance across the full range of possible thresholds. Even though it is possible that a classifier with high AUC can be outperformed by a lower AUC classifier at some region of the ROC space, in general the high AUC classifier is better on average when the underlying distribution is not known.

Mainstream classifiers are usually designed to minimize the classification error, therefore they may not perform well when they are applied to ranking problems. Support Vector Machines (SVMs) are exceptions. Even though SVMs are optimized for error minimization, by the nature of their margin maximization property they turn out to be good rankers. It has been shown that accuracy is not always the best performance measure to compare classifier performance, especially for datasets with skewed class or cost distributions as many real world problems are.

The machine learning community has recently explored the question of which performance measure is better in general. Cortes and Mohri investigated the relationships

Kaan Ataman is with the Department of Management Sciences, The University of Iowa, Iowa City, IA 52242, USA (phone: (319)-335-0922; fax: (319) 335-0297; email: kaan-ataman@uiowa.edu).

W. Nick Street is with the Department of Management Sciences, The University of Iowa, Iowa City, IA 52242, USA (phone: (319) 335-1016; fax: (319) 335-0297; email: nick-street@uiowa.edu)

Yi Zhang is with the Department of Management Sciences, The University of Iowa, Iowa City, IA 52242, USA (phone: (319) 353-0972; fax: (319) 335-0297; email: yi-zhang-2@uiowa.edu)

between AUC and accuracy [6]. Ling and Huang showed in their recent study that AUC is a statistically consistent and a more discriminating value than accuracy [12]. Recent research indicates an increase in the popularity of AUC as a performance measure especially in cases where class distributions and/or misclassification costs are unknown.

C. Relations to Wilcoxon-Mann-Whitney Statistic

Assume that in a dataset there are two random variables such that positive points are produced by one of the random variables and negatives are produced by the other. If a pairwise comparison is to be made such that each positive point is compared to each negative point then in this case AUC would be the ratio of the number of correct pairwise orderings vs. the number of all possible pairs. This quantity is called the Wilcoxon-Mann-Whitney (WMW) statistic [13, 22] and given as:

$$W = \frac{\sum_{i=0}^{p-1} \sum_{j=0}^{n-1} I(x_i, y_j)}{pn}$$

$$I(x_i, y_j) = \begin{cases} 1 & \text{if } f(x_i) > f(y_j) \\ 0 & \text{otherwise} \end{cases}$$

where p is the number of positive points, n is the number of negative points, x_i is the i^{th} positive point and y_j is the j^{th} negative point. The function $f(x)$ assigns a score to the point x which is then used for ranking those points.

D. Related Literature

At this point it seems intuitive that an algorithm maximizing the WMW statistic will also maximize the AUC and hence the ranking performance. The WMW statistic itself is not a continuous function, therefore making it difficult to maximize as is. Several papers suggested an approximation approach to the WMW statistic. Both Yan et al. [24] and Herschtal et.al. [10] used a continuous function as an approximation to the WMW statistic. By doing so they were able to use gradient methods to solve the optimization problem. However, an approximation in the case of rank optimization has to be handled carefully otherwise information related to ranking may be lost easily in the process of approximation.

Recent research on instance ranking also utilized scenarios where some kind of user feedback was available on the initial rankings [11, 17]. This type of ranking is usually very important and typical for ranking search engine results, or ranking products where a user can interfere and the algorithm can obtain valuable feedback from the user. This feedback can be in the form of a subject evaluating the initial rankings and reporting mis-ranked points or simply returning a partial ranking where only a subset of points are ranked. Note that in binary classification problems that we will address in this paper, we have no prior or intermediate knowledge of intra-class rankings or we do not obtain any feedback for the correctness of the rankings at any level in our algorithm. The problem of AUC maximization in the context of binary classification is also a very similar problem to ordinal regression [16].

The literature also includes algorithms such as Rankboost by Freund et al. that utilize a multi-stage approach (ensembling) [8]. This algorithm works by combining preferences based on the well-known boosting approach. Also a variation to Rankboost was later introduced by Rudin et al. [19]. Rankboost is considered to be state of the art algorithm, however for the sake of fairness since it is a multi-stage algorithm as opposed to our one shot optimization algorithm we will not compare our results with Rankboost or any other multi-stage methods. We believe that we can further improve the performance of our LP ranker by ensembling it. Only then we plan to compare our ranking performance with Rankboost and other multistage rank optimizing algorithms. This paper we will only compare the performance to that of SVMs. Our algorithm is quite similar to SVMs in formulation except that it is specifically setup to maximize ranking performance as opposed to minimizing the error rate. In literature it is a common practice to rank new examples based on the output of a classifier. Among classifier methods SVMs are considered to be good rankers, therefore we believe that initially comparing our algorithm to SVMs will provide us valuable feedback on the improvement we can get over an already good ranking algorithm by setting the formulation up as a rank optimization problem.

Rank optimizing SVMs have came into focus recently. An effort by Rakotomamonjy to investigate rank optimizing kernels within the context of SVMs led to a formulation that produced comparatively inferior results to regular SVM formulations [18]. However his work led to more investigations in the area. Brefeld et al. introduced a similar kernel formulation which led to a better ranking performance compared to the previous work [3]. In their algorithm they used regular quadratic optimization in the objective with a kernel structure which improved the ranking performance.

From the discussions above we can conclude that the rank optimization problem should minimize the number of pairwise misrankings. We achieve this objective with a formulation that produces scores for each instance where these scores we can be used to rank the data points. The algorithm should be able to handle nonlinear problems so that it can be applied to a wide range of problems. We achieve this with the help of kernel functions. Also the algorithm should be robust to any size dataset. Our formulation introduces two $exact$ speed-up approaches to improve the solving time of the algorithm, however this part is still a work in progress. Our initial results discussed later in this paper are promising.

Next section introduces our LP formulation that implements the properties mentioned above. Then we present our results and discussions and we sum it up with conclusions.

II. The Algorithm

Let (x, y) be an instance in the training set, X, where x is the data vector and $y \in \{-1, 1\}$ is the class label for that instance. We refer to the set of positive points in the data set as X^+ and negatives as X^-. To optimize the WMW statistic,

we would like to have all positive points ranked higher than all negative points, such as:

$$f(x_i) > f(x_j) \quad \forall x_i \in X^+, x_j \in X^-.$$

where f is some scoring function. A perfect separation of classes is impossible for most real world datasets. Therefore an LP is constructed to minimize some error term corresponding to the number of incorrect orderings. We construct a linear program with soft constraints penalizing negatives that are ranked above positives. This requires p (number of positive points) times n (number of negative points) constraints in total. Our formulation avoids combinatorial complexity by minimizing the error, z, which is the difference in scores of incorrectly ordered pairs. We select a scoring function, $f(x)$, such that it assigns a real valued score to each data point while making the optimization problem continuous. This leads to the following set of constraints:

$$f(x_i) - f(x_j) \geq \varepsilon - z_{ij} \quad \forall x_i \in X^+, x_j \in X^-. \tag{1}$$

where ε is a small quantity usually set to 1 and added to the right hand side to avoid strict inequalities in the constraints. The scoring function is similar to an SVM classification function without the intercept. The intercept is not necessary since it would add the same amount for all the points in the scoring function and would have no influence on both the relative scores and the overall ranking. The scoring function is defined as:

$$f(x) = \sum_{i \in X} y_i \alpha_i k(x, x_i) \tag{2}$$

where α_i represents a weight for each point in the training set. The function $k(.)$ is a kernel function through which we can induce nonlinearity in the linear constraints. In our objective function we would like to minimize the error, z's, along with the magnitude of the coefficients, α. We minimize α to maximize the separation margin similar to SVMs. The proposed objective function and LP can be obtained as follows by substituting the scoring function (2) in (1) and rearranging the left side of the inequality:

$$\begin{aligned} \min_{\alpha, z} \quad & e^T \alpha + C e^T z \\ \text{s.t.} \quad & \textstyle\sum_{i \in X} y_i \alpha_i [k(x^+, x_i) - k(x^-, x_i)] \geq 1 - z_{\pm}, \\ & \forall x^+ \in X^+, x^- \in X^- \\ & \alpha, z \geq 0 \end{aligned} \tag{3}$$

where e is the unit vector, C is the tradeoff parameter between the two parts of the objective and $z_{\pm}$ represents an error term for each positive/negative pair. In our experiments we use both RBF (4) and polynomial (5) kernels of the form

$$k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \tag{4}$$

$$k(x_i, x_j) = (1 + x_i x_j)^p \tag{5}$$

where γ is the RBF parameter that controls the smoothness of decision surface and p is the polynomial parameter that controls the degree or complexity. The output of the LP gives the optimal set of weights, α^*. The instances with non-zero weights, which we will call $ranking\ vectors$, represent the unique points that influence the ranking. By using a kernel, we are mapping the input space to a feature space defined by the kernel function. This makes it possible to rank the data points which were otherwise unrankable, in the mapped space. We tested our algorithm using an RBF kernel as well as polynomial kernels with order 2 and 3. Once the optimal weights are found for each point in the training set, α^*, we then use the scoring function

$$f(x_{test}) = \sum_{i \in X} y_i \alpha_i^* k(x_{test}, x_i) \tag{6}$$

to obtain scores for each test point. Finally we can rank the test points by the scores obtained.

Our algorithm differs from Brefeld et al. [3] in that we directly solve the primal problem with the nonlinear kernel function. The major advantage of the primal formulation is that, it requires less resources to solve the problem. It only needs to compute a size n^2 kernel matrix to cover all the constraints while Brefeld's dual formulation needs to compute a size $(n^+ \times n^-)^2$ (number of positive points times negative points) kernel matrix. Also the number of α variables is n for the primal form and $n^+ \times n^-$ for the dual form. In addition, the primal form provides flexibility in picking the objective function. In our case, 1-norm is used for both the error terms and the weights in the objective function so that the whole problem becomes a linear program. In addition, our algorithm produces n ranking vectors to score future test points against $n^+ \times n^-$ ranking pairs by Brefeld's algorithm. In practical applications, ranking vectors can provide more intuitive explanation then ranking pairs.

When an RBF kernel is used, our approach is similar to distance-weighted k-nearest neighbor (kNN) algorithm, where k is equal to the number of ranking vectors. Consider the scoring function of distance-weighted kNN where only the ranking vectors are used as training points, $f(x_{test}) = \sum_{i \in X} y_i \times distance(x_{test}, x_i)^{-1}$. If the inverse of RBF is used as the distance function such that $distance(x_i, x_j) = e^{\gamma \|x_i - x_j\|^2}$, the above scoring function is different from (6) only by the coefficient α_i, which is obtained by the optimization procedure in our approach.

A linear programming formulation that includes pairwise constraints presents a challenge such that the number of constraints grow quadratic with the number of data points. Therefore we implement a chunking like approach to speed up the solving time of the optimization problem. The variations we experimented with are explained in the next section. A general overview of the complete algorithm is shown in Figure 1.

Create all possible constraints, C
Divide C into manageable chunks, C_i
Initialize empty constraint set S
While size of S not converged
  For each chunk of constraints, $c_i \subset C$,
    Solve (3) with constraints $S + c_i$, obtaining α^*
    Evaluate α^* on $C - c_i$ for pairwise violations
    Add violated constraints to S
  End For
End While

Fig. 1. Algorithm Outline

III. Results and Discussion

In our experiments we used 11 data sets from the UCI repository. Multi-class data sets are converted to binary classification problems by using one class vs. others scheme. Details of these conversions are given in Table I. We compared our results with a regular 2-norm SVM using RBF and Polynomial kernels.

We implemented our algorithm in Matlab and used CPLEX for optimization in the Matlab environment. For performance comparisons with SVM we used the Sequential Minimal Optimization (SMO) algorithm [14] implemented in WEKA [23]. We constructed a grid search to find the best RBF parameter settings for both our ranker LP and SVM using $\gamma = \{0.001, 0.01, 0.1, 1\}$ and $C = \{1000, 100, 10, 1\}$. Best results for both algorithms were obtained when $C = \{1, 100\}$ and $\gamma = 0.01$. Also for the polynomial kernel comparisons we used $p = \{2, 3\}$ as the degree of the polynomial. For all the datasets, we averaged 5×10-fold cross validation results.

To investigate the effects of rank optimization on classification performance we also included accuracy results from both algorithms. We believe that the nature of the optimization that ranks positive points above negatives could have a favorable effect on accuracy since the process also enforces separation of two classes. To obtain accuracy for our algorithm we find the threshold (score) that gives the best accuracy value on the training set. We used the same threshold on the test set to classify the test points.

Table II shows that our ranking algorithm performed better in general then SVM both in ranking and accuracy performance when RBF kernel is used. The last row of the table shows statistical comparisons in the form of (win, loss, tie) where significance was set at the 0.05 level. Our ranker, optimized for ranking, was significantly better in AUC performance for eight datasets and worse for two. When accuracies were compared, ranker won four times and lost twice. We got similar results for parameter settings of $C = 100, \gamma = 0.01$ which we did not include here. We have experimented with polynomial kernels in our algorithm however the results we obtained so far are not promising. The results for polynomial kernel is given in Table III. In this case there was only one significant win versus six significant losses. We are currently investigating the possible reasons for this. Intuitively the choice of kernels and their performance should be data dependent which is the case in classification. However, in our evaluations RBF kernel performed better in almost all the datasets while polynomial kernel performed poorly in most cases. In the light of these results we believe that RBF kernels are better suited for ranking problems than polynomial kernels.

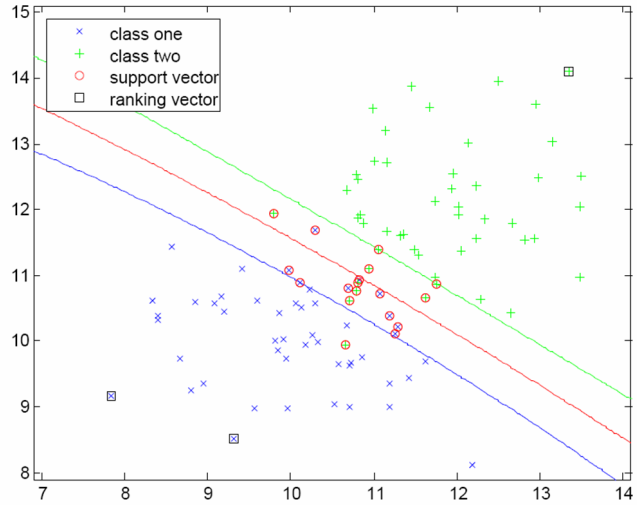

Fig. 2. Ranking vs Support Vectors: Linearly separable case

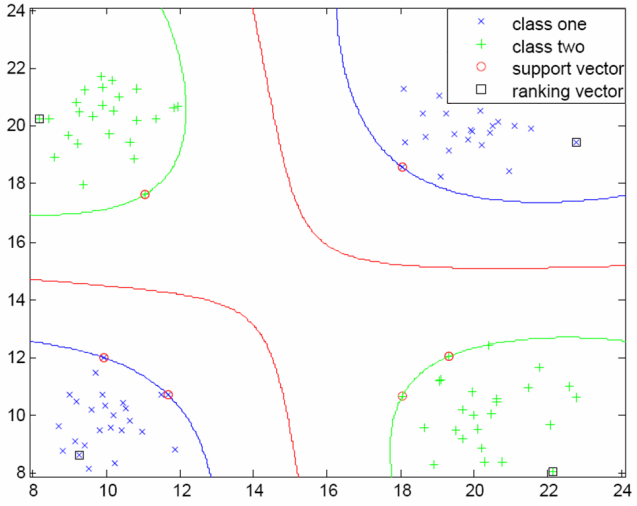

Fig. 3. Ranking vs Support Vectors: Linearly non-separable case

We observe that ranking vectors are quite different than support vectors. We created two artificial data sets to observe those differences. Figures 2 and 3 show two different two-dimensional problems where one problem requires a linear decision surface and the other requires a nonlinear surface.

TABLE I

OVERVIEW OF THE DATASETS AND MODIFICATION DETAILS

Datasets	**# of points**	**# attributes**	**% rare class**	**# of class - Comments**
BOSTON	506	14	9	(MEDV $<$ 35) = 1, REST = 0
ECOLI	336	8	15	"PP" = 1, REST = 0
GLASS	214	10	14	"7" = 1, REST = 0
HEART	270	14	44	2
SONAR	208	61	47	2
SPECTF	351	45	28	2
CANCER(WPBC)	194	34	24	2
IONOSPHERE	351	35	36	2
HABERMAN	306	4	26	2
LIVER(BUPA)	345	7	42	2
CANCER(WBC)	699	10	34	2

TABLE II

AUC & ACCURACY RESULTS USING RBF KERNEL FROM 5$\times$10-FOLD CROSS-VALIDATION.

$\gamma = 0.01$,$C = 1$	**AUC**		**Accuracy**	
Datasets	**LP Ranker**	**SVM**	**LP Ranker**	**SVM**
BOSTON	0.9732(0.0034)	0.9535 (0.0085)	0.9598 (0.0021)	0.9476 (0.0021)
ECOLI	0.9632(0.0051)	0.9387 (0.0069)	0.9602 (0.0035)	0.9120 (0.0035)
GLASS	0.9753(0.0032)	0.9590 (0.0035)	0.9741 (0.0044)	0.9608 (0.0044)
HEART	0.9112(0.0046)	0.9012 (0.0039)	0.8185 (0.0042)	0.8274 (0.0042)
SONAR	0.8735(0.0079)	0.7950 (0.0092)	0.7895 (0.0123)	0.7108 (0.0123)
SPECTF	0.9127(0.0006)	0.8835 (0.0055)	0.7927 (0.0109)	0.7995 (0.0109)
WPBC	0.7624(0.0299)	0.7649 (0.0064)	0.7893 (0.0085)	0.7851 (0.0085)
IONOSPHERE	0.9554(0.0079)	0.9196 (0.0034)	0.9270 (0.0048)	0.8904 (0.0048)
HABERMAN	0.5387(0.0304)	0.7028 (0.0164)	0.7068 (0.0037)	0.7434 (0.0093)
LIVER	0.5647(0.0267)	0.6912 (0.0082)	0.5607 (0.0093)	0.6600 (0.0037)
WBC	0.9955(0.0006)	0.9949 (0.0004)	0.9716 (0.0011)	0.9663 (0.0011)
(W,L,T)	(8,2,1)		(4,2,5)	

TABLE III

AUC & ACCURACY RESULTS USING POLYNOMIAL KERNEL FROM 5$\times$10-FOLD CROSS-VALIDATION.

$p = 3$,$C = 1$	**AUC**		**Accuracy**	
Datasets	**LP Ranker**	**SVM**	**LP Ranker**	**SVM**
BOSTON	0.9636 (0.0063)	0.9812 (0.0040)	0.9538(0.0041)	0.9569 (0.0041)
ECOLI	0.9659 (0.0063)	0.9519 (0.0058)	0.9364(0.0047)	0.9180 (0.0047)
GLASS	0.9647 (0.0165)	0.9660 (0.0179)	0.9759(0.0051)	0.9667 (0.0051)
HEART	0.8417 (0.0079)	0.8471 (0.0103)	0.7807(0.0210)	0.8015 (0.0210)
SONAR	0.9075 (0.0108)	0.9472 (0.0088)	0.8198(0.0098)	0.8576 (0.0098)
SPECTF	0.9124 (0.0126)	0.9332 (0.0029)	0.9156(0.0081)	0.8751 (0.0081)
WPBC	0.7699 (0.0278)	0.7895 (0.0232)	0.7969(0.0123)	0.7598 (0.0123)
IONOSPHERE	0.9326 (0.0058)	0.9619 (0.0066)	0.8928(0.0069)	0.8843 (0.0069)
HABERMAN	0.6866 (0.0216)	0.7076 (0.0156)	0.7293(0.0095)	0.7321 (0.0095)
LIVER	0.7159 (0.0087)	0.7004 (0.0111)	0.6963(0.0128)	0.6712 (0.0128)
WBC	0.9753 (0.0041)	0.9954 (0.0004)	0.9500(0.0040)	0.9613 (0.0040)
(W,L,T)	(1,6,4)		(3,3,5)	

For the data in the figures, RBF kernel is used for both SVM and ranker LP. While support vectors, as expected, are the points that appear close to the boundary, ranking vectors are usually positioned at the extremity of one class in the given two dimensional problems. As can be seen in Figure 2 only three and in Figure 2 only four ranking vectors were necessary to optimally rank all the points. Ranking vectors are also usually fewer in number and are the influential points for ranking the data points. Figures 2 and 3 also illustrate the power of RBF kernels for its applicability to linear problems as well as nonlinear ones.

A remaining challenge is the quadratic expansion of the number of constraints with the increasing number of data points. Ideally the number of constraints needed to obtain the real solution is equal to the number of positive points times the number of negatives. This number grows very quickly as the number of data points increases making the linear program solving time unreasonably long. We have tried several speed up tricks, related to chunking [1], to reduce the number of constraints by removing potentially redundant ones. Note that the schemes we implement here can still obtain exact solution to the problem. Chunking divides the data into manageable bins and optimizes separately so that the whole problem can be solved in reasonable time. After solving for each chunk, points that are most influential (points with nonzero α's) are added to the next bin as the algorithm iterates through the whole dataset. This approach did not work well for our formulations. The reason is that in our case non-zero α's represent ranking vectors. They are far away from the decision boundary and in general are not necessarily relevant to other chunks as in regular chunking.

We experimented with two variations of chunking. In our first scheme, after each chunk we evaluated the weights on the remaining data points, and for the next chunk we add to the set of previously violated constraints only those constraints that represent pair-wise violations. In this case we observed a majority of the violated constraints accumulating during the first few iterations instead of showing a gradual increase through iterations as we expected. As a second approach, to reduce the effect of a sudden increase of constraints, for each chunk we added only the top k most violated constraints in the LP where we set k to be 1000. This seemed to work well in most of the datasets. In this case the number of passes, hence the number of iterations until convergence increased slowing the LP solution time. An overview of the speed up comparisons are given in Table IV. The second column shows the total number of possible constraints for the given dataset. Columns 3 to 5 show the results for the first approach and columns 6 to 8 show the results for the second approach. The column labeled 'pass' in columns 3 and 6 represents the number of full passes on the dataset until the number of retained constraints converge. With the implemented approaches we were able to significantly reduce the number of constraints needed to solve the optimization problem in most of the datasets. However, the overall speed up was not as impressive as we thought would be. We are still investigating similar approaches to reduce LP solution times.

Several researchers also tackled this problem by an approximation heuristic. Herschtal et al. used a random sampling approach for reducing the number of constraints [10]. Rakotomamonjy suggested reducing the number of constraints in the optimization problem by only considering the constraints created by a predefined number of nearest positive neighbors for each negative point [18]. Brefeld et al. proposed a clustering approach for the reduction of constraints [3]. All of these heuristics helped speed up the time to obtain a solution, however they are all approximate methods and hence the performance of each heuristic would vary greatly with the nature of the data.

IV. Conclusions

We have introduced an LP formulation to optimize ranking performance by maximizing an approximation to the WMW statistic. Our results show that our algorithm which is similar to 1-norm SVMs, gives superior ranking performance compared to a regular 2-norm SVM when RBF kernels are used. Performance degraded when we used polynomial kernels. We have tackled speed issues with variations to the chunking approach, but we were not able to significantly improve LP solving times by using exact heuristics. As a next step we intend to look into approximate heuristics to speed up LP times. We will also investigate the loss of performance with polynomial kernels.

References

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[2] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[3] U. Brefeld and T. Scheffer. Auc maximizing support vector learning. In *Preceedings of ICML 2005 workshop on ROC Analysis in Machine Learning*, 2005.

[4] R. Caruana, S. Baluja, and T. Mitchell. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 959–965. The MIT Press, 1996.

[5] W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 243–270. The MIT Press, 1998.

[6] C. Cortes and M. Mohri. Auc optimization vs. error rate minimization. In S. Thrun, L. Saul, and B. Scholkopf, editors, *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

TABLE IV
COMPARISON OF SPEED UP HEURISTICS

		1st Speed up Approach			2nd Speed up Approach		
Datasets	**# of constraints**	**pass**	**# of constraints in LP**	**% reduction**	**pass**	**# of constraints in LP**	**% reduction**
BOSTON	18172	2	4920	72.92	5	4501	75.23
ECOLI	12336	3	4170	66.20	4	4312	65.05
GLASS	4509	3	3036	32.67	3	3039	32.60
HEART	14824	3	6730	54.60	5	4692	68.35
SONAR	8888	5	3874	56.41	5	3939	55.68
SPECTF	20240	3	4775	76.41	5	3186	84.26
CANCER(WPBC)	5628	3	3981	29.26	5	3585	36.30
IONOSPHERE	23142	2	5850	74.72	4	5060	78.13
HABERMAN	15022	2	12580	16.26	7	6409	57.34
LIVER(BUPA)	23711	2	22500	5.11	8	8250	65.21
CANCER(WBC)	86616	2	7370	91.49	2	3288	96.20

[7] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[8] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[9] J.A. Hanley and B.J. McNeil. The meaning and the use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

[10] A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *ICML '04: Twenty-First International Conference on Machine Learning*. ACM Press, 2004.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM Press, 2002.

[12] C. Ling, J. Huang, and H. Zhang. AUC: A statistically consistent and more discriminating measure than accuracy. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2003.

[13] H.B. Mann and D.R. Whitney. On a test whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statististics*, 18:50–60, 1947.

[14] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In C. Burges B. Schlkopf and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[15] F.J. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Knowledge Discovery and Data Mining*, pages 43–48, 1997.

[16] K. Obermayer R. Herbrich, T. Graepel. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. The MIT Press, 2000.

[17] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. In *KDD '05: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*. ACM Press, 2005.

[18] A. Rakotomamonjy. Optimizing area under ROC curve with SVMs. In *ROC Analysis in Artificial Intelligence*, pages 71–80, 2004.

[19] C. Rudin, C. Cortes, M. Mohri, and R. Schapire. Margin-based ranking meets boosting in the middle. In *In 18th Annual Conference on Computational Learning Theory*, 2005.

[20] K.A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 160–163. Morgan Kaufman, 1989.

[21] C. Vogt and G. Cottrell. Using d' to optimize rankings. Technical Report CS98-601, U.C. San Diego, CSE Department, 1998.

[22] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[23] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

[24] L. Yan, R.H. Dodier, M. Mozer, and R.H. Wolniewicz. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *International Conference on Machine Learning*, pages 848–855, 2003.