An Alternative Ranking Problem for Search Engines

Corinna Cortes[1], Mehryar Mohri[2,1], and Ashish Rastogi[2]

[1] Google Research,
76 Ninth Avenue,
New York, NY 10011
[2] Courant Institute of Mathematical Sciences,
251 Mercer Street
New York, NY 10012.

Abstract. This paper examines in detail an alternative ranking problem for search engines, movie recommendation, and other similar ranking systems motivated by the requirement to not just accurately predict pairwise ordering but also preserve the magnitude of the preferences or the difference between ratings. We describe and analyze several cost functions for this learning problem and give stability bounds for their generalization error, extending previously known stability results to non-bipartite ranking and magnitude of preference-preserving algorithms. We present algorithms optimizing these cost functions, and, in one instance, detail both a batch and an on-line version. For this algorithm, we also show how the leave-one-out error can be computed and approximated efficiently, which can be used to determine the optimal values of the trade-off parameter in the cost function. We report the results of experiments comparing these algorithms on several datasets and contrast them with those obtained using an AUC-maximization algorithm. We also compare training times and performance results for the on-line and batch versions, demonstrating that our on-line algorithm scales to relatively large datasets with no significant loss in accuracy.

1 Motivation

The learning problem of ranking has gained an increasing amount of interest in the machine learning community over the last decade, in part due to the remarkable success of web search engines and recommender systems (Freund et al., 1998; Crammer & Singer, 2001; Joachims, 2002; Shashua & Levin, 2003; Cortes & Mohri, 2004; Rudin et al., 2005; Agarwal & Niyogi, 2005). The recent Netflix challenge has further stimulated the learning community fueling its research with invaluable datasets (Netflix, 2006).

The goal of information retrieval engines is to return a set of documents, or clusters of documents, ranked in decreasing order of relevance to the user. The order may be common to all users, as with most search engines, or tuned to individuals to provide personalized search results or recommendations. The accuracy of this ordered list is the key quality measure of theses systems.

In most previous research studies, the problem of ranking has been formulated as that of learning from a labeled sample of pairwise preferences a scoring function with small pairwise misranking error (Freund et al., 1998; Herbrich et al., 2000; Crammer & Singer, 2001; Joachims, 2002; Rudin et al., 2005; Agarwal & Niyogi, 2005). But this formulation suffers some short-comings.

Firstly, most users inspect only the top results. Thus, it would be natural to enforce that the results returned near the top be particularly relevant and correctly ordered. The quality and ordering of the results further down the list matter less. An average pairwise misranking error directly penalizes errors at both extremes of a list more heavily than errors towards the middle of the list, since errors at the extremes result in more misranked pairs. However, one may wish to explicitly encode the requirement of ranking quality at the top in the cost function. One common solution is to weigh examples differently during training so that more important or high-quality results be assigned larger weights. This imposes higher accuracy on these examples, but does not ensure a high-quality ordering at the top. A good formulation of this problem leading to a convex optimization problem with a unique minimum is still an open question.

Another shortcoming of the pairwise misranking error is that this formulation of the problem and thus the scoring function learned ignore the magnitude of the preferences. In many applications, it is not sufficient to determine if one example is preferred to another. One may further request an assessment of how large that preference is. Taking this magnitude of preference into consideration is critical, for example in the design of search engines, which originally motivated our study, but also in other recommendation systems. For a recommendation system, one may choose to truncate the ordered list returned where a large gap in predicted preference is found. For a search engine, this may trigger a search in parallel corpora to display more relevant results.

This motivated our study of the problem of ranking while preserving the magnitude of preferences, which we will refer to in short by *magnitude-preserving ranking*.[3] The problem that we are studying bears some resemblance with that of ordinal regression (McCullagh, 1980; McCullagh & Nelder, 1983; Shashua & Levin, 2003; Chu & Keerthi, 2005). It is however distinct from ordinal regression since in ordinal regression the magnitude of the difference in target values is not taken into consideration in the formulation of the problem or the solutions proposed. The algorithm of Chu and Keerthi (2005) does take into account the ordering of the classes by imposing that the thresholds be monotonically increasing, but this still ignores the difference of target values and thus does not follow the same objective. A crucial aspect of the algorithms we propose is that they penalize misranking errors more heavily in the case of larger magnitudes of preferences.

We describe and analyze several cost functions for this learning problem and give stability bounds for their generalization error, extending previously known stability results to non-bipartite ranking and magnitude of preference-preserving algorithms. In particular, our bounds extend the framework of (Bousquet &

[3] This paper is an extended version of (Cortes et al., 2007).

Elisseeff, 2000; Bousquet & Elisseeff, 2002) to the case of cost functions over pairs of examples, and extend the bounds of Agarwal and Niyogi (2005) beyond the bi-partite ranking problem. Our bounds also apply to algorithms optimizing the so-called *hinge rank loss*.

We present several algorithms optimizing these cost functions, and in one instance detail both a batch and an on-line version. For this algorithm, MPRank, we also show how the leave-one-out error can be computed and approximated efficiently, which can be used to determine the optimal values of the trade-off parameter in the cost function. We also report the results of experiments comparing these algorithms on several datasets and contrast them with those obtained using RankBoost (Freund et al., 1998; Rudin et al., 2005), an algorithm designed to minimize the exponentiated loss associated with the Area Under the ROC Curve (AUC), or pairwise misranking. We also compare training times and performance results for the on-line and batch versions of MPRank, demonstrating that our on-line algorithm scales to relatively large datasets with no significant loss in accuracy.

The remainder of the paper is organized as follows. Section 2 describes and analyzes our algorithms in detail. Section 3 presents stability-based generalization bounds for a family of magnitude-preserving algorithms. Section 4 presents the results of our experiments with these algorithms on several datasets.

2 Algorithms

Let S be a sample of m labeled examples drawn i.i.d. from a set X according to some distribution D:

$$(x_1, y_1), \ldots, (x_m, y_m) \in X \times \mathbb{R}. \tag{1}$$

For any $i \in [1, m]$, we denote by S^{-i} the sample derived from S by omitting example (x_i, y_i), and by S^i the sample derived from S by replacing example (x_i, y_i) with an other example (x'_i, y'_i) drawn i.i.d. from X according to D. For convenience, we will sometimes denote by $y_x = y_i$ the label of a point $x = x_i \in X$.

The quality of the ranking algorithms we consider is measured with respect to pairs of examples. Thus, a cost functions c takes as arguments two sample points. For a fixed cost function c, the empirical error $\widehat{R}(h, S)$ of a hypothesis $h : X \mapsto \mathbb{R}$ on a sample S is defined by:

$$\widehat{R}(h, S) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} c(h, x_i, x_j). \tag{2}$$

The true error $R(h)$ is defined by

$$R(h) = \mathrm{E}_{x,x' \sim D}[c(h, x, x')]. \tag{3}$$

2.1 Cost functions

We introduce several cost functions related to magnitude-preserving ranking. The first one is the so-called *hinge rank loss* which is a natural extension of the pairwise misranking loss (Cortes & Mohri, 2004; Rudin et al., 2005). It penalizes a pairwise misranking by the magnitude of preference predicted or the nth power of that magnitude ($n = 1$ or $n = 2$):

$$c^n_{\mathrm{HR}}(h, x, x') = \begin{cases} 0, \text{ if } (h(x') - h(x))(y_{x'} - y_x) \geq 0 \\ \big|(h(x') - h(x))\big|^n, \text{ otherwise.} \end{cases} \tag{4}$$

c^n_{HR} does not take into consideration the true magnitude of preference $y_{x'} - y_x$ for each pair (x, x') however. The following cost function has this property and penalizes deviations of the predicted magnitude with respect to the true one. Thus, it matches our objective of magnitude-preserving ranking ($n = 1, 2$):

$$c^n_{\mathrm{MP}}(h, x, x') = \big|(h(x') - h(x)) - (y_{x'} - y_x)\big|^n. \tag{5}$$

A one-sided version of that cost function penalizing only misranked pairs is given by ($n = 1, 2$):

$$c^n_{\mathrm{HMP}}(h, x, x') = \begin{cases} 0, \text{ if } (h(x') - h(x))(y_{x'} - y_x) \geq 0 \\ \big|(h(x') - h(x)) - (y_{x'} - y_x)\big|^n, \text{ otherwise.} \end{cases} \tag{6}$$

Finally, we will consider the following cost function derived from the ϵ-insensitive cost function used in SVM regression (SVR) (Vapnik, 1998) ($n = 1, 2$):

$$c^n_{\mathrm{SVR}}(h, x, x') = \begin{cases} 0, \text{ if } |\big[(h(x') - h(x)) - (y_{x'} - y_x)\big]| \leq \epsilon \\ \big|(h(x') - h(x)) - (y_{x'} - y_x) - \epsilon\big|^n, \text{ otherwise.} \end{cases} \tag{7}$$

Note that all of these cost functions are convex functions of $h(x)$ and $h(x')$.

2.2 Objective functions

The regularization algorithms based on the cost functions c^n_{MP} and c^n_{SVR} correspond closely to the idea of preserving the magnitude of preferences since these cost functions penalize deviations of a predicted difference of score from the target preferences. We will refer by MPRank to the algorithm minimizing the regularization-based objective function based on c^n_{MP}:

$$F(h, S) = \|h\|^2_K + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} c^n_{\mathrm{MP}}(h, x_i, x_j), \tag{8}$$

and by SVRank to the one based on the cost function c^n_{SVR}

$$F(h, S) = \|h\|^2_K + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} c^n_{\mathrm{SVR}}(h, x_i, x_j). \tag{9}$$

For a fixed n, $n = 1, 2$, the same stability bounds hold for both algorithms as seen in the following section. However, their time complexity is significantly different.

2.3 MPRank

We will examine the algorithm in the case $n = 2$. Let $\Phi : X \mapsto F$ be the mapping from X to the reproducing Hilbert space. The hypothesis set H that we are considering is that of linear functions h, that is $\forall x \in X, h(x) = w \cdot \Phi(x)$. The objective function can be expressed as follows

$$\begin{aligned} F(h, S) &= \|w\|^2 + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\big[(w \cdot \Phi(x_j) - w \cdot \Phi(x_i)) - (y_j - y_i)\big]^2 \\ &= \|w\|^2 + \frac{2C}{m}\sum_{i=1}^{m}\|w \cdot \Phi(x_i) - y_i\|^2 - 2C\|w \cdot \bar{\Phi} - \bar{y}\|^2, \end{aligned}$$

where $\bar{\Phi} = \frac{1}{m}\sum_{i=1}^{m}\Phi(x_i)$ and $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i$. The objective function can thus be written with a single sum over the training examples, which results in a more efficient computation of the solution.

Let N be the dimension of the feature space F. For $i = 1, \ldots, m$, let $\mathbf{M}_{x_i} \in \mathbb{R}^{N\times 1}$ denote the column matrix representing $\Phi(x_i)$, $\mathbf{M}_{\bar{\Phi}} \in \mathbb{R}^{N\times 1}$ a column matrix representing $\bar{\Phi}$, $\mathbf{W} \in \mathbb{R}^{N\times 1}$ a column matrix representing the vector w, $\mathbf{M}_Y \in \mathbb{R}^{m\times 1}$ a column matrix whose ith component is y_i, and $\mathbf{M}_{\bar{Y}} \in \mathbb{R}^{m\times 1}$ a column matrix with all its components equal to $\bar{y}$. Let $\mathbf{M}_X, \mathbf{M}_{\bar{X}} \in \mathbb{R}^{N\times m}$ be the matrices defined by:

$$\mathbf{M}_X = [\mathbf{M}_{x_1} \ldots \mathbf{M}_{x_m}] \quad \mathbf{M}_X = [\mathbf{M}_{\bar{\Phi}} \ldots \mathbf{M}_{\bar{\Phi}}]. \tag{10}$$

Then, the expression giving F can be rewritten as

$$F = \|\mathbf{W}\|^2 + \frac{2C}{m}\|\mathbf{M}_X^\top \mathbf{W} - \mathbf{M}_Y\|^2 - \frac{2C}{m}\|\mathbf{M}_{\bar{X}}^\top \mathbf{W} - \mathbf{M}_{\bar{Y}}\|^2.$$

The gradient of F is then given by: $\nabla F = 2\mathbf{W} + \frac{4C}{m}\mathbf{M}_X(\mathbf{M}_X^\top \mathbf{W} - \mathbf{M}_Y) - \frac{4C}{m}\mathbf{M}_{\bar{X}}(\mathbf{M}_{\bar{X}}^\top \mathbf{W} - \mathbf{M}_{\bar{Y}})$. Setting $\nabla F = 0$ yields the unique closed form solution of the convex optimization problem:

$$\mathbf{W} = C'\big(\mathbf{I} + C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top\big)^{-1}(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}}), \tag{11}$$

where $C' = \frac{2C}{m}$. Here, we are using the identity $\mathbf{M}_X\mathbf{M}_X^\top - \mathbf{M}_{\bar{X}}\mathbf{M}_{\bar{X}}^\top = (\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top$, which is not hard to verify. This provides the solution of the primal problem. Using the fact the matrices $(\mathbf{I} + C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top\big)^{-1}$ and $\mathbf{M}_X - \mathbf{M}_{\bar{X}}$ commute leads to:

$$\mathbf{W} = C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})\big(\mathbf{I} + C'(\mathbf{M}_X - \mathbf{M}_{\bar{X}})(\mathbf{M}_X - \mathbf{M}_{\bar{X}})^\top\big)^{-1}(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}}). \tag{12}$$

This helps derive the solution of the dual problem. For any $x' \in X$,

$$h(x') = C'\mathbf{K}'(\mathbf{I} + \bar{\mathbf{K}})^{-1}(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}}), \tag{13}$$

where $\mathbf{K}' \in \mathbb{R}^{1\times m}$ is the row matrix whose jth component is

$$K(x', x_j) - \frac{1}{m}\sum_{k=1}^{m} K(x', x_k) \tag{14}$$

and $\bar{\mathbf{K}}$ is the kernel matrix defined by

$$\frac{1}{C'}(\bar{\mathbf{K}})_{ij} = K(x_i, x_j) - \frac{1}{m}\sum_{k=1}^{m}(K(x_i, x_k) + K(x_j, x_k)) + \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l=1}^{m} K(x_k, x_l),$$

for all $i, j \in [1, m]$. The solution of the optimization problem for MPRank is close to that of a kernel ridge regression problem, but the presence of additional terms makes it distinct, a fact that can also be confirmed experimentally. However, remarkably, it has the same computational complexity, due to the fact that the optimization problem can be written in terms of a single sum, as already pointed out above. The main computational cost of the algorithm is that of the matrix inversion, which can be computed in time $O(N^3)$ in the primal, and $O(m^3)$ in the dual case, or $O(N^{2+\alpha})$ and $O(m^{2+\alpha})$, with $\alpha \approx .376$, using faster matrix inversion methods such as that of Coppersmith and Winograd.

2.4 SVRank

We will examine the algorithm in the case $n = 1$. As with MPRank, the hypothesis set H that we are considering here is that of linear functions h, that is $\forall x \in X, h(x) = w \cdot \Phi(x)$. The constraint optimization problem associated with SVRank can thus be rewritten as

$$\begin{aligned} &\text{minimize} \quad F(h, S) = \|w\|^2 + C\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\xi_{ij} + \xi^*_{ij}) \\ &\text{subject to} \begin{cases} w \cdot (\Phi(x_j) - \Phi(x_i)) - (y_j - y_i) \leq \epsilon + \xi_{ij} \\ (y_j - y_i) - w \cdot (\Phi(x_j) - \Phi(x_i)) \leq \epsilon + \xi^*_{ij} \\ \xi_{ij}, \xi^*_{ij} \geq 0, \end{cases} \end{aligned}$$

for all $i, j \in [1, m]$. Note that the number of constraints are quadratic with respect to the number of examples. Thus, in general, this results in a problem that is more costly to solve than that of MPRank.

Introducing Lagrange multipliers $\alpha_{ij}, \alpha^*_{ij} \geq 0$, corresponding to the first two sets of constraints and $\beta_{ij}, \beta^*_{ij} \geq 0$ for the remaining constraints leads to the following Lagrange function

$$\begin{aligned} L = \|w\|^2 + C\tfrac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\xi_{ij} + \xi^*_{ij}) + \\ \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_{ij}(w \cdot (\Phi(x_j) - \Phi(x_i)) - (y_j - y_i) - \epsilon + \xi_{ij}) + \\ \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha^*_{ij}(-w \cdot (\Phi(x_j) - \Phi(x_i)) + (y_j - y_i) - \epsilon + \xi^*_{ij}) + \\ \sum_{i=1}^{m}\sum_{j=1}^{m}(\beta_{ij}\xi_{ij} + \beta^*_{ij}\xi^*_{ij}). \end{aligned}$$

Taking the gradients, setting them to zero, and applying the Karush-Kuhn-Tucker conditions leads to the following dual maximization problem

$$\begin{aligned}\text{maximize } & \frac{1}{2}\sum_{i,j=1}^{m}\sum_{k,l=1}^{m}(\alpha^*_{ij}-\alpha_{ij})(\alpha^*_{kl}-\alpha_{kl})K_{ij,kl}- \\ & \epsilon\sum_{i,j=1}^{m}(\alpha^*_{ij}-\alpha_{ij})+\sum_{i,j=1}^{m}(\alpha^*_{ij}-\alpha_{ij})(y_j-y_i) \\ \text{subject to } & 0\le\alpha_{ij},\alpha^*_{ij}\le C, \forall i,j\in[1,m],\end{aligned}$$

where $K_{ij,kl} = K(x_i,x_k)+K(x_j,x_l)-K(x_i,x_l)-K(x_j,x_k)$. This quadratic optimization problem can be solved in a way similar to SVM regression (SVR) (Vapnik, 1998) by defining a kernel K' over pairs with $K'((x_i,x_j),(x_k,x_l)) = K_{ij,kl}$, for all $i,j,k,l\in[1,m]$, and associating the target value y_i-y_j to the pair (x_i,x_j).

The computational complexity of the quadratic programming with respect to pairs makes this algorithm less attractive for relatively large samples.

2.5 On-line Version of MPRank

Recall from Section 2.3 that the cost function for MPRank can be written as

$$F(h,S)=\|w\|^2+\frac{2C}{m}\sum_{i=1}^{m}\big((w\cdot\Phi(x_i)-y_i)^2-(w\cdot\bar{\Phi}-\bar{y})\big)^2. \tag{15}$$

This expression suggests that the solution w can be found by solving the following optimization problem

$$\begin{aligned}\underset{w}{\text{minimize}}\quad & F=\|w\|^2+\frac{2C}{m}\sum_{i=1}^{m}\xi_i^2 \\ \text{subject to } & (w\cdot\Phi(x_i)-y_i)-(w\cdot\bar{\Phi}-\bar{y})=\xi_i \text{ for } i=1,\ldots,m\end{aligned}$$

Introducing the Lagrange multipliers β_i corresponding to the ith equality constraint leads to the following Lagrange function:

$$L(w,\xi,\beta)=\|w\|^2+\frac{2C}{m}\sum_{i=1}^{m}\xi_i^2-\sum_{i=1}^{m}\beta_i\big((w\cdot\Phi(x_i)-y_i)-(w\cdot\bar{\Phi}-\bar{y})-\xi_i\big)$$

Setting $\partial L/\partial w=0$, we obtain $w=\frac{1}{2}\sum_{i=1}^{m}\beta_i(\Phi(x_i)-\bar{\Phi})$, and setting $\partial L/\partial\xi_i=0$ leads to $\xi_i=-\frac{m}{4C}\beta_i$. Substituting these expression backs in and letting $\alpha_i=\beta_i/2$ result in the optimization problem

$$\underset{\alpha_i}{\text{maximize}}\ -\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\widetilde{K}(x_i,x_j)-\frac{m}{2C}\sum_{i=1}^{m}\alpha_i^2+2\sum_{i=1}^{m}\alpha_i\widetilde{y_i}, \tag{16}$$

where $\widetilde{K}(x_i,x_j)=K(x_i,x_j)-\frac{1}{m}\sum_{k=1}^{m}(K(x_i,x_k)+K(x_j,x_k))+\frac{1}{m^2}\sum_{k,l=1}^{m}K(x_k,x_l)$ and $\widetilde{y_i}=y_i-\bar{y}$.

Based on the expressions for the partial derivatives of the Lagrange function, we can now describe a gradient descent algorithm that avoids the prohibitive complexity of MPRank that is associated with matrix inversion:

```
1  for $i \leftarrow 1$ to $m$ do  $\alpha_i \leftarrow 0$
2  repeat
3          for $i \leftarrow 1$ to $m$
4            do  $\alpha_i \leftarrow \alpha_i + \eta\big[2(\widetilde{y}_i - \sum_{j=1}^{m} \alpha_j \widetilde{K}(x_i, x_j)) - \frac{m}{C}\alpha_i\big]$
5  until convergence
```

The gradient descent algorithm described above can be straightforwardly modified to an on-line algorithm where points in the training set are processed in T passes, one by one, and the complexity of updates for ith point is $O(i)$ leading to an overall complexity of $O(T \cdot m^2)$. Note that even using the best matrix inversion algorithms, one only achieves a time complexity of $O(m^{2+\alpha})$, with $\alpha \approx .376$. In addition to a favorable complexity if $T = o(m^{.376})$, an appealing aspect of the gradient descent based algorithms is their simplicity. They are quite efficient in practice for datasets with a large number of training points.

2.6 Leave-One-Out Analysis for MPRank

The leave-one-out error of a learning algorithm is typically costly to compute as it in general requires training the algorithm on m subsamples of the original sample. This section shows how the leave-one-out error of MPRank can be computed and approximated efficiently by extending the techniques of Wahba (1990).

The standard definition of the leave-one-out error holds for errors or cost functions defined over a single point. The definition can be extended to cost functions defined over pairs by leaving out each time one pair of points $(x_i, x_j), i \neq j$ instead of a single point.

To simplify the notation, we will denote $h_{S-\{x_i,x_j\}}$ by h_{ij} and $h_{S-\{x,x'\}}$ by $h_{xx'}$. The leave-one-out error of an algorithm L over a sample S returning the hypothesis h_{ij} for a training sample $S - \{x_i, x_j\}$ is then defined by

$$\mathrm{LOO}(L, S) = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j=1, i\neq j}^{m} c(h_{ij}, x_i, x_j). \tag{17}$$

The following proposition shows that with our new definition, the fundamental property of LOO is preserved.

Proposition 1. *Let $m \geq 2$ and let h' be the hypothesis returned by L when trained over a sample S' of size $m-2$. Then, the leave-one-out error over a sample S of size m is an unbiased estimate of the true error over a sample of size $m-2$:*

$$\mathrm{E}_{S\sim D}[\mathrm{LOO}(L, S)] = R(h'), \tag{18}$$

Proof. Since all points of S are drawn i.i.d. and according to the same distribution D,

$$\begin{aligned}
\mathrm{E}_{S\sim D}[\mathrm{LOO}(L,S)] &= \frac{1}{m(m-1)} \sum_{i,j=1, i\neq j}^{m} \mathrm{E}_{S\sim D}[c(h_{ij}, x_i, x_j)] && (19)\\
&= \frac{1}{m(m-1)} \sum_{x,x'\in S, x\neq x'}^{m} \mathrm{E}_{S\sim D, x,x'\in S}[c(h_{xx'}, x, x')] && (20)\\
&= \mathrm{E}_{S\sim D, x,x'\in S}[c(h_{xx'}, x, x')] && (21)
\end{aligned}$$

This last term coincides with $\mathrm{E}_{S',x,x'\sim D, |S'|=m-2}[c(h_{xx'}, x, x')] = R(h')$. □

In Section 2.3, it was shown that the hypothesis returned by MPRank for a sample S is given by $h(x') = C'\mathbf{K}'(\mathbf{I}+\bar{\mathbf{K}})^{-1}(\mathbf{M}_Y - \mathbf{M}_{\bar{Y}})$ for all $x' \in \mathbf{M}_X$. Let $\mathbf{K}_c$ be the matrix derived from $\mathbf{K}$ by replacing each entry $\mathbf{K}_{ij}$ of $\mathbf{K}$ by the sum of the entries in the same column $\sum_{j=1}^{m} \mathbf{K}_{ij}$. Similarly, let $\mathbf{K}_r$ be the matrix derived from $\mathbf{K}$ by replacing each entry of $\mathbf{K}$ by the sum of the entries in the same row, and let $\mathbf{K}_{rc}$ be the matrix whose entries all are equal to the sum of all entries of $\mathbf{K}$. Note that the matrix $\bar{\mathbf{K}}$ can be written as:

$$\frac{1}{C'}\bar{\mathbf{K}} = \mathbf{K} - \frac{1}{m}(\mathbf{K}_r + \mathbf{K}_c) + \frac{1}{m^2}\mathbf{K}_{rc}. \tag{22}$$

Let $\mathbf{K}''$ and $\mathbf{U}$ be the matrices defined by:

$$\mathbf{K}'' = \mathbf{K} - \frac{1}{m}\mathbf{K}_r \quad \text{and} \quad \mathbf{U} = C'\mathbf{K}''(\mathbf{I}+\bar{\mathbf{K}})^{-1}. \tag{23}$$

Then, for all $i \in [1, m]$, $h(x_i) = \sum_{k=1}^{m} \mathbf{U}_{ik}(y_k - \bar{y})$. In the remainder of this section, we will consider the particular case of the $n = 2$ cost function for MPRank, $c^2_{\mathrm{MP}}(h, x_i, x_j) = [(h(x_j) - y_j) - (h(x_i) - y_i)]^2$.

Proposition 2. *Let h' be the hypothesis returned by MPRank when trained on $S - \{x_i, x_j\}$ and let $\bar{h}' = \frac{1}{m}\sum_{k=1}^{m} h'(x_k)$. For all $i, j \in [1, m]$, let $\mathbf{V}_{ij} = \mathbf{U}_{ij} - \frac{1}{m-2}\sum_{k\notin\{i,j\}} \mathbf{U}_{ik}$. Then, the following identity holds for $c^2_{\mathrm{MP}}(h', x_i, x_j)$.*

$$\begin{aligned}
&\big[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big]^2 c^2_{\mathrm{MP}}(h', x_i, x_j) = \qquad (24)\\
&\big[(1-\mathbf{V}_{ii}-\mathbf{V}_{ij})(h(x_j)-y_j) - (1-\mathbf{V}_{ji}-\mathbf{V}_{jj})(h(x_i)-y_i)\\
&-[(1-\mathbf{V}_{ii}-\mathbf{V}_{ij})(\mathbf{V}_{jj}+\mathbf{V}_{ji})(1-\mathbf{V}_{ji}-\mathbf{V}_{jj})(\mathbf{V}_{ii}+\mathbf{V}_{ij})](\bar{h}'-\bar{y})\big]^2 .
\end{aligned}$$

Proof. By Equation 15, the cost function of MPRank can be written as:

$$F = \|w\|^2 + \frac{2C}{m}\sum_{k=1}^{m}\big[(h(x_k)-y_k) - (\bar{h}-\bar{y})\big]^2, \tag{25}$$

where $\bar{h} = \frac{1}{m}\sum_{k=1}^{m} h(x_k)$. h' is the solution of the minimization of F when the terms corresponding to x_i and x_j are left out. Equivalently, one can keep these terms but select new values for y_i and y_j to ensure that these terms are zero. Proceeding this way, the new values y'_i and y'_j must verify the following:

$$h'(x_i) - y'_i = h'(x_j) - y'_j = \bar{h}' - \bar{y}', \tag{26}$$

with $\bar{y}' = \frac{1}{m}[y'(x_i) + y'(x_j) + \sum_{k\not\in\{i,j\}} y_k]$. Thus, by Equation 23, $h'(x_i)$ is given by $h'(x_i) = \sum_{k=1}^{m} \mathbf{U}_{ik}(y_k - \overline{y}')$. Therefore,

$$\begin{aligned}
h'(x_i) - y_i &= \sum_{k\not\in\{i,j\}} \mathbf{U}_{ik}(y_k - \overline{y}') + \mathbf{U}_{ii}(y'_i - \overline{y}') + \mathbf{U}_{ij}(y'_j - \overline{y}') - y_i \\
&= \sum_{k\not\in\{i,j\}} \mathbf{U}_{ik}(y_k - \overline{y}) - \sum_{k\not\in\{i,j\}} \mathbf{U}_{ik}(\overline{y}' - \overline{y}) + \mathbf{U}_{ii}(h'(x_i) - \bar{h}') \\
&\quad + \mathbf{U}_{ij}(h'(x_j) - \bar{h}') - y_i \\
&= (h(x_i) - y_i) - \mathbf{U}_{ii}(y_i - \bar{y}) - \mathbf{U}_{ij}(y_j - \bar{y}) - \sum_{k\not\in\{i,j\}} \mathbf{U}_{ik}(\overline{y}' - \overline{y}) \\
&\quad + \mathbf{U}_{ii}(h'(x_i) - \bar{h}') + \mathbf{U}_{ij}(h'(x_j) - \bar{h}') \\
&= (h(x_i) - y_i) + \mathbf{U}_{ii}(h'(x_i) - y_i) + \mathbf{U}_{ij}(h'(x_j) - y_j) - (\mathbf{U}_{ii} + \mathbf{U}_{ij})(\bar{h}' - \bar{y}) \\
&\quad - \sum_{k\not\in\{i,j\}} \mathbf{U}_{ik}(\overline{y}' - \overline{y}) \\
&= (h(x_i) - y_i) + \mathbf{U}_{ii}(h'(x_i) - y_i) + \mathbf{U}_{ij}(h'(x_j) - y_j) - (\mathbf{U}_{ii} + \mathbf{U}_{ij})(\bar{h}' - \bar{y}) \\
&\quad - \sum_{k\not\in\{i,j\}} \mathbf{U}_{ik} \frac{1}{m-2}[(h'(x_i) - y_i) + (h'(x_j) - y_j) - 2(\bar{h}' - \bar{y})] \\
&= (h(x_i) - y_i) + \mathbf{V}_{ii}(h'(x_i) - y_i) + \mathbf{V}_{ij}(h'(x_j) - y_j) - (\mathbf{V}_{ii} + \mathbf{V}_{ij})(\bar{h}' - \bar{y}).
\end{aligned}$$

Thus,

$$(1 - \mathbf{V}_{ii})(h'(x_i) - y_i) - \mathbf{V}_{ij}(h'(x_j) - y_j) = (h(x_i) - y_i) - (\mathbf{V}_{ii} + \mathbf{V}_{ij})(\bar{h}' - \bar{y}),$$

Similarly, we have

$$-\mathbf{V}_{ji}(h'(x_i) - y_i) + (1 - \mathbf{V}_{jj})(h'(x_j) - y_j) = (h(x_j) - y_j) - (\mathbf{V}_{jj} + \mathbf{V}_{ji})(\bar{h}' - \bar{y}).$$

Solving the linear system formed by these two equations with unknown variables $(h'(x_i) - y_i)$ and $(h'(x_j) - y_j)$ gives:

$$\begin{aligned}
\big[(1 - \mathbf{V}_{jj})(1 - \mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big](h'(x_i) - y_i) &= (1 - \mathbf{V}_{jj})(h(x_i) - y_i) + \mathbf{V}_{ij}(h(x_j) - y_j) \\
&\quad - [(\mathbf{V}_{ii} + \mathbf{V}_{ij})(1 - \mathbf{V}_{jj}) + (\mathbf{V}_{jj} + \mathbf{V}_{ji})\mathbf{V}_{ij}](\bar{h}' - \bar{y}).
\end{aligned}$$

Similarly, we obtain:

$$\begin{aligned}
\big[(1 - \mathbf{V}_{jj})(1 - \mathbf{V}_{ii}) - \mathbf{V}_{ij}\mathbf{V}_{ji}\big](h'(x_j) - y_j) &= \mathbf{V}_{ji}(h(x_i) - y_i) + (1 - \mathbf{V}_{ii})(h(x_j) - y_j) \\
&\quad - [(\mathbf{V}_{ii} + \mathbf{V}_{ij})\mathbf{V}_{ji} + (\mathbf{V}_{jj} + \mathbf{V}_{ji})(1 - \mathbf{V}_{ii})](\bar{h}' - \bar{y}).
\end{aligned}$$

Taking the difference of these last two equations and squaring both sides yields the expression of $c^2_{\mathrm{MP}}(h', x_i, x_j)$ given in the statement of the proposition. □

Given $\bar{h}'$, Proposition 2 and Equation 17 can be used to compute the leave-one-out error of h efficiently, since the coefficients $\mathbf{U}_{ij}$ can be obtained in time $O(m^2)$ from the matrix $(\mathbf{I}+\bar{\mathbf{K}})^{-1}$ already computed to determine h.

Note that by the results of Section 2.3 and the strict convexity of the objective function, h' is uniquely determined and has a closed form. Thus, unless the points x_i and x_j coincide, the expression $[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii})-\mathbf{V}_{ij}\mathbf{V}_{ji}]$ factor of $c^2_{\mathrm{MP}}(h', x_i, x_j)$ cannot be null. Otherwise, the system of linear equations found in the proof is reduced to a single equation and $h'(x_i)$ (or $h'(x_j)$) is not uniquely specified.

For larger values of m, the average value of h over the sample S should not be much different from that of h', thus we can approximate $\bar{h}'$ by $\bar{h}$. Using this approximation, for a sample with distinct points, we can write for $L=$MPRank

$$\mathrm{LOO}(L,S) \approx \frac{1}{m(m-1)} \sum_{i \neq j} \Bigg[\frac{(1-\mathbf{V}_{ii}-\mathbf{V}_{ij})(h(x_j)-y_j)-(1-\mathbf{V}_{ji}-\mathbf{V}_{jj})(h(x_i)-y_i)}{[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii})-\mathbf{V}_{ij}\mathbf{V}_{ji}]} - \frac{[(1-\mathbf{V}_{ii}-\mathbf{V}_{ij})(\mathbf{V}_{jj}+\mathbf{V}_{ji})-(1-\mathbf{V}_{ji}-\mathbf{V}_{jj})(\mathbf{V}_{ii}+\mathbf{V}_{ij})]}{[(1-\mathbf{V}_{jj})(1-\mathbf{V}_{ii})-\mathbf{V}_{ij}\mathbf{V}_{ji}]}(\bar{h}-\bar{y}) \Bigg]^2 .$$

This can be used to determine efficiently the best value of the parameter C based on the leave-one-out error.

Observe that the sum of the entries of each row of $\bar{\mathbf{K}}$ or each row of $\mathbf{K}''$ is zero. Let $\mathbf{M}_1 \in \mathbb{R}^{m\times 1}$ be column matrix with all entries equal to 1. In view of this observation, $\bar{\mathbf{K}}\mathbf{M}_1 = 0$, thus $(\mathbf{I}+\bar{\mathbf{K}})\mathbf{M}_1 = \mathbf{M}_1$, $(\mathbf{I}+\bar{\mathbf{K}})^{-1}\mathbf{M}_1 = \mathbf{M}_1$, and $\mathbf{U}\mathbf{M}_1 = C'\mathbf{K}''(\mathbf{I}+\bar{\mathbf{K}})^{-1}\mathbf{M}_1 = C'\mathbf{K}''\mathbf{M}_1 = 0$. This shows that the sum of the entries of each row of $\mathbf{U}$ is also zero, which yields the following identity for the matrix $\mathbf{V}$:

$$\mathbf{V}_{ij} = \mathbf{U}_{ij} - \frac{1}{m-2}\sum_{k \notin \{i,j\}} \mathbf{U}_{ik} = \mathbf{U}_{ij} + \frac{\mathbf{U}_{ii}+\mathbf{U}_{ij}}{m-2} = \frac{(m-1)\mathbf{U}_{ij}+\mathbf{U}_{ii}}{m-2}. \tag{27}$$

Hence the matrix $\mathbf{V}$ computes

$$\sum_{k=1}^{m} \mathbf{V}_{ik}(y_k-\bar{y}) = \sum_{k=1}^{m} \mathbf{V}_{ik}(y_k-\bar{y}) = \frac{m-1}{m-2}h(x_i). \tag{28}$$

These identities further simplify the expression of matrix $\mathbf{V}$ and its relationship with h.

3 Stability bounds

Bousquet and Elisseeff (2000) and Bousquet and Elisseeff (2002) gave stability bounds for several regression and classification algorithms. This section shows similar stability bounds for ranking and magnitude-preserving ranking algorithms. This also generalizes the results of Agarwal and Niyogi (2005) which were given in the specific case of bi-partite ranking.

The following definitions are natural extensions to the case of cost functions over pairs of those given by Bousquet and Elisseeff (2002).

Definition 1. *A learning algorithm L is said to be uniformly β-stable with respect to the sample S and cost function c if there exists $\beta \geq 0$ such that for all $S \in (X \times \mathbb{R})^m$ and $i \in [1, m]$,*

$$\forall x, x' \in X, \ |c(h_S, x, x') - c(h_{S^{-i}}, x, x')| \leq \beta. \tag{29}$$

Definition 2. *A cost function c is is σ-admissible with respect to a hypothesis set H if there exists $\sigma \geq 0$ such that for all $h, h' \in H$, and for all $x, x' \in X$,*

$$|c(h, x, x') - c(h', x, x')| \leq \sigma(|\Delta h(x')| + |\Delta h(x)|), \tag{30}$$

with $\Delta h = h' - h$.

3.1 Magnitude-preserving regularization algorithms

For a cost function c such as those just defined and a regularization function N, a regularization-based algorithm can be defined as one minimizing the following objective function:

$$F(h, S) = N(h) + C \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} c(h, x_i, x_j), \tag{31}$$

where $C \geq 0$ is a constant determining the trade-off between the emphasis on the regularization term versus the error term. In much of what follows, we will consider the case where the hypothesis set H is a reproducing Hilbert space and where N is the squared norm in a that space, $N(h) = \|h\|_K^2$ for a kernel K, though some of our results can straightforwardly be generalized to the case of an arbitrary convex N. By the reproducing property, for any $h \in H$, $\forall x \in X, \ h(x) = \langle h, K(x, .) \rangle$ and by Cauchy-Schwarz's inequality,

$$\forall x \in X, |h(x)| \leq \|h\|_K \sqrt{K(x, x)}. \tag{32}$$

Assuming that for all $x \in X, K(x, x) \leq \kappa^2$ for some constant $\kappa \geq 0$, the inequality becomes: $\forall x \in X, |h(x)| \leq \kappa \|h\|_K$. With the cost functions previously discussed, the objective function F is then strictly convex and the optimization problem admits a unique solution. In what follows, we will refer to the algorithms minimizing the objective function F with a cost function defined in the previous section as *magnitude-preserving regularization algorithms*.

Lemma 1. *Assume that the hypotheses in H are bounded, that is for all $h \in H$ and $x \in S$, $|h(x) - y_x| \leq M$. Then, the cost functions c^n_{HR}, c^n_{MP}, c^n_{HMP}, and c^n_{SVR} are all σ_n-admissible with $\sigma_1 = 1$, $\sigma_2 = 4M$.*

Proof. We will give the proof in the case of c^n_{MP}, $n = 1, 2$, the other cases can be treated similarly.

By definition of c^1_{MP}, for all $x, x' \in X$,

$$\begin{aligned} |c^1_{\mathrm{MP}}(h', x, x') - c^1_{\mathrm{MP}}(h, x, x')| = \big| &|(h'(x') - h'(x)) - (y_{x'} - y_x)| - \\ &|(h(x') - h(x)) - (y_{x'} - y_x)| \big|. \end{aligned} \tag{33}$$

Using the identity $\big||X'-Y|-|X-Y|\big| \leq |X'-X|$, valid for all $X, X', Y \in \mathbb{R}$, it follows that

$$\begin{aligned} |c^1_{\text{MP}}(h', x, x') - c^1_{\text{MP}}(h, x, x')| &\leq |\Delta h(x') - \Delta h(x)| && (34) \\ &\leq |\Delta h(x')| + |\Delta h(x)|, && (35) \end{aligned}$$

which shows the σ-admissibility of c^1_{MP} with $\sigma = 1$. For c^2_{MP}, for all $x, x' \in X$,

$$\begin{aligned} |c^2_{\text{MP}}(h', x, x') - c^2_{\text{MP}}(h, x, x')| &= ||(h'(x') - h'(x)) - (y_{x'} - y_x)|^2 && (36) \\ &\quad -|(h(x') - h(x)) - (y_{x'} - y_x)|^2| \\ &\leq |\Delta h(x') - \Delta h(x)|(|h'(x') - y_{x'}| + && (37) \\ &\quad |h(x') - y_{x'}| + |h'(x) - y_x| + |h(x) - y_x|) \\ &\leq 4M(|\Delta h(x')| + |\Delta h(x)|), && (38) \end{aligned}$$

which shows the σ-admissibility of c^2_{MP} with $\sigma = 4M$. □

Proposition 3. *Assume that the hypotheses in H are bounded, that is for all $h \in H$ and $x \in S$, $|h(x) - y_x| \leq M$. Then, a magnitude-preserving regularization algorithm as defined above is β-stable with $\beta = \frac{4C\sigma_n^2\kappa^2}{m}$.*

Proof. Fix the cost function to be c, one of the σ_n-admissible cost function previously discussed. Let h_S denote the function minimizing $F(h, S)$ and $h_{S^{-k}}$ the one minimizing $F(h, S^{-k})$. We denote by $\Delta h_S = h_{S^{-k}} - h_S$.

Since the cost function c is convex with respect to $h(x)$ and $h(x')$, $\widehat{R}(h, S)$ is also convex with respect to h and for $t \in [0, 1]$,

$$\widehat{R}(h_S + t\Delta h_S, S^{-k}) - \widehat{R}(h_S, S^{-k}) \leq t\left[\widehat{R}(h_{S^{-k}}, S^{-k}) - \widehat{R}(h_S, S^{-k})\right]. \quad (39)$$

Similarly,

$$\widehat{R}(h_{S^{-k}} - t\Delta h_S, S^{-k}) - \widehat{R}(h_{S^{-k}}, S^{-k}) \leq t\left[\widehat{R}(h_S, S^{-k}) - \widehat{R}(h_{S^{-k}}, S^{-k})\right]. \quad (40)$$

Summing these inequalities yields

$$\widehat{R}(h_S + t\Delta h_S, S^{-k}) - \widehat{R}(h_S, S^{-k}) + \widehat{R}(h_{S^{-k}} - t\Delta h_S, S^{-k}) - \widehat{R}(h_{S^{-k}}, S^{-k}) \leq 0. \quad (41)$$

By definition of h_S and $h_{S^{-k}}$ as functions minimizing the objective functions, for all $t \in [0, 1]$,

$$F(h_S, S) - F(h_S + t\Delta h_S, S) \leq 0 \text{ and } F(h_{S^{-k}}, S^{-k}) - F(h_{S^{-k}} - t\Delta h_S, S^{-k}) \leq 0. \quad (42)$$

Multiplying Inequality 41 by C and summing it with the two Inequalities 42 lead to

$$A + \|h_S\|_K^2 - \|h_S + t\Delta h_S\|_K^2 + \|h_{S^{-k}}\|_K^2 - \|h_{S^{-k}} - t\Delta h_S\|_K^2 \leq 0. \quad (43)$$

with $A = C\left(\widehat{R}(h_S, S) - \widehat{R}(h_S, S^{-k}) + \widehat{R}(h_S + t\Delta h_S, S^{-k}) - \widehat{R}(h_S + t\Delta h_S, S)\right)$. Since

$$A = \tfrac{C}{m^2}\Big[\sum_{i \neq k} c(h_S, x_i, x_k) - c(h_S + t\Delta h_S, x_i, x_k) + \sum_{i \neq k} c(h_S, x_k, x_i) - c(h_S + t\Delta h_S, x_k, x_i)\Big], \tag{44}$$

by the σ_n-admissibility of c,

$$|A| \leq \frac{2Ct\sigma_n}{m^2} \sum_{i \neq k}(|\Delta h_S(x_k)| + |\Delta h_S(x_i)|) \leq \frac{4Ct\sigma_n\kappa}{m}\|\Delta h_S\|_K.$$

Using the fact that $\|h\|_K^2 = \langle h, h\rangle$ for any h, it is not hard to show that

$$\|h_S\|_K^2 - \|h_S + t\Delta h_S\|_K^2 + \|h_{S^{-k}}\|_K^2 - \|h_{S^{-k}} - t\Delta h_S\|_K^2 = 2t(1-t)\|\Delta h_S\|_K^2.$$

In view of this and the inequality for $|A|$, Inequality 43 implies $2t(1-t)\|\Delta h_S\|_K^2 \leq \frac{4Ct\sigma_n\kappa}{m}\|\Delta h_S\|_K$, that is after dividing by t and taking $t \to 0$,

$$\|\Delta h_S\|_K \leq \frac{2C\sigma_n\kappa}{m}. \tag{45}$$

By the σ_n-admissibility of c, for all $x, x' \in X$,

$$\begin{aligned} |c(h_S, x, x') - c(h_{S^{-k}}, x, x')| &\leq \sigma_n(|\Delta h_S(x')| + |\Delta h_S(x)|) & (46)\\ &\leq 2\sigma_n\kappa\|\Delta h_S\|_K & (47)\\ &\leq \frac{4C\sigma_n^2\kappa^2}{m}. & (48)\end{aligned}$$

This shows the β-stability of the algorithm with $\beta = \frac{4C\sigma_n^2\kappa^2}{m}$. □

To shorten the notation, in the absence of ambiguity, we will write in the following $\widehat{R}(h_S)$ instead of $\widehat{R}(h_S, S)$.

Theorem 1. *Let c be any of the cost functions defined in Section 2.1. Let L be a uniformly β-stable algorithm with respect to the sample S and cost function c and let h_S be the hypothesis returned by L. Assume that the hypotheses in H are bounded, that is for all $h \in H$, sample S, and $x \in S$, $|h(x) - y_x| \leq M$. Then, for any $\epsilon > 0$,*

$$\Pr_{S \sim D}\left[|R(h_S) - \widehat{R}(h_S)| > \epsilon + 2\beta\right] \leq 2e^{-\frac{m\epsilon^2}{2(\beta m + (2M)^n)^2}}. \tag{49}$$

Proof. We apply McDiarmid's inequality (McDiarmid, 1998) to $\Phi(S) = R(h_S) - \widehat{R}(h_S, S)$. We will first give a bound on $\mathrm{E}[\Phi(S)]$ and then show that $\Phi(S)$ satisfies the conditions of McDiarmid's inequality.

We will denote by $S^{i,j}$ the sample derived from S by replacing x_i with x_i' and x_j with x_j', with x_i' and x_j' sampled i.i.d. according to D.

Since the sample points in S are drawn in an i.i.d. fashion, for all $i, j \in [1, m]$,

$$\mathrm{E}_S[\widehat{R}(h_S, S)] = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\mathrm{E}[c(h_S, x_i, x_j)] \tag{50}$$

$$= \mathrm{E}_{S\sim D}[c(h_S, x_i, x_j)] \tag{51}$$

$$= \mathrm{E}_{S^{i,j}\sim D}[c(h_{S^{i,j}}, x'_i, x'_j)] \tag{52}$$

$$= \mathrm{E}_{S,x'_i,x'_j\sim D}[c(h_{S^{i,j}}, x'_i, x'_j)]. \tag{53}$$

Note that by definition of $R(h_S)$, $\mathrm{E}_S[R(h_S)] = \mathrm{E}_{S,x'_i,x'_j\sim D}[c(h_S, x'_i, x'_j)]$. Thus, $\mathrm{E}_S[\Phi(S)] = \mathrm{E}_{S,x,x'}[c(h_S, x'_i, x'_j) - c(h_{S^{i,j}}, x'_i, x'_j)]$, and by β-stability (Proposition 3)

$$|\,\mathrm{E}_S[\Phi(S)]| \leq \mathrm{E}_{S,x,x'}[|c(h_S, x'_i, x'_j) - c(h_{S^i}, x'_i, x'_j)|] + \tag{54}$$

$$\mathrm{E}_{S,x,x'}[|c(h_{S^i}, x'_i, x'_j) - c(h_{S^{i,j}}, x'_i, x'_j)|] \tag{55}$$

$$\leq 2\beta. \tag{56}$$

Now,

$$|R(h_S) - R(h_{S^k})| = |\,\mathrm{E}_S[c(h_S, x, x') - c(h_{S^k}, x, x')]| \tag{57}$$

$$\leq \mathrm{E}_S[|c(h_S, x, x') - c(h_{S^k}, x, x')|] \tag{58}$$

$$\leq \beta. \tag{59}$$

For any $x, x' \in X$, $|c(h_S, x_k, x_j) - c(h_{S^k}, x_i, x'_k)| < \mathrm{E}_S[|c(h_{S^k}, x, x') - c(h_{S^k}, x, x')|] \leq (2M)^n$, where $n = 1$ or $n = 2$. Thus, we have

$$|\widehat{R}(h_S) - \widehat{R}(h_S^k)| \leq \frac{1}{m^2}\sum_{i\neq k}\sum_{j\neq k}|c(h_S, x_i, x_j) - c(h_{S^k}, x_i, x_j)| + \tag{60}$$

$$\frac{1}{m^2}\sum_{j=1}^{m}|c(h_S, x_k, x_j) - c(h_{S^k}, x'_k, x_j)| + \tag{61}$$

$$\frac{1}{m^2}\sum_{i=1}^{m}|c(h_S, x_k, x_j) - c(h_{S^k}, x_i, x'_k)| \tag{62}$$

$$\leq \frac{1}{m^2}(m^2\beta) + \frac{m}{m^2}2(2M)^n = \beta + 2(2M)^n/m. \tag{63}$$

Thus,

$$|\Phi(S) - \Phi(S^k)| \leq 2(\beta + (2M)^n/m), \tag{64}$$

and $\Phi(S)$ satisfies the hypotheses of McDiarmid's inequality. □

The following Corollary gives stability bounds for the generalization error of magnitude-preserving regularization algorithms.

Corollary 1. *Let L be a magnitude-preserving regularization algorithm and let c be the corresponding cost function and assume that for all $x \in X, K(x,x) \leq \kappa^2$. Assume that the hypothesis set H is bounded, that is for all $h \in H$, sample S, and $x \in S$, $|h(x) - y_x| \leq M$. Then, with probability at least $1 - \delta$,*

– *for* $n = 1$,

$$R(h_S) \leq \widehat{R}(h_S) + \frac{8\kappa^2 C}{m} + 2(2\kappa^2 C + M)\sqrt{\frac{2}{m} \log \frac{2}{\delta}}; \qquad (65)$$

– *for* $n = 2$,

$$R(h_S) \leq \widehat{R}(h_S) + \frac{128\kappa^2 C M^2}{m} + 4M^2(16\kappa^2 C + 1)\sqrt{\frac{2}{m} \log \frac{2}{\delta}}. \qquad (66)$$

Proof. By Proposition 3, these algorithms are β-stable with $\beta = \frac{4C\sigma_n^2\kappa^2}{m}$. □

These bounds are of the form $R(h_S) \leq \widehat{R}(h_S) + O(\frac{C}{\sqrt{m}})$. Thus, they are effective for values of $C \ll \sqrt{m}$.

4 Experiments

In this section, we report the results of experiments with two of our magnitude-preserving algorithms, MPRank and SVRank.

The algorithms were tested on four publicly available data sets, three of which are commonly used for collaborative filtering: MovieLens, Book-Crossings, and Jester Joke. The fourth data set is the Netflix data. The first three datasets are available from the following URL:

`http://www.grouplens.org/taxonomy/term/14`.

The Netflix data set is available at

`http://www.netflixprize.com/download`.

4.1 MovieLens Dataset

The MovieLens dataset consists of approximately 1M ratings by 6,040 users for 3,900 movies. Ratings are integers in the range of 1 to 5. For each user, a different predictive model is designed. The ratings of that user on the 3,900 movies (not all movies will be rated) form the target values y_i. The other users' ratings of the ith movie form the ith input vector x_i.

We followed the experimental set-up of Freund et al. (1998) and grouped the reviewers according to the number of movies they have reviewed. The groupings were $20 - 40$ movies, $40 - 60$ movies, and $60 - 80$ movies.

Test reviewers were selected among users who had reviewed between 50 and 300 movies. For a given test reviewer, 300 reference reviewers were chosen at random from one of the three groups and their rating were used to form the input vectors. Training was carried out on half of the test reviewer's movie ratings and testing was performed on the other half. The experiment was done for 300 different test reviewers and the average performance recorded. The whole

Table 1. Performance results for MPRank, SVRank, and RankBoost.

Data set	Mean Squared Difference			Mean 1-Norm Difference		
	MPRank	SVRank	RBoost	MPRank	SVRank	RBoost
MovieLens 20-40	2.01 ± 0.02	2.43 ± 0.13	12.88 ± 2.15	1.04 ± 0.05	1.17 ± 0.03	2.59 ± 0.04
MovieLens 40-60	2.02 ± 0.06	2.36 ± 0.16	20.06 ± 2.76	1.04 ± 0.02	1.15 ± 0.07	2.99 ± 0.12
MovieLens 60-80	2.07 ± 0.05	2.66 ± 0.09	21.35 ± 2.71	1.06 ± 0.01	1.24 ± 0.02	3.82 ± 0.23
Jester 20-40	51.34 ± 2.90	55.00 ± 5.14	77.08 ± 17.1	5.08 ± 0.15	5.40 ± 0.20	5.97 ± 0.16
Jester 40-60	46.77 ± 2.03	57.75 ± 5.14	80.00 ± 18.2	4.98 ± 0.13	5.27 ± 0.20	6.18 ± 0.11
Jester 60-80	49.33 ± 3.11	56.06 ± 4.26	88.61 ± 18.6	4.88 ± 0.14	5.25 ± 0.19	6.46 ± 0.20
Netflix Density:32%	1.58 ± 0.04	1.80 ± 0.05	57.5 ± 7.8	0.92 ± 0.01	0.95 ± 0.02	6.48 ± 0.55
Netflix Density:46%	1.55 ± 0.03	1.90 ± 0.06	23.9 ± 2.9	0.95 ± 0.01	1.02 ± 0.02	4.10 ± 0.23
Netflix Density:58%	1.49 ± 0.03	1.93 ± 0.06	12.33 ± 1.47	0.94 ± 0.01	1.06 ± 0.02	3.01 ± 0.15
Books	4.00 ± 3.12	3.64 ± 3.04	7.58 ± 9.95	1.38 ± 0.60	1.32 ± 0.56	1.72 ± 1.05

process was then repeated ten times with a different set of 300 reviewers selected at random. We report mean values and standard deviation for these ten repeated experiments for each of the three groups. Missing review values in the input features were populated with the median review score of the given reference reviewer.

4.2 Jester Joke Dataset

The Jester Joke Recommender System dataset contains 4.1M continuous ratings in the range -10.00 to +10.00 of 100 jokes from 73,496 users. The experiments were set up in the same way as for the MovieLens dataset.

4.3 Netflix Dataset

The Netflix dataset contains more than 100M ratings by 480,000 users for 17,700 movies. Ratings are integers in the range of 1 to 5. We constructed three subsets of the data with different user densities. Subsets were obtained by thresholding against two parameters: the minimum number of movies rated by a user and the minimum of ratings for a movie. Thus, in choosing users for the training and testing set, we only consider those users who have reviewed more than 150, 500, or 1500 movies respectively. Analogously, in selecting the movies that would appear in the subset data, we only consider those movies that have received at least $360, 1200$, or 1800 reviews. The experiments were then set-up in the same way as for the MovieLens dataset. The mean densities of the three subsets (across the ten repetitions) were $32\%, 46\%$ and 58% respectively. Finally, the test raters were selected from a mixture of the three densities.

4.4 Book-Crossing Dataset

The book-crossing dataset contains 1,149,780 ratings for 271,379 books for a group of 278,858 users. The low density of ratings makes predictions very noisy in this task. Thus, we required users to have reviewed at least 200 books, and then only kept books with at least 10 reviews. This left us with a dataset of 89 books and 131 reviewers. For this dataset, each of the 131 reviewers was in turn selected as a test reviewer, and the other 130 reviewers served as input features. The results reported are mean values and standard deviations over these 131 leave-one-out experiments.

4.5 Performance Measures and Results

The performance measures we report correspond to the problem we are solving. The cost function of MPRank is designed to minimize the squared difference between all pairs of target values, hence we report the mean squared difference (MSD) over all pairs in the test set of size m' of a hypothesis h:

$$\frac{1}{m'^2} \sum_{i=1}^{m'} \sum_{j=1}^{m'} \left((h(x_j) - h(x_i)) - (y_j - y_i)\right)^2 . \tag{67}$$

The cost function of SVRank minimizes the absolute value of the difference between all pairs of examples, hence we report the average of the 1-norm difference, M1D:

$$\frac{1}{m'^2} \sum_{i=1}^{m'} \sum_{j=1}^{m'} \left|(h(x_j) - h(x_i)) - (y_j - y_i)\right| . \tag{68}$$

The results for MPRank and SVRank are obtained using Gaussian kernels. The width of the kernel and the other cost function parameters were first optimized on a held-out sample. The performance on their respective cost functions was optimized and the parameters fixed at these values.

The results are reported in Table 1. They demonstrate that the magnitude-preserving algorithms are both successful at minimizing their respective objective. MPRank obtains the best MSD values and the two algorithms obtain comparable M1D values. However, overall, in view of these results and the superior

Table 2. Comparison of MPRank and RankBoost for pairwise misrankings.

Data set	Pairwise Misrankings	
	MPRank	RBoost
MovieLens 40-60	0.471 ± 0.005	0.476 0 ± 0.007
MovieLens 60-80	0.442 ± 0.005	0.463 ± 0.011
Jester 20-40	0.414 ± 0.005	0.479 ± 0.008
Jester 40-60	0.418 ± 0.007	0.432 ± 0.005
Netflix Density:32%	0.433 ± 0.018	0.447 ± 0.027
Netflix Density:46%	0.368 ± 0.014	0.327 ± 0.008
Netflix Density:58%	0.295 ± 0.006	0.318 ± 0.008

computational efficiency of MPRank already pointed out in the previous section, we consider MPRank as the best performing algorithm for such tasks.

To further examine the ranking properties of MPRank we conducted a number of experiments where we compared the pairwise misranking performance of the algorithm to that of RankBoost, an algorithm designed to minimize the number of pairwise misrankings (Rudin et al., 2005). We used the same features for RankBoost as for MPRank that is we used as weak rankers threshold functions over other reviewers' ratings. As for the other algorithms, the parameter of RankBoost, that is the number of boosting rounds required to minimize pairwise misranking was determined on a held-out sample and then fixed at this value.

Table 2 shows a comparison between these two algorithms. It reports the fraction of pairwise misrankings for both algorithms using the same experimental set-up as previously described:

$$\frac{\sum_{i,j=1}^{m'} 1_{y_i > y_j \wedge h(x_i) \leq h(x_j)}}{\sum_{i,j=1}^{m'} 1_{y_i > y_j}}. \tag{69}$$

The results show that the pairwise misranking error of MPRank is comparable to that of RankBoost. This further increases the benefits of MPRank as a ranking algorithm.

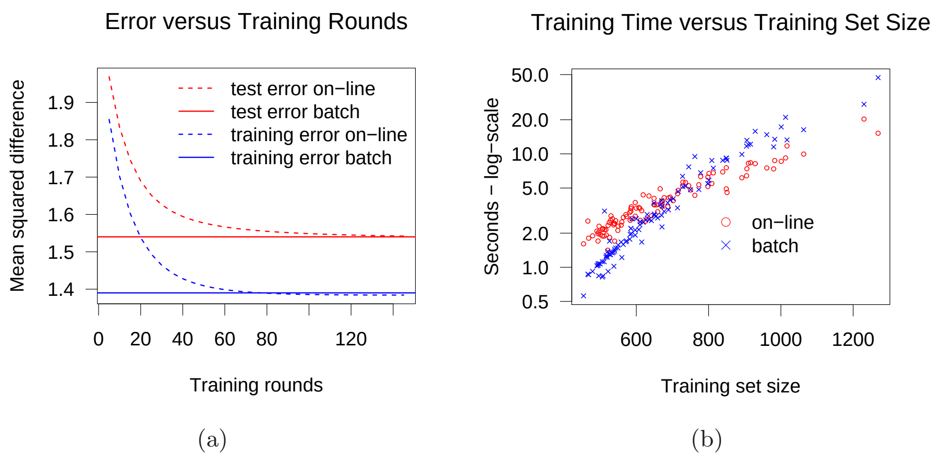

Fig. 1. (a) Convergence of the on-line learning algorithm towards the batch solution. Rounding errors give rise to slightly different solutions. (b) Training time in seconds for the on-line and the batch algorithm. For small training set sizes the batch version is fastest, but for larger training set sizes the on-line version is faster. Eventually the batch version becomes infeasible.

We also tested the performance of RankBoost with respect to MSD and M1D (see Table 1). Naturally, RankBoost is not designed to optimize these performance measure and does not lead to competitive results with respect to MPRank and SVRank on any of the datasets examined.

4.6 On-line Version of MPRank

Using the Netflix data we also experimented with the on-line version of MPRank described in Section 2.5. The main questions we wished to investigate were the convergence rate and CPU time savings of the on-line version with respect to the batch algorithm MPRank (Equation 13). The batch solution requires a matrix inversion and becomes infeasible for large training sets.

Figure 1(a) illustrates the convergence rate for a typical reviewer. In this instance, the training and test sets each consisted of about 700 movies. As can be seen from the plot, the on-line version converges to the batch solution in about 120 rounds, where one round is a full cycle through the training set.

Based on monitoring several convergence plots, we decided on terminating learning in the on-line version of MPRank when consecutive rounds of iterations over the full training set would change the cost function by less than .01 %. Figure 1(b) compares the CPU time for the on-line version of MPRank with the batch solution. For both computations of the CPU times, the time to construct the Gram matrix is excluded. The figure shows that the on-line version is significantly faster for large datasets, which extends the applicability of our algorithms beyond the limits of intractable matrix inversion.

5 Conclusion

We presented several algorithms for magnitude-preserving ranking problems and provided stability bounds for their generalization error. We also reported the results of several experiments on public datasets comparing these algorithms. We presented an on-line version of one of the algorithms and demonstrated its applicability for very large data sets. We view accurate magnitude-preserving ranking as an important problem for improving the quality of modern recommendation and rating systems. An alternative for incorporating the magnitude of preferences in cost functions is to use weighted AUC, where the weights reflect the magnitude of preferences and extend existing algorithms. This however, does not exactly coincide with the objective of preserving the magnitude of preferences.

Acknowledgments

The work of Mehryar Mohri and Ashish Rastogi was partially funded by the New York State Office of Science Technology and Academic Research (NYSTAR). This project was also sponsored in part by the Department of the Army Award Number W81XWH-04-1-0307. The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

Bibliography

Agarwal, S., & Niyogi, P. (2005). Stability and generalization of bipartite ranking algorithms. *Proceedings of COLT 2005*.

Bousquet, O., & Elisseeff, A. (2000). Algorithmic stability and generalization performance. *Advances in Neural Information Processing Systems (NIPS 2000)*.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *J. Mach. Learn. Res.*, *2*, 499–526.

Chu, W., & Keerthi, S. S. (2005). New approaches to support vector ordinal regression. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 145–152). New York, NY, USA: ACM Press.

Cortes, C., & Mohri, M. (2004). AUC Optimization vs. Error Rate Minimization. *Advances in Neural Information Processing Systems (NIPS 2003)*. Vancouver, Canada: MIT Press.

Cortes, C., Mohri, M., & Rastogi, A. (2007). *Magnitude-preserving ranking algorithms* (Technical Report TR-2007-887). Courant Institute of Mathematical Sciences, New York University.

Crammer, K., & Singer, Y. (2001). Pranking with ranking. *Advances in Neural Information Processing Systems (NIPS 2001)*.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. *Proceedings of the 15th International Conference on Machine Learning* (pp. 170–178). Madison, US: Morgan Kaufmann Publishers, San Francisco, US.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In Smola, Bartlett, Schoelkopf and Schuurmans (Eds.), *Advances in large margin classifiers*, 115–132. MIT Press, Cambridge, MA.

Joachims, T. (2002). Evaluating retrieval performance using clickthrough data.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society B*, *42*.

McCullagh, P., & Nelder, J. A. (1983). *Generalized linear models*. Chapman & Hall, London.

McDiarmid, C. (1998). Concentration. *Probabilistic Methods for Algorithmic Discrete Mathematics* (pp. 195–248).

Netflix (2006). Netflix prize. http://www.netflixprize.com/.

Rudin, C., Cortes, C., Mohri, M., & Schapire, R. E. (2005). Margin-Based Ranking Meets Boosting in the Middle. *Proceedings of COLT 2005* (pp. 63–78). Springer, Heidelberg, Germany.

Shashua, A., & Levin, A. (2003). Ranking with large margin principle: Two approaches. *Advances in Neural Information Processing Systems (NIPS 2003)*.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley-Interscience.

Wahba, G. (1990). *Spline models for observational data*. SIAM [Society for Industrial and Applied Mathematics].