Learning to Rank with Nonlinear Monotonic Ensemble

Nikita Spirin[1] and Konstantin Vorontsov[2]

[1] University of Illinois at Urbana-Champaign,
[2] Dorodnicyn Computing Center of the Russian Academy of Sciences,
spirin2@illinois.edu, vokov@forecsys.ru

Abstract. Over the last decade learning to rank (L2R) has gained a lot of attention and many algorithms have been proposed. One of the most successful approach is to build an algorithm following the ensemble principle. Boosting is the key representative of this approach. However, even boosting isn't effective when used to increase the performance of individually strong algorithms, scenario when we want to blend already successful L2R algorithms in order to gain an additional benefit. To address this problem we propose a novel algorithm, based on a theory of *nonlinear monotonic ensembles*, which is able to blend strong base rankers effectively. Specifically, we provide the concept of defect of a set of algorithms that allows to deduce a popular pairwise approach in strict mathematical terms. Using the concept of defect, we formulate an optimization problem and propose a sound method of its solution. Finally, we conduct experiments with real data which shows the effectiveness of the proposed approach.

1 Introduction

Learning to rank (L2R) has become a hot research topic over the last decades. Numerous amount of methods previously applied to regression and classification have been adapted to L2R. Specifically, one can categorize all L2R algorithms into three big categories: pointwise (reduction of L2R to regression) [4, 8]; pairwise (reduction of L2R to classification) [2, 5, 7, 9, 13]; and listwise (direct optimization) [3, 12, 16].

One of the most popular approach that has been applied in all three categories is boosting [5,10]. Thus, weak rankers are trained sequentially and then they are blended in a *linear* composition. It is a common knowledge that boosting allows to combine hundreds of base algorithms and isn't inclined to overfitting [10]. However, boosting isn't effective if we want to build an ensemble from a small set of already strong algorithms. Particularly, one cannot build a boosted ensemble over SVM properly. Different methods was developed to cope with this problem and build small ensembles effectively [6, 14]. We adapted methods from [14] to L2R domain and in this paper we propose a novel algorithm solving the L2R problem within a nonlinear monotonic ensemble framework. Monotonic ensembles have expanded the existing variability of ensemble learning methods and allowed to effectively blend a small set of individually strong algorithms.

For instance, in our experiments the size of an ensemble varied from 4 to 7 base rankers. Moreover, the algorithm in question is built on a strict mathematical foundation, theoretically consistent with the internal structure of L2R problem and allows to induce pairwise approach merely from theoretical constructions instead of heuristic speculations.

1.1 The learning to rank problem

The L2R problem can be formalized as follows. There is an ordered set of ranks $\mathbb{Y} = \{r_1, \ldots, r_K\}$ and a set of queries $Q = \{q_1, \ldots, q_n\}$. A list of documents $D_q = \{d_{q1}, \ldots, d_{q,n(q)}\}$ is associated with each query $q \in Q$, where $n(q)$ is the number of documents associated with the query q. A factor ranking model is admitted, i.e. each query-document pair (q, d), $d \in D_q$ is represented by the vector of features $\boldsymbol{x}_{qd} = \big(f_1(q,d), \ldots, f_m(q,d)\big) \in \mathbb{R}^m$. Thus, the training set is

$$S = \{\boldsymbol{x}_{qd}, r(q,d)\}, \quad q \in Q,\ d \in D_q,$$

where $r(q,d) \in \mathbb{Y}$ is the corresponding correct relevance score for a (q,d) pair. The objective is to build a ranking function $A\colon \mathbb{R}^m \to \mathbb{Y}$ that maximizes a performance measure on the training set and has a good generalization ability.

2 Our Method: MonoRank

2.1 Nonlinear monotonic ensemble: underlying theory

Monotonicity constraints often arise in real world machine learning tasks. For example, we can observe such constraints in a credit scoring task where the objective is to build an algorithm that will classify applicants given their responses to questionnaires (e. g. the bigger annual household income and value of property the more probably a customer will pay a loan back). Generally speaking, monotonicity constraints can arise in any task where the factor model is admitted, and the order on targets is in agreement with the order on ordinal features [11].

Another application of this principle is to impose monotonicity constraints not on features but on base algorithms predictions [6, 14]. It is very natural to construct an ensemble of base predictors according to the following principle: if output of a predictor is higher for an object, provided that outputs of other predictors are the same, then the output of the entire ensemble must be also higher for this object. This implies that the aggregating function is to be monotonic. Obviously, linear blending meets the monotonicity restriction if only all the weights are nonnegative. In this paper we use nonlinear monotonic aggregating functions and argue that monotonicity is a more natural and less restrictive principle than the weighted voting, especially for L2R domain.

Let $\Omega = \mathbb{R}^m$ be an object space of query-document feature vectors, according to general factor ranking model; X and Y be partially ordered sets, B be a set of base algorithms $b\colon \Omega \to X$, X is referred to as an estimation space (predictions of base algorithms) and Y as an output space (labels, responses, relevance

scores). A training set of object–output pairs $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^{\ell}$ from $\Omega \times Y$ and a set of base algorithms $b_1, \ldots, b_p$ induces a sequence of estimation vectors $\{\boldsymbol{u}_k\}_{k=1}^{\ell}$ from X^p, where $\boldsymbol{u}_k = \big(u_k^1 = b_1(\boldsymbol{x}_k), \ldots, u_k^p = b_p(\boldsymbol{x}_k)\big)$.

Let us define an order on X^p: $(u^1, \ldots, u^p) \leq (v^1, \ldots, v^p)$, if $u^i \leq v^i$ for all $i = 1, \ldots, p$. If vectors $\boldsymbol{u}, \boldsymbol{v} \in X^p$ aren't comparable we will denote it as $\boldsymbol{u} \parallel \boldsymbol{v}$. If $\boldsymbol{u} \neq \boldsymbol{v}$ and $\boldsymbol{u} \leq \boldsymbol{v}$, then $\boldsymbol{u} < \boldsymbol{v}$. A map $F\colon X^p \to Y$ is referred to as monotonic, if $\boldsymbol{u} \leq \boldsymbol{v}$ implies $F(\boldsymbol{u}) \leq F(\boldsymbol{v})$ for all $\boldsymbol{u}, \boldsymbol{v} \in X^p$.

Monotonic ensemble of base algorithms $b_1, \ldots, b_p$ with *monotonic aggregating function* F is a map $a\colon \Omega \to Y$ defined as $a(\boldsymbol{x}) = F\big(b_1(\boldsymbol{x}), \ldots, b_p(\boldsymbol{x})\big)$, $\forall \boldsymbol{x} \in \Omega$.

If base algorithms are fixed, then learning of a monotonic function F from data can be stated as a task of monotonic interpolation. Given a sequence of vectors $\{\boldsymbol{u}_k\}_{k=1}^{\ell}$ from X^p and a sequence of targets $\{y_k\}_{k=1}^{\ell}$ from Y, one should build such a monotonic function F that meets the *correctness condition*

$$F(\boldsymbol{u}_k) = y_k, \quad k = 1, \ldots, \ell. \tag{1}$$

Definition 1 *A pair (i, j) is **defective** for the base algorithm b, if $b(\boldsymbol{x}_i) \geq b(\boldsymbol{x}_j)$ and $y_i < y_j$. A set of all defective pairs of b will be denoted as $\mathbb{D}(b)$. A set $\mathbb{D}(b_1, \ldots, b_p) = \mathbb{D}(b_1) \cap \cdots \cap \mathbb{D}(b_p)$ will be called **cumulative defect** of a set of base algorithms $(b_1, \ldots, b_p)$. Similarly, a pair is **clean** if $b(\boldsymbol{x}_i) < b(\boldsymbol{x}_j)$ and $y_i < y_j$. We will use $\mathbb{C}(b)$ and $\mathbb{C}(b_1, \ldots, b_p)$ for that analogously.*

Directly from this definition it can be derived that for $p = 1$ a monotonic function F exists **iff** $\mathbb{D}(b) = \varnothing$. Thus, the number of defective pairs $|\mathbb{D}(b)|$ can play a role of a quality measure for a base algorithm b. By definition, the cumulative defect $\mathbb{D}(b_1, \ldots, b_p)$ consists of those defective pairs on which all base algorithms fail. So, if we build the next base algorithm b_{p+1} so that it yields the right order $b_{p+1}(\boldsymbol{x}_i) < b_{p+1}(\boldsymbol{x}_j)$ on pairs (i, j) from the cumulative defect, then $\mathbb{D}(b_1, \ldots, b_p, b_{p+1}) = \varnothing$ and a monotonic function F satisfying the condition (1) exists. It is worth mentioning that artificial "emptyfication" of defect (for example, by taking two base algorithms with inverted predictions) won't give practically useful results (generalization ability will be poor). Instead, base algorithms should be trained in succession so that they all be individually strong and latter base algorithms corrected predictions of former ones. From practical point of view we need to analyze a defect of an ensemble $\mathbb{D}(F(b_1, \ldots, b_p))$, but not a defect of a set of base algorithms. The next two statements from [14] show the relationship between the sets $\mathbb{D}(F(b_1, \ldots, b_p))$ and $\mathbb{D}(b_1, \ldots, b_p)$.

Lemma 1 *For each p-ary monotonic aggregating function F, $\mathbb{D}(F(b_1, \ldots, b_p)) \supseteq \mathbb{D}(b_1, \ldots, b_p)$.*

Theorem 2 *The cumulative defect $\mathbb{D}(b_1, \ldots, b_p)$ is empty **iff** there exist a monotonic aggregating function F such that $\mathbb{D}(F(b_1, \ldots, b_p)) = \varnothing$.*

From the statements above it can be derived that if we build a set of base algorithms with zero defect, then we will gain a correct, on a training set, algorithm. The stronger statement on convergence is valid [14], i.e. we need only a finite number of steps (base algorithms) in order to build a correct algorithm.

From this, an iterative strategy of building a monotonic ensemble follows. In order to minimize the size p of a composition the choice of the next algorithm b_{p+1} should be guided by the minimization of the number of defective pairs produced by all preceding base algorithms:

$$b_{p+1}(\boldsymbol{x}_i) < b_{p+1}(\boldsymbol{x}_j) : \ (i,j) \in \mathbb{D}(b_1, \ldots, b_p). \tag{2}$$

However, the correctness condition (1) is too restrictive and may result in overfitting (generalization might be poor). The trick is to stop iterations earlier using a stopping criterion like a degradation of quality on a validation set.

Similar to arching and boosting algorithms, we propose to enrich the optimization task (2) with weights in order to add more flexibility to our model.

$$b_{p+1}(\boldsymbol{x}_i) < b_{p+1}(\boldsymbol{x}_j) \ \text{ with } w_{ij} \colon (i,j) \in \mathbb{D}(b_1, \ldots, b_p), \tag{3}$$

where w_{ij} is a weight of a defective pair (i,j). So, the task is to find the heaviest consistent sub-system of inequalities. In Section 2.2 we restate this problem in terms of quality functional and analyze its properties, crucial for L2R.

2.2 The algorithm

Inspired by outstanding performance of monotonic ensembles on classification problems [6,14], we applied the notion of monotonic aggregation to L2R problem and developed an algorithm for it. The algorithm is referred to as MonoRank and the pseudocode is presented in Algorithm 1.

Let us briefly go over all key stages of the algorithm and then we will discuss each stage in detail. First, we train the first base algorithm using the entire training set (line 3). Then we reweigh pairs (line 8) according to the strategies discussed in the Section 2.4. It is worth noting that a monotonic aggregating function isn't needed after the first step, because we have only one base algorithm. Then using updated weights we train the second base algorithm (line 3). Here, all pairs are used but weights already aren't uniform. Having built two base algorithms, we train a monotonic aggregating function in $\mathbb{R}^2$ (lines 4-6), according to the logic we describe in Section 2.5. Then we compute current ensemble performance on a validation set and save it for future reference (line 7). Then the process is repeated: we reweigh pairs based on a current cumulative defect, train the next base algorithm and then fit a monotonic aggregating function. Stopping condition (line 9) is determined by performance of the algorithm on an independent validation set, standard criterion in machine learning research.

The problem (3), restated as a minimization of a quality functional $\mathscr{Q}$ with base algorithms $b_1, \ldots, b_p$ fixed, will look like:

$$\mathscr{Q}(b_1, \ldots, b_p, b_{p+1}) \ = \ \sum_{q \in Q} \sum_{(d,d')} w_{qdd'} \left[b_{p+1}(\boldsymbol{x}_{qd}) \ \geq \ b_{p+1}(\boldsymbol{x}_{qd'}) \right] \ \to \ \min_{b_{p+1}}, \tag{4}$$

where (d,d') are all documents from D_q such that $(qd, qd') \in \mathbb{D}(b_1, \ldots, b_p)$. Note that in L2R task the indices i,j from (3) become qd, qd' respectively, and only those documents d, d' are comparable that corresponds to the same query q.

Algorithm 1 MonoRank pseudocode.

Input: training set $S = \{\boldsymbol{x}_{qd}, r(q,d)\}, q \in Q, d \in D_q$;
δ — number of unsuccessful iterations before stop;
Output: nonlinear monotonic ensemble of rankers $M_T(\boldsymbol{x}_{qd})$ of size T;
initialize weights $w_{qdd'} = 1,\ q \in Q,\ d, d' \in D_q$;
for $t = 1, \ldots, n$ **do**
  train base ranker $b_t(\boldsymbol{x}_{qd})$ using weights $\{w_{qdd'}\}$;
  get predictions $b_t(\boldsymbol{x}_{qd})$ of b_t on a training set S;
  monotonize $\big\{\big(b_1(\boldsymbol{x}_{qd}), \ldots, b_t(\boldsymbol{x}_{qd})\big), r(q,d)\big\}$;
  build a composition $M_t(\boldsymbol{x}_{qd})$ of size t;
  $T = \underset{p:\ p \le t}{\operatorname{argmin}} \mathscr{Q}(M_p)$;
  update $\{w_{qdd'}\}$ using M_t;
  if $t - T \ge \delta$ **then**
    return M_T;

In classification and regression tasks special efforts are to be made to reduce the pairwise criterion $\mathscr{Q}$ to usual pointwise empirical risk [14]. In L2R such tricks are needless as long as base rankers can be learned directly from $\mathscr{Q}$ minimization; this is the reason why monotonic ensembles fit so well to L2R.

2.3 Adaptation of base rankers for usage in monotonic ensemble

Now we will briefly discuss ways to modify existing L2R algorithms with the objective to use them effectively later in a monotonic ensemble. The general idea is to define weights on pairs of query-document feature vectors and hence guide the learning process accordingly. In RankSVM we only have to add weights $w_{qdd'}$ whenever we come across slack variables in the objective function. RankBoost already contains weight distribution over the set of document pairs $q \in Q$, $d, d' \in D_q$, which is by default uniform. Weights $w_{qdd'}$ can also be easily inserted in FRank [13] and RankNet [2].

2.4 Reweighing strategies

Now let us discuss the initialization of weights $w_{qdd'}$ for algorithms described above in order for them to form a strong and diversified set of base rankers. Particularly, below we will describe various reweighing strategies.

1. The weight for a defective pair equals one. The weight for a nondefective pair equals zero. This is the most natural strategy that is induced from the general theory of monotonic aggregating functions and can also be referred to as *the defect minimization principle*. If we train the next base algorithm using only defective pairs, we will minimize the number of base algorithms and reach the state of empty cumulative defect. Thus, according to theorem 2, we will be able to build a monotonic function on predictions of base rankers. However, this approach has a significant shortcoming. If we train our

base algorithms only using defective pairs the generalization ability of the entire algorithm will be poor and hence it won't be practically applicable.

2. The weight for a defective pair is nonzero. The weight for a nondefective pair equals zero. According to our experiments it doesn't allow to gain any rise in quality and only increases the complexity of the model. Our conclusion is in agreement with conclusions made in related research for classification [6].

3. The weight for a defective and clean pair is nonzero. The weight for an incomparable pair is zero. Then the strategy will look like:

$$w_{qdd'} = \begin{cases} w_{\mathbb{D}}(t), & (qd, qd') \in \mathbb{D}(b_1, \ldots, b_p); \\ 1, & (qd, qd') \in \mathbb{C}(b_1, \ldots, b_p); \\ 0, & \text{otherwise}, \end{cases}$$

This is the most successful strategy according to our experiments. Moreover, this strategy combines *the defect minimization principle* and *complete cross-validation minimization* that characterizes the generalization ability of the entire algorithm [15]. In this case the weight for a clean pair equals one. And the weight $w_{\mathbb{D}}(t)$ for a defective pair may depend on iteration $t = 1, \ldots, T$. Particularly, it may increase from iteration to iteration in order to lead the training algorithm to turn out the defect on these pairs. We used $w_{\mathbb{D}}(t) = 2^{t-1}$ in our experiments.

2.5 Monotonic aggregating function

In this section we present the core part of our approach — how to blend base rankers $b_1, \ldots, b_p$ with a nonlinear monotonic aggregating function $F(b_1, \ldots, b_p)$. We will build our algorithm so as to minimize the quality functional induced by the cumulative defect $|\mathbb{D}(b_1, \ldots, b_p)|$. To begin with, we describe general constructions inherent to regression and classification following [14], and then turn to analysis of structures specific for ranking. It is worth noting that we don't impose any constraints on a set of base rankers while learning them. Therefore, according to the theorem 2 monotonic function F might not exist, because a sequence of base predictions might not be monotonic. To cope with this problem we use monotonization based on isotonic regression [1]. Given a nonmonotonic sequence $\{(\boldsymbol{u}_k, y_k)\}_{k=1}^{\ell}$, where $\boldsymbol{u}_k \in \mathbb{R}^p$ is a vector of base algorithms predictions and $y_k \in \mathbb{R}$ is the corresponding target, monotonization algorithm finds $\{y'_k\}_{k=1}^{\ell}$ minimizing $\sum_{i=1}^{\ell}(y'_k - y_k)^2$ subject to $y'_i \le y'_j$ for all (i, j) such that $\boldsymbol{u}_i \le \boldsymbol{u}_j$.

So, let us have a monotonic sequence $\{(\boldsymbol{u}_k, y_k)\}_{k=1}^{\ell}$. The task is to build a monotonic function F that meets the correctness condition (1).

Definition 2 *For any vector $\boldsymbol{u} \in \mathbb{R}^p$ denote its upper and lower set respectively by $M^{\Delta} = \{\boldsymbol{v} \in \mathbb{R}^p \colon \boldsymbol{u} \le \boldsymbol{v}\}$ and $M^{\nabla} = \{\boldsymbol{v} \in \mathbb{R}^p \colon \boldsymbol{v} \le \boldsymbol{u}\}$.*

Consider a continuous function $\mu \colon \mathbb{R}^p \to [0, +\infty)$ nondecreasing by any argument. For example, one can take $\mu(\boldsymbol{x}) = \sum_{i=1}^{p} x^i$ or $\mu(\boldsymbol{x}) = \max\{x^1, \ldots, x^p\}$.

Definition 3 *For any vector* $\boldsymbol{u} \in \mathbb{R}^p$ *denote the distance from* $\boldsymbol{u}$ *to an upper and lower set of a vector* $\boldsymbol{u}_i$ *respectively by*

$$r_i^{\triangle} = \mu\big((u_i^1 - u^1)_+, \ldots, (u_i^p - u^p)_+\big),$$
$$r_i^{\triangledown} = \mu\big((u^1 - u_i^1)_+, \ldots, (u^p - u_i^p)_+\big),$$

where $(z)_+ = z$ *if* $z \geq 0$ *and* $(z)_+ = 0$ *if* $z < 0$.

Now let us define functions $h^{\triangle}(\boldsymbol{u}, \theta)$ and $h^{\triangledown}(\boldsymbol{u}, \theta)$ that estimate the distance from a vector $\boldsymbol{u} \in \mathbb{R}^p$ to a nearest vector from upper and lower sets:

$$h^{\triangle}(\boldsymbol{u}, \theta) = \min_{i:\ y_i > \theta} r_i^{\triangle}(\boldsymbol{u}), \quad h^{\triangledown}(\boldsymbol{u}, \theta) = \min_{i:\ y_i \leq \theta} r_i^{\triangledown}(\boldsymbol{u}).$$

Then, define a relative distance from a vector $\boldsymbol{u}$ to the union of all upper sets:

$$\Phi(\boldsymbol{u}, \theta) = \frac{h^{\triangledown}(\boldsymbol{u}, \theta)}{h^{\triangledown}(\boldsymbol{u}, \theta) + h^{\triangle}(\boldsymbol{u}, \theta)}, \text{ where } \boldsymbol{u} \in \mathbb{R}^p,\ \theta \in \mathbb{R}. \tag{5}$$

The first two functions can be used immediately for two-class classification. Specifically, an object is prescribed to the first class if $h^{\triangle}(\boldsymbol{u}, \theta) > h^{\triangledown}(\boldsymbol{u}, \theta)$ and to the zero class otherwise, where θ can be any number in $(0, 1)$, e. g. $\theta = \frac{1}{2}$. The third function is a regression stair that equals 0 on a union of lower sets, equals 1 on a union of upper sets, and is a continuous, monotone non-decreasing, piecewise bilinear function in between.

Theorem 3 *Let* $\{(\boldsymbol{u}_k, y_k)\}_{k=1}^{\ell}$ *be a monotonic sequence, and a set* $\{y_k\}_{k=1}^{\ell}$ *is sorted in ascending order. Then the function* $F\colon \mathbb{R}^p \to \mathbb{R}$ *defined below is continuous, monotone non-decreasing, and meets the correctness condition* (1).

$$F(\boldsymbol{u}) = y_1 + \sum_{k=1}^{\ell-1} (y_{k+1} - y_k) \Phi(\boldsymbol{u}, y_k)$$

Learning to rank. At first, it is worth noting that due to the structure of the training set comparable documents are only those which are associated with the same query. Another distinction from the above cases is that we don't really need to meet the correctness condition (1) in the case of ranking. The only constraint to meet is to keep the right ordering of documents on the training set.

Provided that documents associated with different queries aren't comparable at all, the quality functional can be rewritten as the sum of functionals, counting defect only for a particular query. We will denote the defect of the entire algorithm with the aggregating function F on a query q as $\mathscr{Q}_q(F)$. Then the optimal aggregating function F must be a solution for a minimization problem

$$\sum_{q \in Q} \mathscr{Q}_q(F) \to \min_F.$$

To give an approximate but computationally efficient solution to this hard problem we propose to use an averaging heuristic. We solve $|Q|$ problems separately:

$$F_q = \arg\min_F \mathscr{Q}_q(F), \quad q \in Q.$$

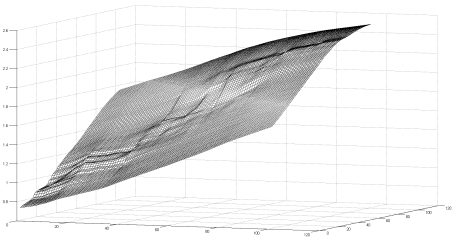

Fig. 1. Monotonic aggregating function for ranking with 2 base algorithms.

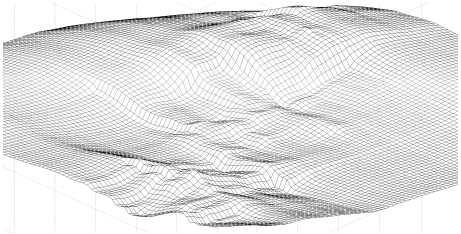

Fig. 2. Texture of the surface of monotonic function from the left figure.

Then we define an aggregating function F by averaging all F'_q:

$$F = \sum_{q \in Q} F'_q. \tag{6}$$

where F'_q is a normalized to $[0, 1]$ function $\frac{F_q}{|D_q|}$. Here we use normalization with the value equal to $|D_q|$ to avoid bias towards queries with the large number of associated documents. Obviously, the function F is monotonic being the sum of monotonic functions. We call a set of base rankers, trained following the logic described above, together with the monotonic aggregating function described in this section as a nonlinear monotonic ensemble for learning to rank.

We provide a few pictures of a monotonic aggregating function for ranking, built according to the theory described above. Due to the large number of queries in a training set and hence due to averaging, the monotonic function from the fig. 1 looks like a plane. However, having changed the scale one can observe a complicated texture of the surface, fig. 2. According to our experiments in Section 3.3 the increase in quality takes place directly thanks to this tiny asperities. It is also interesting to notice that asperities appear only above the diagonal. This can be explained as follows. We use strong base rankers, like RankBoost and RankSVM, that's why their predictions are highly correlated (nevertheless, we can blend them effectively) and lie along the diagonal.

3 Experimental results

3.1 Yahoo! LETOR 2010 Challenge Dataset

This is a dataset provided by Yahoo! company for L2R competition. There are 34815 query-document pairs and 1266 unique queries. Relevance grades are discrete from range $[0, 4]$. Each query-document pair is described by a vector with 575 components. We used 5-fold cross validation to calculate the performance of algorithms. As a base algorithm for MonoRank we used RankBoost. The results are reported in table 1 and in a graphical form in fig. 3.

Table 1. Yahoo! LETOR 2010 results.

Metric	MonoRank	RankBoost	RankSVM
NDCG@1	**0.8149**	0.8029	0.8057
NDCG@3	**0.7783**	0.7424	0.7711
NDCG@5	**0.7754**	0.7692	0.7678
NDCG@10	**0.7973**	0.7935	0.7949

Table 2. OSHUMED LETOR 3.0 results.

Metric	MonoRank	RankBoost	RankSVM
NDCG@1	**0.5231**	0.4632	0.4958
NDCG@3	**0.4602**	0.4555	0.4207
NDCG@5	**0.4500**	**0.4494**	0.4164
NDCG@10	**0.4337**	0.4302	0.4140

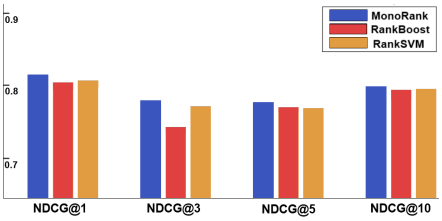

Fig. 3. Performance on Yahoo! LETOR.

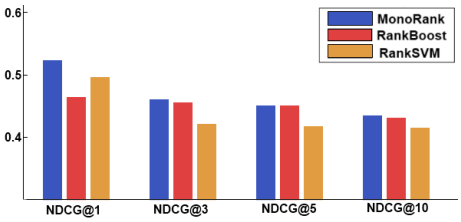

Fig. 4. Performance on OSHUMED.

3.2 OSHUMED LETOR 3.0 Dataset

This is a dataset from LETOR repository[3], prepared by Microsoft Research Asia. There are 16140 query-document pairs and 106 unique queries. Relevance grades are discrete from range $[0, 2]$. There are 36 features. In order to guarantee the consistency of algorithms comparison, we used evaluation scripts from LETOR project. The results are reported in table 2 and in fig. 4.

3.3 Experiment analysis

Now we will briefly discuss the key interesting feature of the monotonic surface we built. Having seen the fig. 1 one might think why we should use so complex construction to blend base rankers. Why couldn't we just use a simple linear combination of base rankers? Of course, this is a reasonable speculation but we have set an experiment to test the hypothesis. We approximated our "wavy" surface with a hyperplane by the least squares method and evaluated the performance on a Yahoo! LETOR 2010 dataset. The results are in table 3.[4]

4 Conclusion

In this paper we proposed a new algorithm for L2R problem, following the ensemble principle. The algorithm is referred to as MonoRank and employs the theory of nonlinear monotonic ensembles in ranking model building. The core

[3] http://research.microsoft.com/en-us/um/beijing/projects/letor/
[4] same results was observed on OSHUMED dataset.

Table 3. Comparison of "wavy" monotonic aggregating function with its linear approximation.

Metric	MonoRank	MonoRank-linearized
NDCG@1	**0.8149**	0.8106
NDCG@3	**0.7783**	0.7735
NDCG@5	**0.7754**	0.7720
NDCG@10	**0.7973**	0.7898

of the algorithm are nonlinear monotonic aggregating functions that enable to blend strong algorithms effectively. The algorithm is based on sound mathematical constructions that are aligned with the popular pairwise approach for L2R. According to computational results MonoRank shows high accuracy in ranking and outperforms existing algorithms, like RankBoost and RankSVM.

References

1. R. Barlow, D. Bartholomew, J. Bremner, and H. Brunk. *Statistical inference under order restrictions; the theory and application of isotonic regression.* 1972.
2. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML 2005*.
3. Z. Cao, T. Qin, T. Y. Liu, M. F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *ICML 2007*, pages 129–136.
4. W. S. Cooper, F. C. Gey, and D. P. Dabney. Probabilistic retrieval based on staged logistic regression. In *SIGIR 1992*.
5. Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 2003.
6. I. S. Guz. Nonlinear monotonic compositions of classifiers. In *MMRO 13*, 2007.
7. T. Joachims. Optimizing search engines using clickthrough data. In *SIGIR 2002*.
8. P. Li, C. J. Burges, and Q. Wu. Learning to rank with nonsmooth cost functions. In *Advances in NIPS'20*, pages 193–200, 2008.
9. T. Qin, T. Y. Liu, W. Lai, X. D. Zhang, D. Wang, and H. Li. Ranking with multiple hyperplanes. In *SIGIR 2007*, pages 279–286.
10. R. E. Schapire. Theoretical views of boosting and applications. In *ALT'99*.
11. J. Sill and Y. Abu-Mostafa. Monotonicity hints. In *Advances in NIPS 9*, 1997.
12. M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: Optimising nonsmooth rank metrics. In *WSDM 2008*.
13. M. F. Tsai, T. Y. Liu, T. Qin, H. H. Chen, and W. Y. Ma. Frank: A ranking method with fidelity loss. In *SIGIR 2007*.
14. K. Vorontsov. Optimization methods for linear and monotone correction in the algebraic approach to the recognition problem. *Comp. Math and Mat. Phys., 2000*.
15. K. Vorontsov. Combinatorial bounds for learning performance. *Doklady Mathematics*, 69(1):145, 2004.
16. M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofirank — maximum margin matrix factorization for collaborative ranking. *Advances in NIPS 20*, 2008.