Journal of Machine Learning Research 8 (2007) 791-812 Submitted 9/05; Revised 12/06; Published 4/07

Concave Learners for Rankboost

Ofer Melnik MELNIK@DIMACS.RUTGERS.EDU
DIMACS
Rutgers University
Piscataway, NJ 08854 USA

Yehuda Vardi VARDI@STAT.RUTGERS.EDU
Cun-Hui Zhang CZHANG@STAT.RUTGERS.EDU
Department of Statistics
Rutgers University
Piscataway, NJ 08854 USA

Editor: Donald Geman

Abstract

Rankboost has been shown to be an effective algorithm for combining ranks. However, its ability to generalize well and not overfit is directly related to the choice of weak learner, in the sense that regularization of the rank function is due to the regularization properties of its weak learners. We present a regularization property called *consistency in preference and confidence* that mathematically translates into monotonic concavity, and describe a new weak ranking learner (MWGR) that generates ranking functions with this property. In experiments combining ranks from multiple face recognition algorithms and an experiment combining text information retrieval systems, rank functions using MWGR proved superior to binary weak learners.

Keywords: rankboost, ranking, convex/concave, regularization

1. Ranking Problems

A ranking problem is a learning problem that involves ranks as the inputs, the outputs or both. An example where ranks are used as inputs is a collaborative filtering application where people are asked to rank movies according to their preferences. In such an application the ranks assigned by different people are combined to generate recommendations. Another type of problem in which ranks are used as inputs are meta-search problems, where the ranks of multiple search engines are combined (Dwork et al., 2001). However, the inputs to a ranking problem are not always ranks. An object recognition ranking problem (Kittler and Roli, 2000) may receive as inputs a graphical representation and output a ranking of the possible objects, sorted by likelihood.

The outputs of a ranking problem may also be ranks. For example, in combining multiple search engines the output are ranks which are a synthesis of the ranks from the individual search engines. A similar meta-recognition problem is the combination of the outputs of multiple face- recognition systems to improve the accuracy of detecting the correct face. While the inputs and outputs of this problem are ranks, the outputs can be simplified to only return the most likely candidate. Another example where the outputs do not need to be complete ranks could be an information retrieval combination task. In such a task the inputs might be ranks of sets of documents with respect to

a particular query by different experts. Again, the outputs could be the complete ranks, or more simply a designation for each document of whether it is relevant or not to the particular query.

2. Rankboost

The rankboost algorithm (Freund et al., 2003) tries to address this variety in ranking problems by maintaining generality in how it regards its inputs and how it applies different loss functions to outputs. The rankboost algorithm works on *instances* which are the discrete items (e.g., movies or faces) that either are ranked as input or are to be ranked as output. To allow a general input mechanism, the inputs to rankboost are called the *ranking features* of an instance, specified as $f_j(x)$ which is the j-th ranking feature of instance x.

While this generality in how inputs are handled is potentially powerful, in the original rankboost paper as well as in this paper, only the case where the inputs are different rankings of the instances is considered. Thus, in this paper, the inputs to rankboost are constituent ranks, denoted as $f_1 \dots f_n$, where $f_j(x') < f_j(x'')$ implies that *instance* x' has a better rank than instance x'' under ranking f_j. For example, in some of the experiments we present, each f_j is the ranking result of a particular face recognition algorithm.

The output of rankboost is a new *ranking function*, $H(x)$, which defines a linear ordering on the instances, that is, $H(x') < H(x'')$ iff x' has a better rank than x''. In rankboost

$$H(x) = \sum_{t=1}^{T} w_t h_t(x), \tag{1}$$

a weighted sum of weak ranking learners, where the $h_t(x)$'s are relatively simple learned functions of the constituent rankings.

To address the variety in possible loss functions of the outputs, in rankboost the desirable properties for the output loss function are specified with a *favor function*, $\Phi : X \times X \to \mathbb{R}$, where X is the space of instances (note that this function has been renamed from "preference function" to avoid confusion with the use of preference in this paper in Section 4). Here $\Phi(x',x'') > 0$ means that x'' should be better ranked than x' for a given query. It is an anti-symmetric function, that is, $\Phi(x',x'') = -\Phi(x'',x')$ and $\Phi(x,x) = 0$, which avoids loops where two instances should both be ranked better than the other. Also $\Phi(x',x'') = 0$ when there is no favor in how two instances should be relatively ranked. For example, as described in Section 6.1, for the face recognition combination problem described above the favor function can be used to specify that the correct identity should be given a better rank than all other identities, while zeroing all other entries in the favor function, giving no favor in how incorrect identities are ranked between them. In a similar fashion for the information retrieval combination task mentioned above, the favor function can be specified such that all relevant documents should be better ranked than irrelevant documents, without specifying favor for the ordering between relevant documents and the ordering between irrelevant documents (Section 7.1).

Rankboost is shown in Algorithm 1 as described in Freund et al. (2003). It uses the favor function to specify an initial weight function over instance pair orderings:

$$D(x',x'') = c \cdot \max\left(0, \Phi\left(x',x''\right)\right), \tag{2}$$

where $c = \left[\sum_{x',x''} \max\left(0,\Phi\left(x',x''\right)\right)\right]^{-1}$. The algorithm itself is very similar to adaboost (Freund and Schapire, 1996). At each iteration the algorithm selects the weak learner that best maximizes a

Algorithm 1 rankboost algorithm for generating a ranking function.

Input: constituent ranks, a favor function, and a class of weak learners with outputs between 0 and 1 and an appropriate training algorithm.
Output: ranking function $H(x)$ (Eq. 1)
Initialize $D^{(1)} = D$ (Eq. 2)
for $t = 1 \ldots T$ **do**
  Find weak learner, h_t, that maximizes $r(h) = \sum_{x',x''} D^{(t)}(x',x'')(h(x') - h(x''))$ (Eq. 4).
  Choose $w_t = 0.5 \ln\left((1 + r(h_t)) / (1 - r(h_t))\right)$. (Eq. 5)
  Update:
  $D^{(t+1)}(x',x'') = D^{(t)}(x',x') \exp\left(w_t \left(h_t(x') - h_t(x'')\right)\right) / Z_t$
  where Z_t is chosen to make $\sum_{x',x''} D^{(t+1)}(x',x'') = 1$
end for

rank function's utility, $r(h)$, (Eq. 4). It then assigns it a weight in proportion to its performance and adds the weighted weak learner to the ranking function, $H(x)$. After which the weight function D is adjusted to reflect the impact of the weak learner. Freund et al. (2003) prove that the rankboost algorithm iteratively minimizes the ranking loss function, a measure of the quantity of misranked pairs:

$$\sum_{x',x''} D\left(x',x''\right) I\left[H\left(x'\right) \leq H\left(x''\right)\right]$$

where I is the indicator function (1 if true, 0 otherwise) and $H(x)$ is a ranking function output by rankboost. The rankboost paper (Freund et al., 2003) uses *binary weak learners* of the following functional form:

$$h(x) = \begin{cases} 1 & if\ f_j(x) > \theta, \\ 0 & if\ f_j(x) \leq \theta, \\ q_{def} & if\ f_j(x)\ undefined. \end{cases} \tag{3}$$

Each binary weak learner, h, operates on a single f_j (ranking feature), giving a binary output of 0 or 1 depending on whether the instance's rank is smaller than a learned parameter, θ. Note that function $h(x)$ in (3), which is dependent on only one $f_j(x)$ with a fixed j, is monotonic increasing but not convex or concave. If there is no rank for the instance then it returns a prespecified q_{def} value.

3. Rankboost, the Weak Learner and Regularization

While rankboost inherits the accuracy of adaboost and has been shown to be very successful in practice, in two important ways it is very different from adaboost and similar classifier leveraging algorithms (Meir and Ratsch, 2003). The first is the choice of the weak learner, h. In adaboost the weak learner is expected to minimize the weighted empirical classification error:

$$\sum_{i=1}^{N} d^{(t)}(i) I\left[y_i \neq h\left(z_i\right)\right],$$

where y_i is the class label, I is the indicator function and $d^{(t)}$ is a weighting over training samples. This is a standard function to minimize in classification with many possible types of algorithms to

choose from as possible weak learners. In contrast the weak ranking learner for rankboost (with outputs in $[0,1]$) needs to maximize the following rank utility function:

$$r = r(h) = \sum_{x',x''} D^{(t)}(x',x'')(h(x') - h(x'')), \tag{4}$$

where $D^{(t)}$ is the weight function over pairs of instances. As Eq. 4 contains the term $h(x') - h(x'')$, short of linear learners this equation can not be concave in h or easily approximated by a concave function. Therefore the weak learner needs to be optimized either by heuristics or by brute force, which limits the choice of h. It is not surprising that the original rankboost paper only included one type of weak learner, a binary threshold function (Eq. 3) that was tuned using brute force.

The second difference between rankboost and adaboost also concerns the weak ranking learner. One feature of boosting that has sparked a great deal of interesting research is the algorithm's ability to avoid overfitting for low noise classification problems (with modifications to higher noise problems), see Meir and Ratsch (2003) for a survey. In contrast for rankboost it is only by limiting the type of underlying weak learners that overfitting is avoided. In their paper, Freund et al. (2003) show that not using weak ranking learners with cumulative positive coefficients leads to overfitting and poor performance quite quickly. Therefore, choosing a weak learner that regularizes the ranking function, the output of rankboost, is very important for achieving accuracy and avoiding overfitting.

It is clear from the above discussion that the choice of a weak ranking learner for rankboost is important and non trivial. First, the learner must be efficiently tunable with respect to Eq. 4, typically limiting its complexity. Second, the learner itself must demonstrate regularization properties that are appropriate for ranking functions.

In this paper we present a new weak ranking learner that enforces *consistency in preference and confidence* for the ranking function by being monotonic and concave. We start with a discussion of these regularization properties, theoretically justify them, and show what they mean in terms of the final ranking function. Then we present a new learner, *Minimum Weighted Group Ranks* (MWGR), that satisfies these properties and can be readily learned. This learner is tested and compared with the binary learner of rankboost on combining multiple face recognition systems from the FERET study (Phillips et al., 2000) and on an information retrieval combination task from TREC (Voorhees and Harman, 2001).

4. Regularizing Ranking Functions with Consistency in Preference and Confidence

In this paper, as Freund et al. (2003), we consider ranking functions $H(x)$ which depend on x only through the values of the ranking features, $y_j = f_j(x)$, for that instance, so that $H(x) = G(f_1(x), \ldots, f_n(x))$, for certain functions $G(y_1, \ldots, y_n) = G(\mathbf{y})$. Here, we assume that the $f_j(x)$ is an actual rank assigned to an instance, x, by the j-th ranker. Note that if the original data are numerical scores then they can easily be converted to rankings. Freund et al. (2003) make a strong case for conversion of raw measures to relative orderings (rankings) over combining measures directly, arguing that it is more general and avoids the semantics of particular measures.

As the y_j's are ranks instead of points in a general space, care should be taken as to the functional form of G. A great deal of literature in social choice theory (Arrow, 1951) revolves around the properties of various rank combination strategies that try to achieve fair rankings. In this machine learning case our motivations are different. Fairness is not the goal; the goal is to improve the

accuracy or performance of the ranking function. Thus, regularization, by functionally constraining G, is used to confer information on how to interpret ranks in order to ultimately improve accuracy.

Freund et al. (2003) imposed the regularization principle of *monotonicity* on the ranking function. Monotonicity encompasses a belief that for individual features a smaller rank is always considered better than a bigger rank. It means that for every two rank vectors, $\mathbf{a}$ and $\mathbf{b}$, if $a_j \leq b_j$, $j = 1,\ldots,n$ then $G(\mathbf{a}) \leq G(\mathbf{b})$. Monotonicity was enforced by requiring that the coefficients, w_t in Eq. 1, combining the binary weak learners (Eq. 3) were cumulatively positive. As shown by Freund et al. (2003), without enforcing monotonicity the rankboost algorithm quickly overfits.

In this section we present another regularization principle, consistency in preference and confidence (which includes monotonicity). A ranking function with this regularization property should be consistent in its preference for the individual rankers and also in how it captures their confidence. The following example illustrates these two concepts.

4.1 Grocer Example

A grocer needs to make 2 decisions, to decide between stocking oat bran vs. granola and to decide between stocking turnips vs. radishes. The grocer asks her consultants to express their opinion about stocking each item, and based on their responses makes her 2 decisions.

First of all, in either problem, the grocer will adopt the opinion of her consultants if they all agree with each other, that is, they all prefer granola over oat bran in the first problem.

Lets assume the grocer considered the first problem and chose granola over oat bran. What this implies is that the grocer adopted the opinions of the consultants that preferred granola over oat bran.

Now consider the turnips vs. radishes decision. Lets say that the same consultants that liked granola more also liked radishes more (and the same ones that like oat bran more like turnips more). Also, for this decision the radish lovers actually feel more confident in their choice than they did for granola, while the turnip lovers are more unsure than they were for oat bran.

Then for this second choice, if the grocer is consistent in her method of combining the opinions of her consultants, we would expect her to pick radishes over the turnips. In addition, we would expect her to be more confident in this second decision as a reflection of the consultants relative confidence. The above properties of preference and confidence imply that preference should be applied consistently across different problems and confidence should reflect the relative confidences of the individual consultants.

4.2 The General Principle of Consistency in Preference and Confidence

To generalize the above grocer's problem consider the problem of combining the opinions of a panel of consultants on several multiple-choice decision problems. Suppose for each decision problem each consultant provides his preference as one of the possible choices and a qualitative measure of the confidence level for his preference. The consistency principle in preference and confidence holds if the combiner always agrees with at least one of the consultants in each decision problem and that the following condition holds for any pair of decision problems.

Definition 1. Consistency principle for a pair of decision problems: *Suppose that for the first decision problem, the combiner adopts the preference of a subset A of consultants in the sense that A is the (non empty) set of all consultants whose preference is identical to that of the combiner.*

Suppose that for the second decision problem, the preference of the consultants in A is again identical. Suppose further that compared with the confidence level for the first decision problem, each consultant in A has higher or the same confidence level for his preference in the second problem, while each consultant with a different preference than A for the second problem has lower or the same confidence level. Then, the preference of the combiner for the second problem is identical to that of the consultants in A, and the confidence level of the combiner for the second problem is at least as high as his confidence level for the first problem.

Let B be the set of consultants whose preferences are different from that of the combiner in the first decision problem, that is, $B = A^c$. If some members of B switch sides or have no preference in the second problem, the combiner would be even more confident about the adoption of the preference of A in the second problem regardless of the confidence of those who switch sides. Thus, Definition 1 requires only those against the preference of A in the second problem (necessarily a subset of B since members of A act in unison in both problems) to have lower or equal confidence in the second problem, instead of all members of B. This is taken into consideration in Theorem 1 below.

Note that while preference for individual consultants is specified within each decision problem, two decision problems are needed to compare the qualitative measure of confidence. However, comparison of confidence is not always possible (it is a partial ordering). In particular, the level of confidence between different experts may not be comparable, and the levels of confidence of a given expert (or the combiner) for different decision problems are not always comparable.

4.3 Confidence for Ranks and Ranking Functions

In order to apply consistency to rank combination we need to specify what we mean by more or less confidence. For ranks we make the assumption that a constant change of ranks requires more confidence for low ranks than high ranks. For example, we would expect the difference between ranks of 1 and 3 to represent a more significant distinction on the part of a ranker than would for example the difference between ranks 34 and 36 which may be almost arbitrarily assigned. Specifically we make the following definition:

Definition 2. Preference and confidence for univariate rank values *For any pair of rank values $\{r, r'\} \subset \mathbf{R}$ with $r < r'$, by monotonicity r is preferred. For any two pairs of rank values $\{r, r'\}$ and $\{r'', r'''\}$ with $r < r'$ and $r'' \leq r'''$, the confidence level for $\{r, r'\}$is higher than that of $\{r'', r'''\}$ if*

$$r \leq r'',\ r''' - r'' \leq r' - r$$

with at least one inequality. The pair$\{r, r'\}$provides no preference if $r = r'$. Likewise, if either $r' - r = r''' - r'' = 0$ or $r'' - r = r''' - r' = 0$, the two pairs $\{r, r'\}$ and $\{r'', r'''\}$are defined to have the same level of confidence.

In this definition confidence between pairs of ranks can only be compared if the pair with the lowest rank has a gap at least as large as the other pair. Thus, we are actually comparing two numbers, that is, we are doing a vector comparison. As such, this comparison does not cover all pairs of ranks and applies only a partial ordering on rank pairs. As a regularization principle, a partial ordering is desirable since it does not overly constrain the ranking function and allows for flexibility.

As the rank function, G, is a univariate function, we apply the same definitions of preference and confidence to it. That is, for a ranking function G and for any fixed rank vectors, $\mathbf{y}$ and $\mathbf{y}'$, the preferred rank vector is the one with smaller G, that is, $\mathbf{y}$ is preferred iff $G(\mathbf{y}) < G(\mathbf{y}')$. For confidence again we need to consider pairs of rank vectors, $\{\mathbf{y},\mathbf{y}'\}$ and $\{\mathbf{y}'',\mathbf{y}'''\}$, with $G(\mathbf{y}) \leq G(\mathbf{y}')$ and $G(\mathbf{y}'') \leq G(\mathbf{y}''')$. If $G(\mathbf{y}) \leq G(\mathbf{y}'')$ and $G(\mathbf{y}') - G(\mathbf{y}) \geq G(\mathbf{y}''') - G(\mathbf{y}'')$ then we say that the first decision, between $\mathbf{y}$and $\mathbf{y}'$, shows equal or more confidence than the second decision between $\mathbf{y}''$ and $\mathbf{y}'''$.

This numerical method of capturing confidence and preference in ranks, $\mathbf{y}$, and ranking functions, G, allows us to apply Definition 1. Specifically, for a pair of binary decisions the confidence of a consistent ranking function increases for the second decision if in the second decision the confidence of the agreeing rank values are comparable and increase and the confidence of the disagreeing rank values are comparable and decrease, with the exception of those that switched sides.

4.4 Three-Point Consistency in Preference and Confidence for Combining Ranks

As described, the consistency property is over two binary decision problems. In this section we consider ranking functions, G, that have consistency in preference and confidence for all pairs of binary decision problems involving three rank vectors $\mathbf{y}$, $\mathbf{y}'$ and $\mathbf{y}''$. We show that such functions have specific mathematical properties under this three-point consistency principle in preference and confidence for ranks..

Theorem 1: *For a ranking function G whose domain is convex, three-point consistency in preference and confidence holds for G iff $G(\mathbf{y})$ is nondecreasing in individual components of $\mathbf{y}$ and is jointly concave in $\mathbf{y}$.*

Proof of Necessity: Assume that the ranking function G exhibits three-point consistency in preference and confidence for any three rank vectors.

If $\mathbf{y} \leq \mathbf{y}'$ component wise, then the individual components of $\mathbf{y}$ all agree in their preference to $\mathbf{y}$, so that $G(\mathbf{y}) \leq G(\mathbf{y}')$ by the consistency of G in preference. This implies the monotonicity of G in $\mathbf{y}$.

We pick three points $\mathbf{y}$, $\mathbf{y}-\mathbf{a}$ and $\mathbf{y}+\mathbf{a}$, such that all points are rank vectors. Consider the pair of binary comparison problems, $\mathbf{y}$ vs. $\mathbf{y}+\mathbf{a}$ and $\mathbf{y}-\mathbf{a}$ vs. $\mathbf{y}$. Assume that $G(\mathbf{y}) < G(\mathbf{y}+\mathbf{a})$. These three points are comparable by Definition 2 ($r''' - r'' = r' - r$ for all components in the rank vectors). Since the agreeing rankers, $A = \{ j \,|\, y_j < y_j + a_j \}$, in the $\mathbf{y}$ vs. $\mathbf{y}+\mathbf{a}$ comparison are greater than in the $\mathbf{y}-\mathbf{a}$ vs. $\mathbf{y}$ comparison, and the disagreeing ranks, outside A, are smaller, then by the consistency of the combiner in preference and confidence (definition 1), $G(\mathbf{y}-\mathbf{a}) \leq G(\mathbf{y})$ and $G(\mathbf{y}) - G(\mathbf{y}-\mathbf{a}) \geq G(\mathbf{y}+\mathbf{a}) - G(\mathbf{y})$.

A function G is concave in a convex domain iff $2G(\mathbf{y}) \geq G(\mathbf{y}-\mathbf{a}) + G(\mathbf{y}+\mathbf{a})$, for every $\mathbf{y}$ and $\mathbf{a}$ with $\mathbf{y} \pm \mathbf{a}$ in its domain. To verify this inequality for a particular $\mathbf{y}$ and $\mathbf{a}$, we must have $G(\mathbf{y}+\mathbf{a}) > G(\mathbf{y})$, $G(\mathbf{y}-\mathbf{a}) > G(\mathbf{y})$ or the third case of $G(\mathbf{y}) \geq \max\{G(\mathbf{y}+\mathbf{a}), G(\mathbf{y}-\mathbf{a})\}$. We have already proven that the consistency properties of preference and confidence imply $G(\mathbf{y}) - G(\mathbf{y}-\mathbf{a}) \geq G(\mathbf{y}+\mathbf{a}) - G(\mathbf{y})$ in the first case. By symmetry this requirement also must hold for $-\mathbf{a}$, so that $G(\mathbf{y}) - G(\mathbf{y}+\mathbf{a}) \geq G(\mathbf{y}-\mathbf{a}) - G(\mathbf{y})$ in the second case. This completes the necessity part of the proof since $2G(\mathbf{y}) \geq G(\mathbf{y}-\mathbf{a}) + G(\mathbf{y}+\mathbf{a})$ automatically holds in the third case.

Proof of Sufficiency: Assume the ranking function G is nondecreasing in individual components of $\mathbf{y}$ and jointly concave in $\mathbf{y}$. We need to prove consistency in preference and confidence for a pair

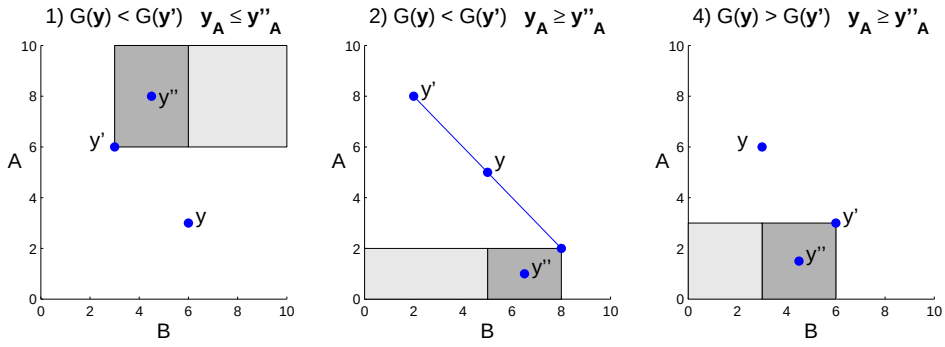

Figure 1: Consider two decision problems in two dimensions involving three points, $\mathbf{y}$, $\mathbf{y}'$ and $\mathbf{y}''$, where the first decision problem is to choose between $\mathbf{y}$ and $\mathbf{y}'$ and the second problem is to choose between $\mathbf{y}$ and $\mathbf{y}''$. The cases where we expect consistency in preference and confidence (Definitions 1 and 2) can be enumerated by the result of the first decision problem and relative location of $\mathbf{y}''$. The darker gray box is the location where B has lesser or equal confidence in the second problem, and the lighter gray box is where B switches sides.

of decision problems involving three rank vectors $\mathbf{y}$, $\mathbf{y}'$ and $\mathbf{y}''$. Without loss of generality, suppose that the first problem is to choose between $\mathbf{y}$ and $\mathbf{y}'$ and the second problem is to choose between $\mathbf{y}$ and $\mathbf{y}''$. We break the proof up by the results of the comparison between $\mathbf{y}$ vs. $\mathbf{y}'$and the relative location of the third ranking vector $\mathbf{y}''$ that satisfies the preference and confidence requirements. If $G(y) = G(y') = G(y'')$, the combiner has equal confidence in the two decision problems by definition 2, so that the consistency principle holds automatically. Thus, we only need to consider the cases where $G(y) \neq G(y')$. Figure 1 illustrates three of these cases two dimensionally, where there is a single agreeing component A, the y-axis, and a single disagreeing component B, the x-axis.

Let A be the set of agreeing indices, $A = \left\{ j : \text{sgn}(y'_j - y_j) = \text{sgn}\left(G(y') - G(y)\right) \neq 0 \right\}$, and $B = A^c$ the set of disagreeing indices. We use the notation $\mathbf{y_A} = (y_j,\, j \in A)$ to describe corresponding subvectors. Also, when used, vector inequalities are component wise.

Case 1: $G(\mathbf{y}) < G(\mathbf{y}')$ **and** $\mathbf{y_A} \leq \mathbf{y''_A}$

In the first decision problem, as G agrees with A and disagrees with B

$$\mathbf{y_A} < \mathbf{y'_A}, \mathbf{y_B} \geq \mathbf{y'_B}.$$

We also know that $A \neq \emptyset$ since otherwise $\mathbf{y} \geq \mathbf{y}'$and therefore by monotonicity $G(\mathbf{y}) \geq G(\mathbf{y}')$, which is a contradiction.

For the second decision problem we consider the values of $\mathbf{y}''$ which are consistent with the confidence assumption (Definitions 1 and 2), that is, agreeing with more confidence along the A indices and disagreeing with less confidence or switching sides along the B indices. As $\mathbf{y_A} \leq \mathbf{y''_A}$, either by the confidence relationship or by switching sides (see Figure 1) these $\mathbf{y}''$ values satisfy

$$\mathbf{y''_A} - \mathbf{y_A} \geq \mathbf{y'_A} - \mathbf{y_A}, \quad \mathbf{y_B} - \mathbf{y''_B} \leq \mathbf{y_B} - \mathbf{y'_B}$$

which implies $\mathbf{y}' \leq \mathbf{y}''$, and therefore $G(\mathbf{y}') \leq G(\mathbf{y}'')$ by monotonicity. Thus, $G(\mathbf{y}) < G(\mathbf{y}') \leq G(\mathbf{y}'')$, which implies that in the second decision problem of choosing between $\mathbf{y}$ and $\mathbf{y}''$, the preference of G is the same as that of A (i.e., $\mathbf{y}$ since $\mathbf{y_A} \leq \mathbf{y''_A}$) and the confidence of G is at least as high as the first decision problem.

Case 2: $G(\mathbf{y}) < G(\mathbf{y}')$ **and** $\mathbf{y_A} \geq \mathbf{y''_A}$

Since in this case $\mathbf{y_A} \geq \mathbf{y''_A}$, then either by the confidence relationship or by switching sides $\mathbf{y}''$ satisfies

$$\mathbf{y_A} - \mathbf{y''_A} \geq \mathbf{y'_A} - \mathbf{y_A}, \quad \mathbf{y''_B} - \mathbf{y_B} \leq \mathbf{y_B} - \mathbf{y'_B}.$$

This implies that $\mathbf{y}' + \mathbf{y}'' \leq 2\mathbf{y}$, which means that

$$G(\mathbf{y}') + G(\mathbf{y}'') \leq 2G\left(\left(\mathbf{y}' + \mathbf{y}''\right)/2\right) \leq 2G(\mathbf{y})$$

by the concavity and monotonicity of G. Thus, $G(\mathbf{y}) - G(\mathbf{y}'') \geq G(\mathbf{y}') - G(\mathbf{y})$ and since $G(\mathbf{y}') > G(\mathbf{y})$, we have that $G(\mathbf{y}) - G(\mathbf{y}'') > 0$ and thus $G(\mathbf{y}) > G(\mathbf{y}'')$. Therefore we see that the preference in G for the two decision problems is the same as A (with $\mathbf{y}$ in the first problem and with $\mathbf{y}''$ in the second problem) and the confidence is no smaller for the second comparison.

Case 3: $G(\mathbf{y}) > G(\mathbf{y}')$ **and** $\mathbf{y_A} \leq \mathbf{y''_A}$

Since $\mathbf{y}'$ is preferred in the first decision problem we have

$$\mathbf{y_A} > \mathbf{y'_A}, \mathbf{y_B} \leq \mathbf{y'_B}.$$

For the confidence assumption to hold, that is, greater confidence for A in the second problem (Definition 2), the smaller ranks in the second problem have to be smaller than the smaller ranks in the first problem. But with $\mathbf{y_A} \geq \mathbf{y'_A}$ and $\mathbf{y_A} \leq \mathbf{y''_A}$ that can only be if $\mathbf{y_A} = \mathbf{y'_A}$, which leads to $\mathbf{y} \leq \mathbf{y}'$, and by monotonicity implies $G(\mathbf{y}) \leq G(\mathbf{y}')$. However that is a contradiction to $G(\mathbf{y}) > G(\mathbf{y}')$, and therefore this is not a viable case for comparing confidence.

Case 4: $G(\mathbf{y}) > G(\mathbf{y}')$ **and** $\mathbf{y_A} \geq \mathbf{y''_A}$

Since in this case $\mathbf{y_A} \geq \mathbf{y''_A}$, then either by the confidence relationship or by switching sides $\mathbf{y}''$ satisfies

$$\mathbf{y_A} - \mathbf{y''_A} \geq \mathbf{y'_A} - \mathbf{y_A}, \quad \mathbf{y''_B} - \mathbf{y_B} \leq \mathbf{y_B} - \mathbf{y'_B},$$

which means that $\mathbf{y}'' \leq \mathbf{y}'$. Thus, by the monotonicity of G, $G(\mathbf{y}'') \leq G(\mathbf{y}')$. Since $G(\mathbf{y}') < G(\mathbf{y})$ the preference in the second decision problem is also with A (i.e., $\mathbf{y}''$) and the confidence is at least as high as the first decision problem.

4.5 Applying Regularization to Rankboost

It follows from the above theorem that to have consistency in preference and confidence we desire ranking functions that are monotonic and concave. In rankboost $H(x) = \sum w_t h_t(x)$ (Eq. 1). To make H an increasing and concave function of constituent rankings $y_j = f_j(x)$ we need to constrain the weak ranking learners. If the learners themselves are monotonically increasing and concave functions of y_j, then linearly combining them with positive weights will give an H that is also an increasing and concave function of y_j.

In this paper, we apply the "third method" (Freund et al., 2003) to setting a w_t weight value. That is, weak learners are selected on their ability to maximize r from Eq. 4 and then

$$w_t = 0.5 \ln \left((1 + r_{\max}) / (1 - r_{\max}) \right). \tag{5}$$

Therefore, using monotonic and concave weak learners we select only ones that rankboost gives a positive r value to, which renders a positive w_t weight. If no r values are positive the rankboost algorithm stops.

We mention that a ranking function can be construed as the negative of a score function, that is, $G(\mathbf{y}) = -S(\mathbf{y})$. For score functions these regularization properties become monotonically decreasing, and convex (Melnik et al., 2004).

5. Minimum Weighted Group Ranks

The functional structure of the new learner we propose is

$$h(y) = \min \{ \gamma_1 y_1, \ldots, \gamma_n y_n, 1 \}, \tag{6}$$

where $\mathbf{y} = (y_1, \ldots, y_n) = (f_1(x), \ldots, f_n(x))$ the vector of ranking features, and the γ_j are learned positive coefficients. Note that the function's range is in $[0, 1]$ due to the 1 term in the min function.

Using rankings as our features, the learner function (Eq. 6) is monotonically increasing. It is also a concave function in $\mathbf{y}$. Thus if these learners are linearly combined with positive weights, the resulting ranking function, H, will have three-point consistency in confidence and preference.

To gain some intuition, the functional form of the learner is related to the Highest Rank combination method (Ho, 1992) that assigns combined ranks as the best rank each class receives from any ranker. This is a confidence based approach, as Highest Rank bets on the classifier that is most confident, the one giving the best rank. As such, a single learner can potentially be error prone. But as we combine many of these learners during boosting, it becomes more powerful, allowing the confidence of different classifiers to be weighed with preference for their potential accuracy.

5.1 Learning

At each boosting iteration, rather than selecting from all possible weak learners of form (Eq. 6), we limit our choices by building new weak learners out of the ones that have been previously trained. Let $\mathcal{F} = \{ f_1(x), \ldots, f_n(x) \}$ be the set of ranking features. Recall that in rankboost $H(x) = \sum_t w_t h_t(x)$, where $h_t(x) = \min \left\{ \gamma_1^{(t)} f_1(x), \ldots, \gamma_n^{(t)} f_n(x), 1 \right\}$ and $\gamma_j^{(t)}$ are learned coefficients. We set

$$\widetilde{\mathcal{H}}_t = \left\{ \widetilde{h}(x) \,\middle|\, \widetilde{h}(x) = \min \left\{ \gamma_1^{(s)} f_1(x), \ldots, \gamma_n^{(s)} f_n(x) \right\}, s \leq t \right\}$$

and select $h_{t+1}(x)$ from weak learners of the form

$$h_{new}(x) = \min \left\{ \alpha \widetilde{h}(x), \beta f(x), 1 \right\} \tag{7}$$

with $\widetilde{h}(x) \in \widetilde{\mathcal{H}}_t$ and $f(x) \in \mathcal{F}$. This learner can be rewritten in the form of Eq. 6 with the $\gamma_j^{(t+1)}$ derived from the learned α, β, $\widetilde{h}(x)$ and $f(x)$. Thus, at each iteration we look at combinations of

the features and existing learners. As discussed in the next section, we can either consider all such combinations, or use a heuristic selection strategy.

We propose to optimize α and β separately, in stages. That is, given a value for one of the variables we optimize the other. As α and β are symmetric we show how to optimize β given α.

We need to find a value of β that maximizes r in Eq. 4. Freund et al. (2003) pointed out that this equation can be rewritten as:

$$\begin{aligned} r &= \sum_{x',x''} D^{(t)}(x',x'')(h(x') - h(x'')) \\ &= \sum_{x} \pi(x) h(x) \end{aligned}$$

where $\pi(x) = \sum_{x'} \left(D^{(t)}(x',x) - D^{(t)}(x,x') \right)$. Given the form of Eq. 7 we can write r as a function of β,

$$\begin{aligned} r(\beta) = & \sum_{\beta f_j(x) \leq \min\left(\alpha\widetilde{h}(x),1\right)} (\beta f(x) - 1)\,\pi(x) \; + \\ & \sum_{\alpha\widetilde{h}(x) < \min(\beta f(x),1)} \left(\alpha\widetilde{h}(x) - 1\right) \pi(x). \end{aligned} \tag{8}$$

Since $\sum_x \pi(x) = 0$, the sum for the third case where 1 is the maximum enters Eq. 8 as the -1 in the two sums. This function is a piecewise linear spline in β, with knots at the locations where $\beta f(x) = \min\left(\alpha\widetilde{h}(x), 1\right)$. The interior extrema (minima and maxima) of piecewise linear splines occur at their knot locations, as these are the only locations where their one-sided derivatives can change. Furthermore, $r(0) = \sum_x \pi(x) = 0$ and $r(\beta)$ is a constant for sufficiently large β. Therefore to maximize Eq. 8 we only need to consider the knots, βs that satisfy

$$\beta = \frac{\min\left(\alpha\widetilde{h}(x), 1\right)}{f(x)}, \tag{9}$$

where x is from the training data. Note that by this formula all βs are positive, maintaining the regularization constraints.

Rather than calculating (Eq. 8) separately for every value of β satisfying Eq. 9, we can use the structure of the equation to efficiently update r for consecutive values of β. Let $B = \{\beta_l\}$ be the set of all βs satisfying Eq. 9 for their corresponding x_l's ($|B| \leq I$), where $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_{|B|}$. Breaking up the sum into its components

$$\begin{aligned} O(\beta_l) &= \sum_{\beta f(x) \leq \min\left(\alpha\widetilde{h}(x),1\right)} \pi(x), \\ P(\beta_l) &= \sum_{\beta f(x) \leq \min\left(\alpha\widetilde{h}(x),1\right)} f_j(x)\pi(x), \\ Q(\beta_l) &= \sum_{\alpha\widetilde{h}(x) < \min(\beta f(x),1)} \pi(x), \\ R(\beta_l) &= \sum_{\alpha\widetilde{h}(x) < \min(\beta f(x),1)} \alpha\widetilde{h}(x)\pi(x), \end{aligned} \tag{10}$$

we can write Eq. 8 as $r(\beta_l) = \beta_l P(\beta_l) - O(\beta_l) + R(\beta_l) - Q(\beta_l)$. Now, as the βs are in order, the functions in Eq. 10 are easily updated. Let $W = \mathbf{I}\left(\alpha\widetilde{h}(x_j) < 1\right)$, then

$$\begin{aligned} O(\beta_{l+1}) &= O(\beta_l) - \pi(x_l), \\ P(\beta_{l+1}) &= P(\beta_l) - f(x_l)\pi(x_l), \\ Q(\beta_{l+1}) &= Q(\beta_l) + W\pi(x_l), \\ R(\beta_{l+1}) &= R(\beta_l) + W\alpha\widetilde{h}(x_l)\pi(x_l). \end{aligned}$$

Combining these formulas gives algorithm 2 for optimizing β.

Algorithm 2 Algorithm for optimizing β

Given α, $\widetilde{h}(x) \in \widetilde{\mathcal{H}}_t$, $f(x) \in \mathcal{F}$ and the training instances.
For all x's generate and sort the set of candidate βs, B, such that $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_{|B|}$ and $\beta_l = \frac{\min\left(\alpha\widetilde{h}(x_l),1\right)}{f(x_l)}$.
$O = \sum_{x_l} \pi(x_l)$
$P = \sum_{x_l} f(x_l)\pi(x_l)$
$Q = 0$
$R = 0$
$rbest = 0$
for $j = 1 \ldots |B|$ **do**
  $r = \beta_l P - O + R - Q$
  if $r > rbest$ **then**
    $rbest = r$
  end if
  $O = O - \pi(x_l), \quad P = P - f(x_l)\pi(x_l)$
  if $\alpha\widetilde{h}(x_l) < 1$ **then**
    $Q = Q + \pi(x_l), \quad R = R + \alpha\widetilde{h}(x_l)\pi(x_l)$
  end if
end for

5.2 Heuristics for Learner Selection

If at each boosting iteration we select from all combinations of $\widetilde{h}(x)$ and $f(x)$ we end up with an $O(T^2)$ algorithm, where T is the number of iterations. However, we can sacrifice accuracy for speed by only evaluating and selecting from a fixed sized pool of previously trained learners $\widetilde{h}(x)$ and features $f(x)$, where at each iteration the pool is randomly chosen from the full $\widetilde{\mathcal{H}}_t$ and $\mathcal{F}$.

To improve performance, instead of using a uniform sampling distribution we can apply a heuristic to focus the distribution on combinations with better potential. As Eq. 8 is composed of two sums, for r to be large the terms $f(x)\pi(x)$ and $\widetilde{h}(x)\pi(x)$ need to be large. We can consider $s_f = \sum_x f(x)\pi(x)$ and $s_h = \sum_x \widetilde{h}(x)\pi(x)$ as indicators of how well these components work with $\pi(x)$. Thus, we might expect larger r values to occur when these two score values are larger. Of course, we are discounting interactions, which is the reason for the combination.

Using these score values, we can order all $\widetilde{h}(x)$ and $f(x)$ separately, and sample such that learners and features with better scores are more likely to be selected. We opted for a polynomial weighting

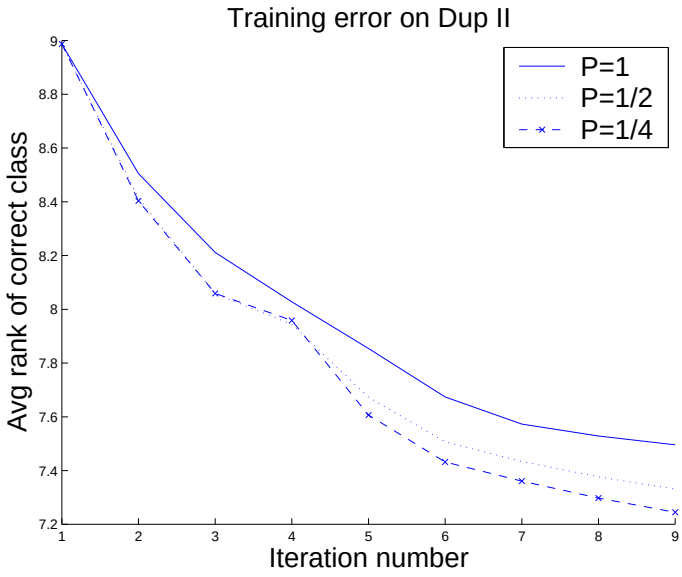

Figure 2: These plots show convergence of training error for 3 values of the heuristic selection pressure value p. These plots are averages over 10 runs and are typical of the other training data sets as well. As can be seen the heuristic improves the convergence rate of the training error.

method. Thus, for example, all $f \in \mathcal{F}$ are sorted by their score and are assigned a number based on the rank of these scores,$(maxrank - rank)/maxrank$, that gives each f an equally sized bin in the range 0 and 1. Given a random number $\xi \sim U(0,1)$, we calculate $\xi^p$and select the f that corresponds to the bin this number falls into. Here $p < 1$ can be construed as a selection pressure, where bins corresponding to higher scores are more likely to be selected. Figure 2 demonstrates the effect of different values of the p parameter in one of our experiments.

6. Face Recognition Experiments

We present experiments on the combination of face recognizers, comparing the binary learner with the MWGR learner. Given an image of a face, a face recognizer assigns a similarity score to each of the faces it is trained to recognize. These scores give a linear order or ranking to the gallery of faces. Different face recognition algorithms have different performance characteristics. Thus, we explore how combining the outputs of multiple face recognizers can improve recognition accuracy.

6.1 Algorithm Methods

We consider a data set I of face images (queries) to train on. For each query image, i in I, we need to rank all $u \in U$ faces in the gallery. In rankboost the favor function, Φ, that specifies output loss, is a function of the query and the item to be ranked. Therefore, the notational convention is to combine the query, i, and the item to be ranked, u, as an instance, $x \equiv (i,u)$. As such, $f_j(x) = f_j((i,u))$ is the

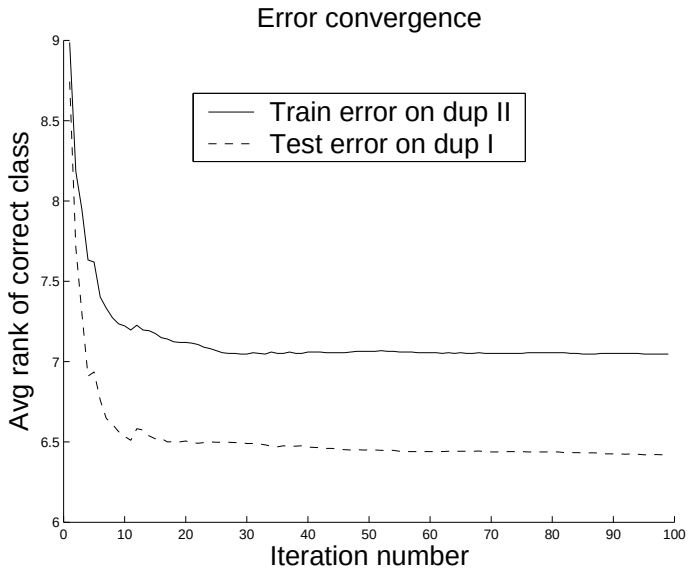

Figure 3: This plot is typical of the convergence behavior of rankboost with MWGR on the FERET data. Both training and test errors tended to converge within 10-30 iterations of boosting with no significant post-convergence divergence.

rank assigned to identity u for query image i by recognition algorithm j. As there is only one correct identity, we only care about the ranking of the one correct identity for each query. We set the favor function as stated by Freund et al. (2003) for this type of output loss. Let u^* be the the correct identity for training image i, then $\Phi((i,u),(i,u^*)) = +1$ and $\Phi((i,u^*),(i,u)) = -1$ for all $u \neq u^*$, setting all remaining elements of $\Phi(x',x'') = 0$. That is, the correct identity of a query image is given positive favor compared to all other identities for that image, while all other rankings, including interactions between training images, are given zero favor. Note that since there is no favor interaction between queries (different i's); $\Phi(x',x'')$ is effectively a function of 3 variables, (i,u',u'').

Both weak learners were trained for 100 iterations, giving them ample time to converge. See Figure 3 for an illustration of convergence times. The binary learner was trained as specified by Freund et al. (2003). At each iteration the MWGR learner was selected from a pool of candidate combinations of $f(x)$ and $\widetilde{h}(x)$, with a selection pressure of $p = 0.5$. For each candidate, first β was optimized with $\alpha = 1$, then α was optimized using the optimized β. The candidate with the most positive r value was always selected. This is summarized in algorithm 3.

6.2 Experimental Setup

FERET (Phillips et al., 2000) was a government sponsored program for the evaluation of face recognition algorithms. In this program commercial and academic algorithms were evaluated on their ability to differentiate between 1,196 individuals. The test consisted of different data sets of varying difficulty, for a total of 3,816 different images. The data sets, in order of perceived difficulty, are: the *fafb* data set of 1,195 images which consists of pictures taken the same day with different facial

Algorithm 3 Algorithm for selecting learner from pool.

Given *poolsize* and selection pressure.
Calculate s_{f_j} and s_h for all rank features and existing learners.
for $p = 1 \ldots poolsize$ **do**
  Generate $d_f \sim U(0,1)$ and $d_h \sim U(0,1)$
  Select which $h = \min\left(\alpha\widetilde{h}(x), \beta f(x), 1\right)$ to try, using s_{f_j}, s_h, u_f, u_h.
  Setting $\alpha = 1$, optimize β (algorithm 2)
  Keeping the optimized β, optimize α (algorithm 2)
  Get r of learner with this α, β
  if $r > 0$ and $r > rbest$ **then**
    $rbest = r$, $hbest = h$, $\widetilde{hbest} = \min\left(\alpha\widetilde{h}(x), \beta f(x)\right)$
  end if
end for
$h_t = hbest$, $\widetilde{\mathcal{H}}_{t+1} = \widetilde{\mathcal{H}}_t \cup \left\{\widetilde{hbest}\right\}$

expressions; the *fafc* data set of 194 images that contains pictures taken with different cameras and lighting conditions; the *dup I* data set of 488 images that has duplicate pictures taken within a year of the initial photo; and the most difficult, the *dup II* data set of 234 images which contains duplicate pictures taken more than a year later. Note that in our experiments we separate the images of dup II from the dup I data set, unlike the FERET study where dup II was also a subset of dup I.

The FERET study evaluated 10 baseline and proprietary face recognition algorithms. The baseline algorithms consisted of a correlation-based method and a number of eigenfaces (principle components) methods that differ in the internal metric they use. Of the proprietary algorithms, most were from different academic institutions and one was commercial. Of the 10 algorithms we selected three dominant algorithms. From the baseline algorithms we chose to use the *ANM* algorithm which uses a Mahalanobis distance variation on angular distances for eigenfaces (Moon and Phillips, 2001). While this algorithm's performance is not distinctive, within the class of baseline algorithms it was strong. Moreover, in accuracy with respect to average rank of the correct class on the dup I data set it demonstrated superior performance to all other algorithms. The other two algorithms we used were the University of Maryland's 1997 test submission (UMD) and the University of Southern California's 1997 test submission (USC). These algorithms clearly outperformed the other algorithms. UMD is based on a discriminant analysis of eigenfaces (Zhao et al., 1998), and USC is an elastic bunch graph matching approach (Wiskott et al., 1997).

The outputs of the 10 face recognizers on the four FERET data sets, fafb, fafc, dup I and dup II were the data for the experiments. Thus, we never had access to the actual classifiers, only to data on how they ranked the different faces in these data sets.

We conducted experiments based on homogeneous and heterogeneous data sets, testing the efficiency and robustness (adaptivity) of the MWGR procedure. For the homogeneous case we took all 4 FERET data sets and randomly shuffled them together. We call this the homogeneous data set as both the training and testing data are selected from the same combined pool. On this combined data set we did 4-fold cross validation. For each fold 75% of the data was used for training and the rest for testing. We combined the results of all four runs together for evaluation purposes.

For the heterogeneous case, in each experiment one of the FERET data sets was selected as a training set and another data set was selected for testing. This gave 12 experiments (not including training and testing on the same data set) per group of face recognizers, where we get combinations of training on easy data sets and testing on hard data sets, training on hard and testing on easy data sets, and training and testing on hard data sets. To reduce noise in our experiments the training ranks were truncated at 150, and outliers were removed.

In face recognition and other classification applications usually only the top ranks are important. Thus, in evaluating the results we focused on the top 30 ranks. All ranks returned by a boosted combiner for the correct class above 30 were truncated to 30.

In evaluating the performance of the combiners not all the test data are equally useful. We consider the following two cases as non informative. When the two best face recognizers, UMD and USC both give the correct class a rank of 1 there is very little reason for the combined rank to be different. Also when both the binary-learner-based combiner and the MWGR-based combiner give the correct class a rank greater than the truncation value (30) it makes little sense to compare between the combiners. The testing data was filtered to remove these cases, and the results are presented without them.

Before presenting the results, it should be said that rankboost with both learner types gives ranking functions that significantly outperform all the individual face recognition algorithms. In addition, in our tests both learners also clearly outperformed other standard rank combination methods, such as the Borda count, highest rank and logistic regression (Ho et al., 1992).

We present two sets of experiments—the combination of the 3 selected classifiers (ANM, UMD and USC) and the combination of all 10 classifiers. These are qualitatively different tasks. In combining 3, we seek to capitalize on the unique strengths of each classifier. In combining 10, we are introducing classifiers which may be correlated and classifiers which are comparably much noisier. The size of the pool was 6 when we combined 3 classifiers and 20 when combining all 10 classifiers.

For all experiments we measure the average rank of the correct class for both learners:

$$A = \frac{1}{N} \sum_{i=1}^{N} \min \{ Rank\,(x_i^*)\,, 30 \}\,,$$

where $Rank\,(x_i^*)$ is the rank of the correct class for query i, and the sum is over all useful test queries, as described above. The average rank difference between the learners is calculated to show the improvement in performance. To evaluate the significance of the improvement we ran paired one-sided t-tests and evaluated the significance of the p-value (a value less than 0.05 is significant). In addition we show the standard deviation of the rank difference.

6.3 Results

In the experiments with the homogeneous data sets, combining all classifiers gives an improvement in the average rank of the correct class of 0.296 for MWGR, with a standard deviation of 3.9, a paired t-test statistic of 1.76 and p-value of 0.039, where combining the ANM, UMD and USC classifiers gives an average rank improvement of 0.1 for MWGR, with standard deviation of 3.6, a paired t-test statistic of 0.68 and p-value of 0.24.

Table 2 contains the results for combining the 3 classifiers in the experiments with heterogeneous data sets (compare the combiner results in columns *bin mean* and *mwgr mean* with the aver-

	dup i	dup ii	fafb	fafc
ANM	9.52	18.14	10.88	19.71
ARL	16.85	16.96	8.94	28.02
EFAVG	15.02	20.61	14.43	26.17
EFML1	18.23	22.21	16.23	17.39
EFML2	15.85	20.33	12.21	19.78
EXCA	11.9	16.28	11.67	20.30
MSU	17.64	23.44	6.81	14.27
RUT	17.87	16.48	12.93	22.79
UMD	13.21	13.74	4.57	8.96
USC	12.44	6.85	5.84	5.17

Table 1: The best average rank of the correct class on the different data sets for all constituent face recognition systems.

age rank of the correct class for all constituent classifiers in Table 1). The *diff mean* column contains the improvement of MWGR over the binary learner in terms of average rank of the correct class. Of the 12 experiments, we see an improvement in 10 cases. Six of those 10 have significant p-values. The two no improvement experiments do not have significant p-values. Table 3 contains the results for combining all 10 classifiers. Of the 12 experiments, we see an improvement for MWGR in 11 cases. Eight or nine of those 11 have significant p-values. The one no improvement experiment does not have a significant p-value. It is interesting to note that we do not seem to see overfitting when increasing the number of constituents. In some case we see improvement, in others we see slight degradation, but all in all the combiner seems resilient to the noise of adding less informative constituents.

All sets of experiments, homogeneous data, heterogeneous data sets, combining 3 select recognizers and combining all 10 recognizers at once yielded significant improvements in accuracy, as is visible in the change in the average rank of the correct class and the significance of the statistical tests.

7. Information Retrieval Experiments

The annual Text REtrieval Conference (TREC) generates high-quality retrieval results of different systems on different retrieval tasks (Voorhees and Harman, 2001). We use the result data sets of the TREC-2001 web ad hoc task that uses actual web queries taken from web logs. This task has been used in other rank fusion experiments (Renda and Straccia, 2003). As did Renda and Straccia (2003) we combine the results of the following top 12 systems: iit01m, ok10wtnd1, csiro0mwal, flabxtd, UniNEn7d, fub01be2, hum01tdlx, JuruFull, kuadhoc, ricMM, jscbtawtl4, apl10wd.

Similar to other TREC information retrieval tasks, the TREC-2001 ad hoc task consists of 50 queries. For each query a list of relevant documents is supplied. Each system returns an ordered list of 1,000 documents for each query. The fusion goal is to combine these 12 individual lists into one list of 1,000 documents, which hopefully has greater precision than the individual systems. For the

test set	train set	bin mean	mwgr mean	diff mean	diff std	pval
dup i	dup ii	7.19	5.96	**1.22**	5.22	.7e-4
dup i	fafb	7.36	6.21	**1.14**	5.15	.001
dup i	fafc	8.31	6.27	**2.04**	5.51	.1e-6
dup ii	dup i	5.25	5.38	-.12	4.0	.658
dup ii	fafb	5.15	4.4	**0.74**	4.61	.018
dup ii	fafc	5.19	4.95	**0.23**	5.1	.275
fafb	dup i	2.62	2.22	**0.39**	3.09	.115
fafb	dup ii	3.38	2.60	**0.78**	3.79	.029
fafb	fafc	2.60	2.28	**0.32**	2.81	.142
fafc	dup i	2.85	2.78	**0.06**	3.33	.424
fafc	dup ii	2.40	2.43	-.03	2.27	.537
fafc	fafb	2.56	2.03	**0.52**	2.57	.028

Table 2: Results of combining the ANM, UMD, and USC classifiers using individual FERET data sets.

test set	train set	bin mean	mwgr mean	diff mean	diff std	pval
dup i	dup ii	6.73	5.89	**0.84**	4.79	.009
dup i	fafb	8.06	7.09	**0.96**	6.01	.014
dup i	fafc	6.68	4.87	**1.81**	5.24	.5e-6
dup ii	dup i	5.75	5.67	**0.08**	4.61	.408
dup ii	fafb	6.24	5.47	**0.76**	4.22	.009
dup ii	fafc	5.86	5.31	**0.55**	4.87	.074
fafb	dup i	2.67	2.07	**0.59**	2.97	.033
fafb	dup ii	3.56	2.45	**1.11**	4.51	.012
fafb	fafc	2.68	2.17	**0.51**	2.03	.01
fafc	dup i	3.36	2.51	**0.85**	3.23	.007
fafc	dup ii	3.17	2.95	**0.22**	3.95	.3
fafc	fafb	2.22	2.23	-0.01	2.08	.52

Table 3: Results of combining all 10 classifiers using individual FERET data sets.

50 queries of the TREC-2001 ad hoc task, the number of relevant documents that intersect with the union of system results range between 662 and 2664.

7.1 Methods

In the information retrieval task the favor function needs to show favor for relevant documents while disregarding other documents. Thus, we can set the favor function similarly to the way it was set in

JuruFull	UniNEn7d	apl10wd	csiro0mwa1	flabxtd	fub01be2
0.759	0.763	0.735	0.721	0.741	0.734
hum01tdlx	iit01m	jscbtawt14	kuadhoc2001	ok10wtnd1	ricMM
0.760	0.762	0.761	0.727	0.745	0.765

Table 4: Normalized mean average precision for each constituent.

the face recognition classification task. Consider a data set with I queries to train on. For each query, i in I, we need to rank all $u \in U$ documents. An instance in this case is a pair, $x \equiv (i,u)$. For each u^*a relevant document and u an irrelevant document for query i, we set $\Phi((i,u),(i,u^*)) = +1$ and $\Phi((i,u^*),(i,u)) = -1$, setting all remaining elements of $\Phi(x_0,x_1) = 0$. Thus, all relevant documents are given a positive favor with respect to irrelevant documents, while all other rankings, including interactions between relevant documents and interactions between irrelevant documents are given zero favor.

As the task has only 50 queries, rather than separating the data into train and test, we opted to do a cross validation performance evaluation. We use 5-fold cross validation on the *normalized mean average precision* measure (Salton and McGill, 1983), a standard IR measure which accounts for the precision (quantity of correct documents retrieved) and their rank position in the list. It is described by the following equation:

$$AveP = \frac{\sum_{r=1}^{N}(Prec(r) \times rel(r))}{number\,of\,relevant\,documents}$$

where r is the rank, N is the number of documents retrieved, $rel()$ is a binary function of the relevance of the document at rank r,and $Prec$ is the precision ([number of relevant documents retrieved] / [number of documents retrieved]) at a given cut-off rank. It is clear that higher values of $AveP$ are better. Note that for purposes of normalization we assigned unretrieved relevant documents a constant rank.

As the binary and MWGR weak learners had significantly different convergence properties on this task (see Figure 4) MWGR was trained for 100 iterations and the binary learner for 300 iterations. As in the FERET experiments, MWGR was optimized using algorithm 3, with a selection pressure of $p = 0.5$.

7.2 Results

As seen in Figure 4, the MWGR learner converges in significantly less iterations than the binary learner. This could possibly be attributed to the fact that the MWGR is a more complex function that can incorporate the rankings of multiple classifiers in each learner. Also the MWGR function is not tuned to particular rank cutoffs, whereas the binary learner is, so the MWGR can better accommodate the variety in the 1000 ranks being considered.

The normalized mean average precision for the MWGR after 100 iterations was 0.8537 and it was 0.8508 for the binary learner after 300 iterations. Compare these results with the precision of the constituents in Table 4. Both weak learners had a performance rate of change of approximately $3 * 10^{-5}$ on their final iteration (better for MWGR). A paired t-test on the cross validation results of the two learners gives a statistically significant p-value of 0.007 in favor of MWGR.

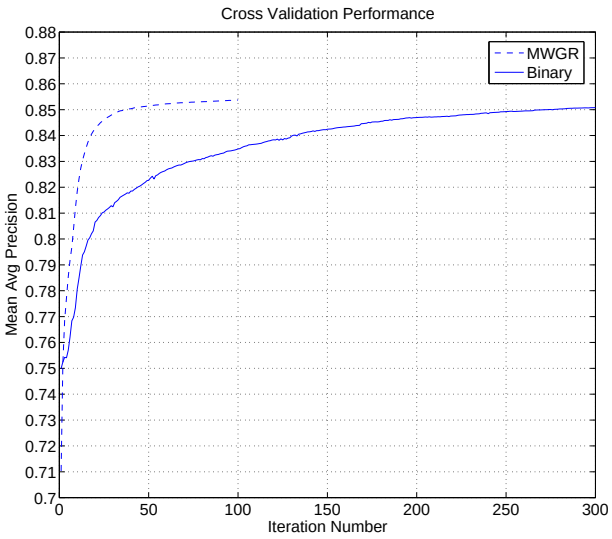

Figure 4: The cross validation mean average precision score of the two weak learners, MWGR and binary, as a function of boosting iteration number.

8. Discussion

The question of how to combine ordinal data has become an active focus of research in machine learning, as applications in pattern recognition, information retrieval and other domains have come to the forefront. A particular question of importance is how can the structure of ranks be correctly exploited to maximize performance. The semi parametric nature of rankboost offers the possibility to generate arbitrarily flexible ranking functions. But as observed this flexibility comes at a cost of significant overfitting without further regularization. Freund et al. (2003) demonstrate that successful generalization only occurs when the resulting ranking functions are constrained to be monotonic. This constraint can be thought of as a regularization that incorporates prior knowledge on the interpretation of ranks and as such, how they can be combined.

We present a regularization framework based on the concept of consistency in confidence and preference. Ranking functions with this property show a consistency in how they treat the preference and relative confidence exhibited by constituents. We prove that under a natural interpretation of preference and confidence for ranks, this consistency property of the combiner is equivalent to monotonicity and concavity of its ranking function.

We enhance rankboost by designing a weak ranking learner that exhibits consistency in preference and confidence. A computational advantage of this weak learner, called minimum weighted group ranks (MWGR) is that its parameters can be individually optimized readily with respect to the rankboost criteria, allowing it to be tested on real-world data.

In our first experiments we compare the original rankboost binary weak learner with MWGR on a task of combining the output of multiple face recognition algorithms from the FERET study. We conducted experiments on homogeneous data, testing the intrinsic efficiency of the MWGR proce-

dure. We also conducted experiments on heterogeneous data, testing the robustness or adaptivity of the procedure. In almost all cases we see that MWGR shows improved performance compared with the binary weak learner, whether combining three or all of the face recognizers, confirming the utility of this monotonic and concave learner. Our second experiment was on an Information Retrieval task taken from the TREC conference. In this task we see MWGR converges in significantly less iterations and generates statistically significant improved performance.

Final Words

Ofer Melnik and Cun-Hui Zhang are very saddened that our colleague and friend Yehuda Vardi passed away before he could give this paper his final stamp of approval. He was very enthusiastic about this research and would have been very pleased to see it come to fruition.

Acknowledgments

This research is partially supported by NSF Grants DMS 04-05202, DMS 05-04387 and, DMS 06-04571, ONR Grant N00014-02-1-056 and NSA Grant H98230-04-1-0041. Ofer Melnik would also like to thank DIMACS for their support in this research. The authors are also grateful to the editor and reviewers for their constructive suggestions which improved the presentation of the results.

Appendix A.

In this paper we showed how three-point consistency in preference and confidence implies concave and monotonic ranking functions. For two decision problems involving two pairs of rank vectors, the four-point consistency property implies the following constraints for a ranking function

$$\begin{cases} G(\mathbf{y}) < G(\mathbf{y}') \\ \mathbf{y}'_{\mathbf{B}} - \mathbf{y}_{\mathbf{B}} \leq 0 < \mathbf{y}'_{\mathbf{A}} - \mathbf{y}_{\mathbf{A}} \\ \mathbf{z}_{\mathbf{A}} \leq \mathbf{y}_{\mathbf{A}}, \mathbf{y}'_{\mathbf{A}} - \mathbf{y}_{\mathbf{A}} \leq \mathbf{z}'_{\mathbf{A}} - \mathbf{z}_{\mathbf{A}} \\ \mathbf{y}_{\mathbf{B}} \leq \mathbf{z}_{\mathbf{B}}, \mathbf{z}_{\mathbf{B}} - \mathbf{z}'_{\mathbf{B}} \leq \mathbf{y}_{\mathbf{B}} - \mathbf{y}'_{\mathbf{B}} \end{cases} \Rightarrow G(\mathbf{z}') - G(\mathbf{z}) > G(\mathbf{y}') - G(\mathbf{y}) \tag{11}$$

where $\mathbf{y}$ and $\mathbf{y}'$ are the rank vectors from the first decision problem and $\mathbf{z}$ and $\mathbf{z}'$ are the rank vectors from the second decision problem.

Unlike the three-point case, the four-point consistency property does not imply a clearly recognizable functional form for the ranking function. What we can say about it though is that as the constraints are linear, in the same way that concavity and monotonicity in the weak learner conferred the same properties to the ranking function, a weak learner that satisfies Eq. 11 will also confer those properties to the ranking function that uses it with positive weights.

References

K. Arrow. *Social Choice and Individual Values*. Wiley, 1951.

C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proc. 10th Intl. World Wide Web Conf.*, pages 613–622, 2001.

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.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

T. K. Ho. *A Theory of Multiple Classifier Systems and Its Application to Visual Word Recognition*. PhD thesis, State University of New York at Buffalo, May 1992.

T. K. Ho, J. J. Hull, and S. N. Srihari. Combination of decisions by multiple classifiers. In H. S. Baird, H. Bunke, and K. Yamamoto (Eds.), editors, *Structured Document Image Analysis*, pages 188–202. Springer-Verlag, Heidelberg, 1992.

J. Kittler and F. Roli, editors. *Multiple Classifier Systems, Lecture Notes in Computer Science 1857*, 2000. Springer.

R. Meir and G. Ratsch. *Advanced Lectures in Machine Learning, Lecture Notes in Computer Science 2600*, chapter An introduction to boosting and leveraging, pages 119–184. Springer, 2003.

O. Melnik, Y. Vardi, and C-H. Zhang. Mixed group ranks: Preference and confidence in classifier combination. *IEEE Pattern Analysis and Machine Intelligence*, 26(8):973–981, 2004.

H. Moon and P.J. Phillips. Computational and performance aspects of PCA-based face-recognition algorithms. *Perception*, 30:303–321, 2001.

P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss. The FERET evaluation methodology for face-recognition algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:1090–1104, 2000.

M.E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *18th Annual ACM Symposium on Applied Computing (SAC-03)*, pages 841–846, Melbourne, Florida, USA, 2003. ACM.

G. Salton and J.M. McGill. *Introduction to Modern Information Retrieval*. Addison Wesley Publ. Co., 1983.

E.M. Voorhees and D.K. Harman, editors. *NIST Special Publication 500-250: The Tenth Text REtrieval Conference (TREC 2001)*, number SN003-003-03750-8, 2001. Department of Commerce, National Institute of Standards and Technology, Government Printing Office. URL `http://trec.nist.gov`.

L. Wiskott, J.-M. Fellous, N. Kruger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):775–779, 1997.

W. Zhao, A. Krishnaswamy, R. Chellappa, D. Swets, and J. Weng. *Face Recognition: From Theory to Applications*, chapter Discriminant Analysis of Principal Components, pages 73–86. Springer-Verlag, Berlin, 1998.