Nonparametric scoring methods as a support decision tool for medical diagnosis – The TreeRank algorithm and its variants

Stéphan Clémençon, Marine Depecker, and Nicolas Vayatis

LTCI - Télécom ParisTech, Paris
LIMA - CEA, LIST, Gif-sur-Yvette
CMLA - ENS Cachan, Cachan

stephan.clemencon@telecom-paristech.fr
marine.depecker@cea.fr
nicolas.vayatis@cmla.ens-cachan.fr

Abstract. In this paper we propose to use nonparametric scoring methods based on ranking trees as a support decision tool for medical diagnosis. The proposed algorithms enable to order cohorts of patients according to the risk level of developing a particular disease. The aim of this paper is to illustrate the potential of various algorithms using ranking trees, particularly the variants with bagging-type aggregation of these trees, through numerical experiments. The main algorithms presented are: LeafRank (splitting rule), TreeRank (recursive partitioning for ranking) and Ranking Forests (aggregation of ranking trees), as well as the main state-of-the-art methods for bipartite ranking. Numerical experiments based on both artificial and real data sets are provided to discuss the performances of these algorithms - in terms of ROC curves and their summaries (AUC or local AUC) - but also their properties with respect to interpretability and robustness.

Keywords: Medical diagnosis, scoring rules, ranking trees, ROC curves, AUC maximization, resampling, feature randomization

1 Introduction

In many applications where decision-making is at stake, statistical modeling and machine learning methods are the main tools in order to reduce the complexity of the data: either in size - focus on a fraction of the patient population, or in dimension - focus on a fraction of decision variables before making a decision. This statement tends to become also true in medical diagnosis as medical sciences generate more and more data from the human body, and the challenge is now to take advantage of the objective information stored in massive databases. From a quantitative perspective, classification and scoring methods are the main tasks which correspond to this context of application and they correspond to some kind of hierarchy in the objectives of a rational diagnosis over a population of patients. For instance, a primary objective may be to discriminate between the potentially diseased and the potentially healthy with a low error rate (classification task), and a secondary objective would be to identify subgroups with higher risk to develop the disease so that the doctors can focus on this high level risk population (scoring task). There, the distinction between the classification task and the scoring task is that classification methods only attempt to separate two populations, while scoring methods provide an order on the individuals depending on their risk level ([14]). The classification problem has been revisited with nonparametric learning methods over the last fifteen years, however the success of state-of-the-art methods was limited in the situation encountered in medical diagnosis where: a) the class asymmetry is important (very few negative examples), b) it is equally important to monitor the two types of error, both the rate of missed detections and of false alarms. Despite the spectacular progress of classification methods, the field of medical diagnosis remained faithful to the good old logistic regression modeling. Improved versions of plain logistic regression often improve performance but lose interpretability and the trade-off between these two objectives is hard to reach. The main message of the paper
is that other nonparametric scoring methods reveal equally or even more efficient than nonpara-
metric logistic regression and present sound properties in terms of postprocessing (especially in
the analysis of variable importance). We refer to various bipartite ranking algorithms which may
be seen as nonparametric scoring methods, such as RankBoost, RankSVM, RankLS, RankNet,
P-norm push ([17], [21], [22], [7], [24]). We particularly defend our approach which relies: a) on
the use of ranking trees which perform recursive partitioning of the input space while optimizing
locally the criterion of interest, and have proved to approximate the optimal ROC curve, b) on the
aggregation of those ranking trees in order to produce a median ranking at each point in the input
space. In this paper, we provide numerical evidence of the efficiency of our approach, we evaluate
different choices of parameters and variants of our algorithms and present comparisons with state-
of-the-art algorithms in terms of performance, interpretability, and robustness. Our discussion also
includes various performance metrics such as ROC curves, Area Under an ROC Curve (AUC) and
also Local AUC when the focus is on the highest scores.

The remainder of the paper is organized as follows. In Section 2, we present the setup of the
scoring problem in presence of binary annotated data. Section 3 contains the main algorithms
derived from the principle of ranking trees. Other nonparametric scoring methods are recalled in
Section 4. Numerical experiments and results are displayed in Section 5.

2 Setup for scoring and bipartite ranking

2.1 Probabilistic Model

The probabilistic setup for bipartite ranking is the same as for binary classification. The data
appear as a sample \( D_n = \{ (X_i, Y_i) : 1 \leq i \leq n \} \), and we assume that the \( (X_i, Y_i) \)’s are i.i.d. rea-
"
2.3 Performance measures for scoring

Most of machine learning algorithms involve functional optimization. One of the main contributions of statistical learning theory was the understanding of the connection between the optimization performed and the performance metrics for assessing the quality of prediction ([3]). The choice of an objective criterion determines the optimal elements for a given problem and allows to have a baseline to compare prediction results for various methods. In the case of binary classification, considering that performance of a classifier \( g : \mathcal{X} \to \{-1, +1\} \) is measured with the misclassification rate \( L(g) = P(Y \neq g(X)) \) implies that the optimal classifier is the indicator of the level set \( R^* \). Now, if the criterion puts asymmetric weights on the two types of error \( L_w(g) = 2wP(Y = +1, g(X) = -1) + 2(1 - w)P(Y = -1, g(X) = +1) \), then the optimal classifier is the indicator of a level set of the regression function whose level is equal to \( w \). In addition, most of the efficient algorithmic approaches, such as boosting or Support Vector Machines, actually optimize surrogate (convex) risk functionals. Under rather weak assumptions, optimizing the surrogate criterion actually optimizes the misclassification rate and proving this fact was definitely one of the major achievements in the recent years (see [4]).

*The ROC curve and the AUC.* When it comes to scoring applications, the golden standard definitely is the ROC curve ([20]) as it represents the discrimination ability of a scoring rule at all levels. The ROC curve of a scoring rule \( s \) plots the true positive rate \( \beta_s(t) = P(s(X) > t \mid Y = +1) \) against the false positive rate \( \alpha_s(t) = P(s(X) > t \mid Y = -1) \) as the threshold \( t \) varies from \(-\infty \) to \(+\infty \). However, the ROC curve is a complex object to optimize as it represents a function-like criterion. It mainly serves as a visual display of performance used *a posteriori* (i.e. after training the scoring rule). A simpler object is the functional defined by the Area Under the ROC Curve, known as the AUC which can be interpreted as the probability of concordant pairs:

\[
AUC(s) = P(s(X) > s(X') \mid Y = +1, Y' = -1) + \frac{1}{2} P(s(X) = s(X') \mid Y = +1, Y' = -1),
\]

where \((X, Y)\) and \((X', Y')\) are i.i.d. pairs of observations.

*Remark 1.* In the case of preference-based ranking, then individual labels are not available and there exists no ROC curve for a preference function. However, it is possible to extend the definition of AUC to this setup. Indeed, denote preference data by triples of the form \((X, X', Z)\) where \(X, X' \in \mathcal{X}\), and \(Z \in \{-1, 0, +1\}\). Then a preference function \( \pi : \mathcal{X} \times \mathcal{X} \to \{-1, 0, +1\}\) has an AUC defined by

\[
AUC(\pi) = P(\pi(X, X') > 0 \mid Z = +1) + \frac{1}{2} P(\pi(X, X') > 0 \mid Z = 0).
\]

Note that a scoring rule \( s \) leads to the obvious choice for a preference function \( \pi(x, x') = s(x) - s(x') \) (see [9] for details and optimal elements for preference data). The converse is less straightforward ([11]).

An important limitation of the AUC is that it only measures the quality of a scoring/ordering in a global manner. Two scoring rules may present the same AUC and simultaneously have a very different behavior. Indeed, the relative weight of a discordant pair \((x, x')\) is only a function of the difference \( s(x) - s(x') \) of the scores between the two instances and not of their position over the range of \( s \). This is a crucial remark, especially if one only focuses on the instances with highest scores (they correspond to the part of the ROC curve close to the origin). There exist several criteria which emphasize the highest scores: the Average Precision (AP) and MAP, the Predicted-Rank-of-Top (PROT), the Coverage, the Discounted Cumulative Gain (DCG) and NDCG, the \( p \)-norm push ([24]). Most of these criteria can be related to linear rank statistics as shown in [10], [11].

*Local AUC.* In [10], we have introduced a truncation of the AUC called the local AUC which examines performance in the sense of the AUC but only at the top of the ranked list. The truncation is shown to be consistent with the bipartite ranking problem as the regression function \( \eta \) or any strictly increasing transform of it is still optimal. Fix \( u \in (0, 1) \) the rate of best instances and let...
$t^* = t(s, u)$ be the corresponding threshold: $u = \mathbb{P}(s(X) > t^*)$. We take the notations: $\alpha^* = \alpha_s(t^*)$ and $\beta^* = \beta_s(t^*)$, which are determined as the coordinates of the intersection of the control line $d_u = p\beta + (1 - p)\alpha$ with the ROC curve (see Figure 1). Then, the Local AUC is defined by:

$$\text{LocAUC}(s, u) = \int_{t=0}^{\alpha^*} \beta_s \circ \alpha_s^{-1}(t) \, dt + \beta^*(1 - \alpha^*),$$

where $\alpha_s^{-1}(t) = \inf\{u \in (0, 1) : \alpha_s(u) \geq t\}$ for all $t \in (0, 1)$. Instead of putting more weight on top-ranked instances, the local AUC only focuses on the top of the list with a truncation.

Fig. 1. The ROC curve, the control line and the LocAUC criterion.

3 Algorithms based on ranking trees

Our focus in the paper lies in a scoring-based method using recursive partitioning in the spirit of decision trees. In the present section, we introduce the main ideas and the algorithms in this approach.

3.1 Ranking Trees

Scoring rules output by TreeRank/LeafRank can be described as oriented binary trees, called ranking trees. Ranking trees are defined by two elements: (i) a partition of the input space $\mathcal{X}$ into cells, and (ii) a permutation on the cells which indicates the ordering induced by the rule. Denote by $\mathcal{T}_D$ the binary tree of depth $D \geq 0$ which represents a recursive partition of the input space $\mathcal{X}$. Let us equip the tree with a left-to-right orientation. We then obtain a collection of scoring rules stored in a tree-like structure (see Figure 2). The root of the tree is $C_{0, 0} = \mathcal{X}$, and each internal node $C_{d,k}$, with $0 \leq d < D$ and $0 \leq k < 2^d$, corresponds to a cell in $\mathcal{X}$ and splits into two non-empty cells $C_{d+1,2k}$ (left sibling) and $C_{d+1,2k+1} = C_{d,k} \setminus C_{d+1,2k}$ (right sibling). Any sub-tree $\mathcal{T} \subset \mathcal{T}_D$ can then be used to define a pre-order on $\mathcal{X}$: all elements in the same terminal cell will be considered as ties, and the ordering is induced by the left-to-right orientation of the sub-tree. The scoring rule is then defined as:

$$s_\mathcal{T}(x) = \sum_{C_{d,k, \text{terminal cell}}} 2^D (1 - k/2^d) \cdot \mathbb{I}\{x \in C_{d,k}\}.$$
3.2 Recursive Maximization of the AUC

The TreeRank algorithm was introduced in [12] and its theoretical properties were discussed there in detail. We now recall the principle of this method. The algorithm follows the principle of CART ([6]) excepts that it builds an oriented binary tree and uses a splitting rule which implements AUC maximization at the node level (optimization step). Consider \((X_1, Y_1), \ldots, (X_n, Y_n)\) a sample of classification data. One of the findings of the analysis is that the optimization step corresponds to solving a cost-sensitive classification problem. At a given node, the cell \(C\) is split into two parts \(C_+\) and \(C_- = C \setminus C_+\) so that the weighted empirical error

\[
\hat{L}_{C, \omega}(\Gamma) = \frac{2(1 - \omega)}{n} \sum_{i=1}^{n} I\{X_i \in C \setminus \Gamma\} \cdot I\{Y_i = +1\} + \frac{2\omega}{n} \sum_{i=1}^{n} I\{X_i \in \Gamma \cap C\} \cdot I\{Y_i = -1\}.
\]

is minimized at \(\Gamma = C_+\), for an adaptively chosen value of the weight \(\omega\). Then the TreeRank algorithm for the recursive optimization of the ROC curve can be formulated as in Figure 3.2.

![Fig. 2. Ranking tree.](image)

**TreeRank**

- **Initialization.** Start with \(C_{0,0} = X\).
- **Iterations.** For \(d = 0, \ldots, D - 1\) and \(k = 0, \ldots, 2^d - 1\),
  1. Set \(n_{d,k} = \sum_{i=1}^{n} I\{X_i \in C_{d,k}\}\). Compute the rate of false positives in cell \(C_{d,k}\):

\[
\alpha(C_{d,k}) = \frac{1}{n_{d,k}} \sum_{i=1}^{n} I\{X_i \in C_{d,k}\} \cdot I\{Y_i = +1\}.
\]

  2. Minimize \(\hat{L}_{C_{d,k}, \omega}(\Gamma)\) with \(\omega = \omega_{d,k} = \alpha(C_{d,k})\).
  3. Set \(C_{d+1,2k} = C_+\) the minimizer, and \(C_{d+1,2k+1} = C_{d,k} \setminus C_{d+1,2k}\).

- **Output.** Oriented binary tree \(T_D = \{C_{d,k} : 0 \leq d \leq D, 0 \leq k < D\}\).

![Fig. 3. The TreeRank algorithm.](image)

**Splitting rules.** The choice of splitting rules is related to the trade-off between efficiency and computability. In plain decision trees, simple rules are retained such as orthogonal splits. Here the task being more involved, more complex splitting rules may be considered. In principle, any classification algorithm with an easy tuning of asymmetric costs can do the job. In ([8]), we propose a specific splitting rule called LeafRank which uses the TreeRank algorithm at the node level recursively to build a partition and then splits the cells of this partition into two categories: one collection of
cells going to the left sibling node, and the rest going to the right sibling node. The classification of cells relies on a result which states that, for a fixed partition, the optimal permutation in the sense of the AUC is the one corresponding to a monotone ordering of the ratio of true positive rate over false positive rate per cell. The advantage of LeafRank is that it uses orthogonal splits and hence preserves the interpretability of the scoring rule. It is possible indeed to recover measures of importance of the variables and of dependence between them (see [8]). An alternative to LeafRank is to use a cost-sensitive version of a SVM classifier (see [2]), which increases the complexity of the scoring rule but at the same time, interpretability of features is lost.

3.3 Resampling and Feature Randomization

Aggregation procedures in machine learning are known to improve both performance and robustness of decision rules. This principle underlies numerous algorithms derived from bagging and boosting methods. Using the idea of resampling and aggregation, a collection of scoring rules can be built from many bootstrap samples in the spirit of bagging methods. By adding the ingredient of randomization an analogue to random forests called Ranking Forests can be derived (see [13] for a presentation in the present setup). For completeness, its description is provided in Figure 3.3. The aggregation results from a median ranking rule over the largest subpartition $P^*$ extracted from the collection of partitions implemented by the ranking trees. Indeed, every ranking tree $\mathcal{T}$ induces an ordering $\succeq_T$ over the cells of $P^*$ and the median ranking tree induces an ordering defined as the solution of a distance minimization problem.

$$\succeq^* = \arg\min_{\succeq \in R(P^*)} \sum_{\mathcal{T}} d(\succeq, \succeq^*_T),$$

where $R(P^*)$ denotes all possible orderings of cells in the subpartition $P^*$, and $d$ defines a distance between orderings, such as the Kendall tau (count of concordant pairs when comparing the two orderings), Spearman rule ($L_1$ distance between the orderings seen as vectors) or Spearman correlation coefficient ($L_2$ distance between the orderings), for instance.

---

**Fig. 4. The Ranking Forest algorithm.**
3.4 Other methods for nonparametric scoring and bipartite ranking

The optimization principles underlying algorithms used in bipartite ranking are of two types: those which focus on the target performance measure and those which aim at recovering the target regression function \( \eta(x) = P\{Y = +1 \mid X = x\}, \forall x \in X \). In the first category belong methods with an optimization based on a pairwise criterion. This is indeed the case with AUC maximization methods such as RankBoost, RankSVM, RankLS. Other algorithms like RankNet optimize a different criterion but they also use pairwise cost functions. Among the methods which focus on the optimal function to recover, we consider classification methods which output a real-valued scoring function such as Logistic Regression, AdaBoost and LogitBoost, Kernel Logistic Regression.

RankBoost ([18]). The method mimics AdaBoost in the pairwise setup. It boils down to the optimization of the following criterion.

\[
\frac{2}{n(n-1)} \sum_{i<j} \exp((Y_i - Y_j)(f(X_i) - f(X_j)))
\]

with \( f \) being a linear combination of weak ranking rules. Weak ranking rules involved are decision stumps.

The functional above is a proxy to the empirical AUC.

RankSVM ([21]) and RankLS ([23]). These two methods are export the SVM methodology to set a pairwise criterion of the form

\[
\frac{2}{n(n-1)} \sum_{i<j} d(Y_i - Y_j, f(X_i) - f(X_j)) + \lambda \| f \|_K
\]

where \( f(x) = \sum_{i=1}^{n} \alpha_i K(x, X_i), \lambda \) is a smoothing parameter and \( \| f \|_K \) is the RKHS norm of \( f \) related to the kernel \( K \).

In the case of RankSVM the loss function \( d \) is the hinge loss \( d(u,v) = (1 - uv)_{+} \) while for RankLS, it corresponds to the least-squares criterion: \( d(u,v) = (u - v)^2 \).

RankNet ([7]). Here the cost function used is based on cross entropy between the target value of the probability of a pair and the estimated probability

\[
d(Y_i - Y_j, f(X_i) - f(X_j)) = -\Delta_{i,j} \log \pi_{i,j} - (1 - \Delta_{i,j}) \log(1 - \pi_{i,j})
\]

where

\[
\Delta_{i,j} = \frac{1+(Y_i-Y_j)^2}{2} \in \{0,1/2,1\} \quad \text{and} \quad \pi_{i,j} = \frac{e^{f(X_i) - f(X_j)}}{1 + e^{f(X_i) - f(X_j)}}. \tag{1}
\]

and the candidate functions \( f \) have the same structure as neural nets.

Note that the cost function can also be written as:

\[
d(Y_i - Y_j, f(X_i) - f(X_j)) = -\Delta_{i,j} (f(X_i) - f(X_j)) + \log \left(1 + e^{f(X_i) - f(X_j)}\right).
\]

LogitBoost ([19]) and Kernel Logistic Regression ([25]). These two methods do not involve pairs of observations and therefore belong to classification methods, except that they output a real-valued function which, in theory, is related to the target regression function \( \eta \). The criterion used in both methods is the following.

\[
\frac{1}{n} \sum_{i=1}^{n} \log \{1 + \exp(-Y_i \cdot f(X_i))\},
\]

or (if \( Y_i \in \{0,1\} \)), \( \frac{1}{n} \sum_{i=1}^{n} (-Y_i \cdot f(X_i) + \log (1 + \exp(f(X_i)))) \).
An iterative boosting-type procedure is used for the optimization in LogitBoost, while Kernel Logistic Regression relies on convex risk minimization with a penalty proportional to the RKHS norm of $f$.

P-norm Push ([24]). The criterion used in this method is very similar to RankBoost but the idea is to put more weight on the largest discrepancies. The algorithm implements the optimization of the following criterion.

$$\frac{2}{n(n-1)} \sum_{i=1}^{n} \left( \sum_{j=i+1}^{n} \exp\{(Y_i - Y_j)(f(X_i) - f(X_j))\} \right)^p$$

with $p \geq 1$, and $f$ being a linear combination of weak ranking rules as in RankBoost. We point out that in the case where $p = 1$, the P-norm Push is the same algorithm as RankBoost, larger values of $p$ put more emphasis on the largest scores.

In the present paper, we only consider those methods for which there is a publicly available implementation: RankBoost, RankLS, RankSVM, KLR, AdaBoost and also P-norm Push.

4 Numerical Experiments

This section is dedicated to the experimental comparison of several variants of the TreeRank algorithm, with each other and with some competitors, and to the evaluation of the performances of such scoring methods as support decision tools for medical diagnosis. In these experiments, the performances of the compared algorithms are evaluated in terms of both scoring accuracy and stability. Before presenting the experimental results, we recall the metrics used for this experimental comparison.

4.1 Performance and stability metrics

The accuracy of the scoring rules output by the considered algorithms is evaluated through two real-valued criteria: the AUC and the Local AUC. When considering the simulated toy examples, the estimation of these quantities is based on a classical resampling procedure with $N = 30$ i.i.d. learning samples drawn from the same distribution. In the case of real benchmark data sets, both quantities are estimated through $V$-fold cross validation: $V = 10$ blocks $\{D^{(1)}, \cdots, D^{(V)}\}$ of equal size are randomly drawn from the original learning data set $D_n$, the expected full AUC being estimated as follows:

$$\overline{\text{AUC}}^{(t)} = \frac{1}{V} \sum_{v=1}^{V} \overline{\text{AUC}}^{(v)}$$

where, for each $v \in \{1, \cdots, V\}$, the quantity $\overline{\text{AUC}}^{(v)}$ denotes the empirical AUC for the scoring rule $\hat{s}^{(v)}$ trained on the data set $D^{(-v)} = D_n \setminus D^{(v)}$ and computed over $D^{(v)}$.

Another interesting indicator to consider when comparing several algorithms is their stability. Therefore, we will provide an estimate of the standard deviation of the considered performance metrics. With the notations previously introduced, the standard type error of the collection $\{\overline{\text{AUC}}^{(v)} : v = 1, \cdots, V\}$ will be denoted by $\hat{\sigma}^2$.

4.2 Artificial Data Sets

In this section, twelve variants of the TreeRank algorithm are tested on three artificial data sets, called GaussEasy20d, GaussMed20d and GaussHard20d, simulated from gaussian mixtures in the space $\mathbb{R}^{20}$. These three examples have an increasing complexity as the distance between the centers of the gaussians decreases. The covariance matrices for each class conditional distributions can be different but both are diagonal. In order to estimate properly the performance and stability
metrics, \( N = 30 \) i.i.d. learning samples are drawn from each gaussian mixture, as well as one single test sample, each sample containing \( n = 1000 \) instances. We are thus able to provide mean-valued estimates of the performance metrics previously cited.

The twelve variants considered in this experiment involve several parameters, which are detailed in Table 4.2 below. In particular, eight of the twelve heuristics are based on resampling and involve an aggregation procedure to define a consensus ranking (see 3.3). Here, the median ranking rule is computed using the pseudo-distance between orderings induced by the Spearman correlation coefficient. Moreover, the Ranking Forest algorithm involves two additional parameters, \( d_1 \) and \( d_2 \), both related to feature randomization. These parameters denote the number of predictors randomly chosen to optimize the local AUC respectively at each node of the master ranking tree output by TreeRank and at each node of the classification subtrees when the LeafRank procedure is used as splitting rule. We denote by \( d \) the total number of explanatory variables.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Splitting Rule</th>
<th>Number of bootstrap learning samples</th>
<th>TreeRank Feature Randomization</th>
<th>LeafRank Feature Randomization</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRK_CART</td>
<td>SVM classifier</td>
<td>no resampling</td>
<td>no randomization</td>
<td>no randomization</td>
</tr>
<tr>
<td>TRK_SVM</td>
<td>SVM classifier</td>
<td>no resampling</td>
<td>no randomization</td>
<td>no randomization</td>
</tr>
<tr>
<td>Bagg_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>no randomization</td>
<td>no randomization</td>
</tr>
<tr>
<td>Bagg_SVM</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>no randomization</td>
<td>no randomization</td>
</tr>
<tr>
<td>RF1_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>( d_1 = \lfloor d/4 \rfloor )</td>
<td>no randomization</td>
</tr>
<tr>
<td>RF1_SVM</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>( d_1 = \lfloor d/4 \rfloor )</td>
<td>no randomization</td>
</tr>
<tr>
<td>RF2_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>( d_1 = 1 )</td>
<td>no randomization</td>
</tr>
<tr>
<td>RF2_SVM</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>( d_1 = 1 )</td>
<td>no randomization</td>
</tr>
<tr>
<td>RF3_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>no randomization</td>
<td>( d_2 = \lfloor d/4 \rfloor )</td>
</tr>
<tr>
<td>RF4_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>no randomization</td>
<td>( d_2 = 1 )</td>
</tr>
<tr>
<td>RF5_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>( d_1 = \lfloor d/2 \rfloor )</td>
<td>( d_2 = \lfloor d/4 \rfloor )</td>
</tr>
<tr>
<td>RF6_CART</td>
<td>SVM classifier</td>
<td>( B = 50 )</td>
<td>( d_1 = \lfloor d/4 \rfloor )</td>
<td>( d_2 = 1 )</td>
</tr>
</tbody>
</table>

**Table 1.** Parametrization description for TreeRank variants.

Visual displays of the results output by the heuristics on the simulated datasets are provided on Figure 5. The histograms represent both performance and standard type error (\( \overline{\text{AUC}}^{(t)}, \hat{\sigma}^2 \)). Local AUC measure corresponding to a proportion \( u = 20\% \) of best instances is also reported.

The observation of these graphs yields two obvious conclusions. On the one hand, it appears that making use of SVM classifiers as splitting rule clearly enhance the performances of the algorithm. This is mainly due to the superior representational capacity of this classification approach, which gives a high flexibility to the splitting rule. On the other hand, we observe that the aggregation of several resampled ranking trees increases the accuracy of the output ranking. Moreover, the choice of the splitting rule seems to have a weaker impact on the performances of resampled heuristics. Eventually, we may also notice that feature randomization doesn’t seem to affect significantly the heuristics based on an SVM-type splitting rule.

On these toy examples, best performances in terms of full and Local AUC and of standard type error are achieved by the RF1_SVM heuristic. Yet, the benefits of feature randomization doesn’t clearly stand out in this experiment. Indeed, on these particular datasets, the more important the level of randomization, the worst the performance evaluated through full and local AUC and standard type error. However, several numerical experiments have shown the positive impact of randomization on the performances of classification algorithms run on real datasets ([5], [16]). Thus, in the following experiment, we will consider the comparison between the two resampled and randomized variants RF1_SVM and RF3_CART, and the six competitors previously cited, that will be
4.3 TreeRank as a Support Decision Tool for Diagnosis

In this serie of experiments, two well-performing versions of the TreeRank algorithm are compared with six competitors: RankBoost, RankLS, RankSVM, KLR, AdaBoost and P-norm Push. Therefore, we consider three real benchmark datasets related to medical diagnosis: Breast Cancer dataset, Heart Disease dataset and Hepatitis dataset. These data come from the UCI Repository (<http://archive.ics.uci.edu/ml/>).

As previously, the performances of the different algorithms are evaluated in terms of full and local AUC (AUC(t) and LocAUC), the latter being computed for a proportion u = 20% of best instances, and in terms of standard type error $\hat{\sigma}^2$. Numerical results are summarized in the graphs displayed on Figure 6.

On these data, both Ranking Forest algorithms, either based on the LeafRank procedure or on a SVM classifier, perform mostly better than the other methods, both in terms of AUC and stability. In particular, TreeRank heuristics provide the better results when focusing on the best instances. This is related to the fact that the TreeRank algorithm recursively optimizes the AUC and thus builds an estimate of the optimal curve ROC*, while all other methods optimize the AUC criterion in a global manner. Graph a. of figure 7 highlights this point: the empirical test ROC curves output by the different algorithms run on the Breast Cancer dataset are plotted on the same graph (ROC curve output by RF3_CART in bold, all others in dotted line). It clearly appears that the ROC curve approximated by RF3_CART heuristic is strictly above all others when considering the best instances (left hand side of the graph). Notice that three competitors achieve also good results on this particular example: RankBoost, AdaBoost and P-norm Push.

Besides, as previously indicated in 3.2 an interesting property of these tree-induction algorithms is their capacity to recover the impact of the predictors involved in the output scoring rule. In particular, we display on Graph b. of Figure 7 the relative impact of the predictors on the rankings output by the RF3_CART heuristic, run on the three benchmark datasets.
Fig. 6. Comparison of TreeRank and some competitors on benchmark datasets.

Fig. 7. ROC curves and predictors relative importance

This visual representation of the relative impact of the predictors enables to quickly identify the more relevant predictors for illness diagnosis. In particular, a quick look at Graph b. of Figure 7 gives the five predominant predictors for heart condition diagnosis, which are the "number of major vessels", the "oldpeak ST3", the age, the maximum heart rate achieved and the type of chest pain. Three more variables seem to be relevant for prediction: "serum cholesterol", "defect nature" and "resting blood pressure". Regarding the Hepatitis dataset, we are also able to isolate three predominant predictors, which are the rate of Albumin, the rate of Bilirubin and the patient age.
References