# A Simple Lexicographic Ranker and Probability Estimator

Peter Flach[1] and Edson Takashi Matsubara[2]

[1] Department of Computer Science, University of Bristol, United Kingdom
Peter.Flach@bristol.ac.uk
[2] Instituto de Ciências e Matemáticas e de Computação, Universidade de São Paulo
edsontm@icmc.usp.br

**Abstract.** Given a binary classification task, a ranker sorts a set of instances from highest to lowest expectation that the instance is positive. We propose a lexicographic ranker, *LexRank*, whose rankings are derived not from scores, but from a simple ranking of attribute values obtained from the training data. When using the odds ratio to rank the attribute values we obtain a restricted version of the naive Bayes ranker. We systematically develop the relationships and differences between classification, ranking, and probability estimation, which leads to a novel connection between the Brier score and ROC curves. Combining *LexRank* with isotonic regression, which derives probability estimates from the ROC convex hull, results in the lexicographic probability estimator *LexProb*. Both *LexRank* and *LexProb* are empirically evaluated on a range of data sets, and shown to be highly effective.

## 1 Introduction

ROC analysis is increasingly being employed in machine learning. It has brought with it a welcome shift in attention from classification to ranking. There are a number of reasons why it is desirable to have a good ranker, rather than a good classifier or a good probability estimator. One of the main reasons is that accuracy requires a fixed score threshold, whereas it may be necessary to change the threshold in response to changing class or cost distributions. Furthermore, good performance in both classification and probability estimation is easily and trivially obtained if one class is much more prevalent than the other, but this wouldn't be reflected in ranking performance.

In this paper we show that, even if one is primarily interested in probability estimation, it is both advantageous and feasible to first construct a ranker. We demonstrate this by proposing a very simple non-scoring ranker, which is based on a linear preference ordering on attributes, which is then used lexicographically. We can obtain calibrated probability estimates from its ROC convex hull, which is in fact equivalent to isotonic regression [1], as noted independently by [2]. In extensive experiments we demonstrate that both our lexicographic ranker and our lexicographic probability estimator perform comparably with models employing a much weaker bias.

The outline of the paper is as follows. In Section 2 we compare and contrast the notions of classification, ranking, and probability estimation, and discuss how to assess

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 575–582, 2007.

performance in each of these cases. In Section 3 we uncover the fundamental relationship between ROC curves and the Brier score or mean squared error of the probability estimates. Section 4 defines lexicographic ranking and the *LexRank* algorithm, which can easily be turned into the lexicographic probability estimator *LexProb* by means of constructing its ROC convex hull. In Section 5 we report on an extensive set of experiments, and Section 6 concludes.

## 2 Classification, Ranking, and Probability Estimation

Let $X = A_1 \times \ldots \times A_n$ be the instance space over the set of discrete attributes $A_1, \ldots, A_n$. A *classifier* is a mapping $\hat{c} : X \to C$, where $C$ is a set of labels. For a binary classifier, $C = \{+, -\}$. A *ranker* orders the instance space $X$, expressing an expectation that some instances are more likely to be positive than others. The ranking is a total order, possibly with ties. The latter are represented by an equivalence relation over $X$, so the total order is on those equivalence classes; we call them *segments* in this paper. For notational convenience we represent a ranker as a function $\hat{r} : X \times X \to \{>, =, <\}$, deciding for any pair of instances whether the first is more likely ($>$), equally likely ($=$), or less likely ($<$) to be positive than the second. (By a slight abuse of notation, we also use $>$ and $<$ for the total order on the segments of $X$). If $X_1, X_2 \subseteq X$ are segments such that $X_1 > X_2$, and there is no segment $X_3$ such that $X_1 > X_3 > X_2$, we say that $X_1$ and $X_2$ are *adjacent*. We can turn a ranker into a binary classifier by splitting the ranking between two adjacent segments. Furthermore, given a ranker $\hat{r}$, we can construct another ranker $\hat{r}'$ by joining two adjacent segments $X_1$ and $X_2$, and removing $X_1 > X_2$ from the total order. We say that $\hat{r}'$ is *coarser* than $\hat{r}$, or equivalently, that the latter is *finer* than the former.

A *scoring classifier* is a mapping $\hat{s} : X \to \mathbb{R}$, assigning a numerical score $\hat{s}(x)$ to each instance $x$. We will use the convention that higher scores express more preference for the positive class. A *probability estimator* is a scoring classifier that assigns probabilities, i.e., a mapping $\hat{p} : X \to [0,1]$. $\hat{p}(x)$ is taken to be an estimate of the posterior $p(+|x)$, i.e., the true probability that a random instance with attribute-value vector $x$ belongs to the positive class. Clearly, given a scoring classifier $\hat{s}$ (or a probability estimator) we can construct a ranker $\hat{r}$ that orders instances on decreasing scores. Furthermore, we can turn a scoring classifier into a classifier by turning the associated ranker into a classifier as described above, or equivalently, by setting a threshold $t \in \mathbb{R}$ and assigning all instances $x$ such that $\hat{s}(x) \geq t$ to the positive class and the remaining instances to the negative class.

We illustrate the above on decision trees. [3] and [4] showed that decision trees can be used as probability estimators and hence as rankers. We obtain a probability estimator from a decision tree by considering the numbers of positive ($n_i^+$) and negative ($n_i^-$) training examples belonging to the $i$-th leaf. The estimated posterior odds in $leaf_i$ is then $\frac{P(+|leaf_i)}{P(-|leaf_i)} = \frac{n_i^+}{n_i^-}$ (or $\frac{n_i^+ + 1}{n_i^- + 1}$ if we apply Laplace correction, as recommended by [4]). The corresponding ranker is obtained by ordering the leaves on decreasing posterior odds. A classifier is obtained by labelling the first $k^+$ leaves in the ordering positive and the remaining $k^-$ leaves negative. Figure 1 shows a small example.

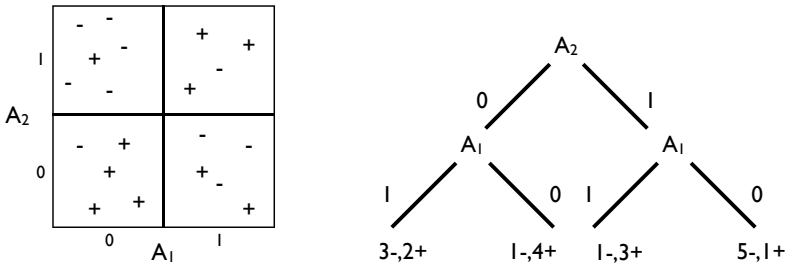


**Fig. 1.** A data set, and an induced decision tree. Instead of leaf labellings, the class distributions of training instances are indicated for each leaf, which can be used to obtain the ranking leaf 2 – leaf 3 – leaf 1 – leaf 4.

The performance of a binary classifier can be assessed by tabulating its predictions on a test set with known labels in a contingency table, from which true and false positive rates can be calculated. An ROC plot plots true positive rate on the Y-axis against false positive rate on the X-axis; a single contingency table corresponds to a single point in an ROC plot. The performance of a ranker can be assessed by drawing a piecewise linear ROC curve. Each segment of the curve corresponds to one of the segments induced by the ranker; the order of the ROC segments corresponds to the total ordering on the ranking segments. If the $i$-th segment contains $n_i^+$ out of a total of $n^+$ positives and $n_i^-$ out of $n^-$ negatives, the segment's vertical length is $n_i^+/n^+$, its horizontal width is $n_i^-/n^-$ and its slope is $l_i = \frac{n_i^+}{n_i^-}c^{-1}$, where $c = n^+/n^-$ is the prior odds. We will denote the proportion of positives in a segment as $p_i = \frac{n_i^+}{n_i^+ + n_i^-} = \frac{l_i}{l_i + 1/c}$. We will call these *empirical probabilities*; they allow us to turn a ranker into a probability estimator , as we will show later. The area under the ROC curve or *AUC* estimates the probability that a randomly selected positive is ranked before a randomly selected negative, and is a widely used measure of ranking performance. An ROC curve is *convex* if the slopes $l_i$ are monotonically non-increasing when moving along the curve from $(0,0)$ to $(1,1)$. A concavity in an ROC curve, i.e., two or more adjacent segments with increasing slopes, indicates a locally worse than random ranking. In this case, we would get better ranking performance by joining the segments involved in the concavity, thus creating a coarser classifier.

The performance of a scoring classifier can be assessed in the same way as a ranker. Alternatively, if we know the true scores $s(x)$ we can calculate a loss function such as *mean squared error* $\frac{1}{|T|}\sum_{x \in T}(\hat{s}(x) - s(x))^2$, where $T$ is the test set. In particular, for a probabilistic classifier we may take $s(x) = 1$ for a positive instance and $s(x) = 0$ for a negative; in that case, mean squared error is also known as the *Brier score* [5]. Note that the Brier score takes probability estimates into account but ignores the rankings (it does not require sorting the estimates). Conversely, ROC curves take rankings into account but ignore the probability estimates. Brier score and AUC thus measure different things and are not directly comparable.

## 3 ROC Curves, the Brier Score, and Calibration

In this section we demonstrate a fundamental and novel relationship between Brier score and ROC curves. We do this by means of a decomposition of the Brier score in terms of calibration loss and refinement loss. A very similar decomposition is well-known in forecasting theory (see, e.g., [6]), but requires a discretisation of the probability estimates and is therefore approximate. Our decomposition uses the segments induced by the ranking and is therefore exact.

**Theorem 1.** *Given an ROC curve produced by a ranker on a test set $T$, let $\hat{p}_i$ be the predicted probability in the i-th segment of the ROC curve. The Brier score is equal to* $BS = \frac{1}{|T|}\sum_i n_i(\hat{p}_i - p_i)^2 + \frac{1}{|T|}\sum_i n_i p_i(1-p_i)$.

*Proof.* $BS = \frac{1}{|T|}\sum_{x\in X}(\hat{p}(x)-p(x))^2 = \frac{1}{|T|}\sum_i [n_i^+(\hat{p}_i-1)^2 + n_i^-\hat{p}_i^2] = \frac{1}{|T|}\sum_i [n_i\hat{p}_i^2 - 2n_i^+\hat{p}_i + n_i^+] = \frac{1}{|T|}\sum_i [n_i(\hat{p}_i - \frac{n_i^+}{n_i})^2 + n_i^+(1-\frac{n_i^+}{n_i})] = \frac{1}{|T|}\sum_i n_i(\hat{p}_i - p_i)^2 + \frac{1}{|T|}\sum_i n_i p_i(1-p_i)$.

Both terms in this decomposition are computed by taking a weighted average over all segments of the ROC curve. The first term, the *calibration loss*, averages the squared prediction error in each segment. It is important to note that the error is taken relative to $p_i$, which is the proportion of positives in the segment *and thus not necessarily 0 or 1*. In other words, the calibration loss as defined above relates the predicted probabilities to the empirical probabilities obtained from the slopes of the segments of the ROC curve. The second term in the Brier score decomposition is called *refinement loss*. This term is 0 if and only if all ROC segments are either horizontal or vertical, which is the case if all segments are singletons. Consequently, refinement loss is related to the coarseness of the ranker, hence its name. For instance, refinement loss is maximal (0.25) for the ranker which ties all test instances. Notice that refinement loss only takes empirical probabilities into account, not predicted probabilities. It is therefore a quantity that can be evaluated for any ranker, not just for probability estimators. Notice also that, while the Brier score itself does not require ranking the probability estimates, its decomposition into calibration loss and refinement loss does. As an illustration, the decision tree from Figure 1 has 0 calibration loss on the training set (if Laplace correction is not used) and refinement loss $(5\cdot 4/5\cdot 1/5 + 4\cdot 3/4\cdot 1/4 + 5\cdot 2/5\cdot 3/5 + 6\cdot 1/6\cdot 5/6)/20 = 0.18$.

**Theorem 2.** *The calibration loss is 0 only if the ROC curve is convex.*

*Proof.* Suppose the ROC curve is not convex, then there are two adjacent segments such that $\hat{p}_i > \hat{p}_j$ but $l_i < l_j$. From the latter it follows that $p_i < p_j$, and thus at least one of the error terms $(\hat{p}_i - p_i)^2$ and $(\hat{p}_j - p_j)^2$ is non-zero.

**Theorem 3.** *Let $\hat{p}$ be a probability estimator with a convex ROC curve but a non-zero calibration loss. Let $\hat{p}'$ be derived from $\hat{p}$ by predicting $p_i$ rather than $\hat{p}_i$ in each segment. Then $\hat{p}'$ has the same AUC as $\hat{p}$ but a lower Brier score.*

*Proof.* $\hat{p}'$ may be coarser than $\hat{p}$ because it may merge adjacent segments with $\hat{p}_i > \hat{p}_j$ but $p_i = p_j$. But this will not affect the shape of the ROC curve, nor the AUC. We thus have that the slopes of all segments remain the same, hence the $p_i$; but since $\hat{p}'$ has zero calibration loss the Brier score is decreased.

Many models do not guarantee convex training set ROC curves. For such models, the above suggests a straightforward procedure to obtain calibrated probabilities, by constructing the *convex hull* of the ROC curve [7]. This can be understood as creating a coarser ranking, by joining adjacent segments that are in the wrong order. Clearly, joining segments results in additional refinement loss, but this is compensated by setting the probability estimates equal to the empirical probabilities, hence obtaining zero calibration loss (although in practice we don't achieve zero calibration loss because we apply the Laplace correction in order to avoid overfitting). This procedure can be shown to be equivalent to isotonic regression [1]; a proof can be found in [2].

## 4 Lexicographic Ranking

While a ranker is commonly obtained by sorting the scores of a scoring classifier as indicated in Section 2, it is possible to define a ranker without scores. Probably the simplest way to do so is to assume a preference order on attribute values, and to use that ordering to rank instances lexicographically. In the rest of this paper, we will show that such a simple ranker, and the probability estimates derived from it, can perform competitively with less biased models such as decision trees and naive Bayes.

For notational convenience we will assume that all attributes are binary; since a nominal attribute with $k$ values can be converted into $k$ binary attributes, this doesn't represent a loss of generality.

**Definition 1 (Lexicographic ranking).** *Let $A_1,\ldots,A_n$ be a set of boolean attributes, such that the index represents a preference order. Let $v_{i+}$ denote the preferred value of attribute $A_i$. The* lexicographic ranker *corresponding to the preference order on attributes and attribute values is defined as follows:*

$$\hat{r}_{lex}(x_1,x_2) = \begin{cases} > \text{ if } A_j(x_1) = v_{j+} \\ < \text{ if } A_j(x_1) \neq v_{j+} \end{cases}$$

*where $j$ denotes the lowest attribute index for which $x_1$ and $x_2$ have different values (if no such index exists, the two instances are tied).*

A lexicographic ranker can be represented as an unlabelled binary decision tree with the following properties: (1) the only attribute occurring at depth $i$ is $A_i$ – i.e., along each path from root to leaf the attributes occur in the preference order; (2) in each split, $v_{i+}$ is the left branch. Consequently, the ranking order is represented by the left-to-right order of the leaves. We call such a tree a *lexicographic ranking tree*. The decision tree shown in Figure 1 is also a lexicographic ranking tree, representing that $A_2$ is preferred to $A_1$, 1 is the preferred value of $A_1$, and 0 the preferred value of $A_2$. However, as a lexicographic ranking tree, its leaves are ranked left-to-right, which results in a non-convex ROC curve. Clearly, decision trees are much more expressive than lexicographic ranking trees.

We can also draw a connection between lexicographic ranking and the naive Bayes classifier, as we will now show. The naive Bayes ranker obtained from the data from Fig. 1 is as follows. Using $LR(\cdot)$ to denoted the likelihood ratio estimated from the

data, we have $LR(A_1=0)=\frac{p(A_1=0|+)}{p(A_1=0|-)}=5/6$, $LR(A_1=1)=\frac{p(A_1=1|+)}{p(A_1=1|-)}=5/4$, $LR(A_2=0)=\frac{p(A_2=0|+)}{p(A_2=0|-)}=6/4$, and $LR(A_2=1)=\frac{p(A_2=1|+)}{p(A_2=1|-)}=4/6$. The prior odds doesn't affect the ranking, and so we can just use the products of these marginal likelihood ratios to determine the ranking: $LR(A_1=1)LR(A_2=0)=30/16 > LR(A_1=0)LR(A_2=0)=30/24 > LR(A_1=1)LR(A_2=1)=20/24 > LR(A_1=0)LR(A_2=1)=20/36$. This is a lexicographic ranking, which is equivalent to the lexicographic ranking tree in Fig. 1.

**Definition 2.** *LexRank is the lexicographic ranker which uses the following preference criteria. The preferred value $v_{i+}$ for attribute $A_i$ is defined as the one which has $LR(A_i=v_{i+})>1$ (if there is no such value then the attribute doesn't express preference and can be discarded). The preference order on attributes is defined by sorting them on decreasing* odds ratio $OR(A_i)=\frac{LR(A_i=v_{i+})}{LR(A_i=v_{i-})}$*, where $v_{i-}$ denotes the non-preferred value.*

**Theorem 4.** *For every LexRank ranker over a given set of binary attributes, there exists a data set such that LexRank and naive Bayes, trained on that data set, result in equivalent rankers.*

*Proof.* Assuming without loss of generality that $A_1,\ldots,A_n$ are already sorted on decreasing odds ratio and that 0 is the preferred value of each attribute, then the leaves of the lexicographic ranking tree can be interpreted as integers in binary representation that are ordered from 0 to $2^n$. The naive Bayes ranker will respect this ordering, *unless* there is an $i$ such that $OR(A_i)<\prod_{j>i}OR(A_j)$, as this would reverse any ranking decision for instances differing not just in $A_i$ but also in attributes ranked lower than $A_i$.

For instance, if $OR(A_1)=5$, $OR(A_2)=3$ and $OR(A_3)=2$, then *LexRank* will rank 011 before 100, whereas naive Bayes will reverse the ranking, since $A_2=0\wedge A_3=0$ outweighs $A_1=0$.

We conclude that *LexRank* exhibits a much stronger bias than naive Bayes, but as we show in the next section this added bias does not result in a loss of ranking performance. We can turn *LexRank* into a calibrated probability estimator by deriving probability estimates from its convex hull. The resulting lexicographic probability estimator is called *LexProb*.

## 5 Experimental Evaluation

Our experiments were conducted on 27 data sets from the UCI repository [8] using the Weka toolbox. Continuous attributes were discretised using unsupervised ten-bin discretisation; non-binary $k$-valued attributes were replaced with $k$ binary-valued attributes. We compared the following algorithms: *LexRank* and *LexProb*, the algorithms proposed in this paper; *NB* and *J48*, naive Bayes and decision tree learners as implemented in Weka; and *CaliNB*, which uses the same wrapper to calibrate probability estimates as *LexRank*, but applied to *NB*. We ran *J48* without pruning and with Laplace correction, to get good ranking performance. We evaluated both ranking performance by means of AUC, and probability estimation performance by means of the Brier score. We used 10-fold cross-validation for all data sets except the ones with less than 270 instances where we used 5-fold cross-validation to ensure the folds contained

enough instances for the tests to be meaningful. Table 1 shows the results of all pairwise counts of wins/ties/losses (row vs. column) using AUC (lower triangle) and BS (upper triangle). Most of these counts are not significant according to a sign test (critical value at 0.05 confidence level is 20.7). For illustrative purposes we included *LexRank* in the Brier score results, by calculating scores based on the binary representation of the ranking; these scores are clearly not meaningful as probabilities, resulting in four out of five of the significant counts.

**Table 1.** Counts of wins/ties/losses using AUC (lower triangle) and BS (upper triangle). Counts in **bold face** are significant according to a sign test at 0.05 confidence level.

| set | LexRank | LexProb | NB | CaliNB | J48 |
|---|---|---|---|---|---|
| LexRank | - | 1 0 **26** | 4 0 **23** | 2 0 **25** | 4 0 **23** |
| LexProb | 12 2 13 | - | 17 0 10 | 10 0 17 | 17 0 10 |
| NB | 18 3 6 | 17 2 8 | - | 3 0 **24** | 11 0 16 |
| CaliNB | 16 1 10 | 16 1 10 | 9 2 16 | - | 17 0 10 |
| J48 | 12 1 14 | 14 2 11 | 11 2 14 | 9 2 16 | - |

The Friedman test uses average ranks of each algorithm over all data sets. From the AUC results, the average ranks are 2.407 for *NB*, 2.926 for *CaliNB*, and 3.222 for *J48*, *LexRank*, and *LexProb*, resulting in an F-statistic of 1.38. The critical value of the F-statistics with 4 and 104 degrees of freedom and at 95 percentile is 2.46. According to the Friedman test, the null-hypothesis that all algorithms have similar ranking performance should not be rejected. The average ranks on the basis of the Brier scores, on the other hand, result in an F-statistic of 17.20, and we proceed to a post-hoc analysis.

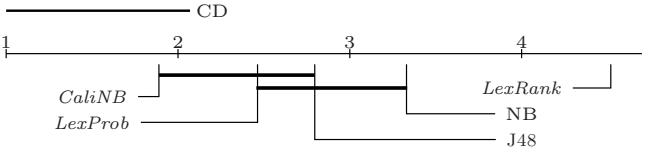


**Fig. 2.** Critical Difference diagram

According to the Bonferroni-Dunn statistic, the Critical Difference (CD) for comparing the mean-ranking of an algorithm to a control at 95 percentile is 1.07. Mean-ranking differences above this value are significant. Following [9], we can now plot the average ranks in a CD diagram (Figure 2). In this diagram, we connect algorithms that are not significantly different. We also show the CD above the main axis. The analysis reveals that *CaliNB* performs significantly better than *NB*. This demonstrates that isotonic regression can significantly improve the probability estimates of naive Bayes. *LexProb* is connected with – i.e., has comparable performance with – *CaliNB*, *J48* and *NB*.

Finally, we report some runtime results. A single train and test run over all 27 data sets without cross-validation takes 46.72 seconds for *LexRank* against 49.50 seconds for *NB*, and 94.14 seconds for *LexProb* against 100.99 seconds for *CaliNB*.

## 6 Conclusions

In this paper we have made a number of contributions. First of all, we have clearly defined the relationship and differences between classification, ranking and probability estimation. Secondly, we have defined the notion of lexicographic ranking, which simply employs a linear preference order on attributes. To the best of our knowledge, this is the first ranker that doesn't base its ranking on numerical scores. Thirdly, we have shown that using the odds ratio for ranking attributes results in a lexicographic ranker, called *LexRank*, which is a restricted version of naive Bayes. Fourthly, we have demonstrated a close and fundamental connection between ROC curves and the Brier score, linking in particular the calibration of probability estimates to the convexity of the ROC curve. Experimental results show that lexicographic ranking, and the probability estimates derived from it using isotonic regression, perform comparably to decision trees and naive Bayes.

**Acknowledgments.** We thank the anonymous reviewers for their constructive comments and the Brazilian Research Council CAPES (Proc.N. 2662060).

## References

1. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: Brodley, C.E., Danyluk, A.P. (eds.) Proceedings of the Eigteenth International Conference on Machine Learning (ICML 2001), pp. 609–616. Morgan Kaufmann, San Francisco (2001)
2. Fawcett, T., Niculescu-Mizil, A.: PAV and the ROC convex hull. Machine Learning 68(1), 97–106 (2007)
3. Ferri, C., Flach, P.A., Hernández-Orallo, J.: Learning decision trees using the area under the ROC curve. In: Sammut, C., Hoffmann, A.G. (eds.) Proceedings of the Nineteenth International Conference (ICML 2002), pp. 139–146. Morgan Kaufmann, San Francisco (2002)
4. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning 52(3), 199–215 (2003)
5. Brier, G.: Verification of forecasts expressed in terms of probabilities. Monthly Weather Review 78, 1–3 (1950)
6. Cohen, I., Goldszmidt, M.: Properties and benefits of calibrated classifiers. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 125–136. Springer, Heidelberg (2004)
7. Provost, F., Fawcett, T.: Robust classification for imprecise environments. Machine Learning 42(3), 203–231 (2001)
8. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)