Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves

Nicolas Lachiche LACHICHE@LSIIT.U-STRASBG.FR
LSIIT UMR 7005 CNRS-Université Louis Pasteur, Pôle API, Bd Brant, 67400 Illkirch, France

Peter Flach PETER.FLACH@BRISTOL.AC.UK
Department of Computer Science, University of Bristol, Woodland Road, Bristol BS8 1UB, United Kingdom

Abstract

The probability estimates of a naive Bayes classifier are inaccurate if some of its underlying independence assumptions are violated. The decision criterion for using these estimates for classification therefore has to be learned from the data. This paper proposes the use of ROC curves for this purpose. For two classes, the algorithm is a simple adaptation of the algorithm for tracing a ROC curve by sorting the instances according to their predicted probability of being positive. As there is no obvious way to upgrade this algorithm to the multi-class case, we propose a hill-climbing approach which adjusts the weights for each class in a pre-defined order. Experiments on a wide range of datasets show the proposed method leads to significant improvements over the naive Bayes classifier's accuracy. Finally, we discuss an method to find the global optimum, and show how its computational complexity would make it untractable.

1. Introduction

The naive Bayes classifier has the seemingly paradoxical property that, while the assumption of conditional independence of the attributes is violated in many domains, the predictions derived from its probability estimates are often fairly accurate. The paradox can be resolved by noting that in order to get good classification performance it is sufficient that the posterior class probability estimates are well-separated. It doesn't matter that the posterior estimates are uncalibrated, as long as positive instances tend to be assigned higher predicted probabilities of being positive than negative instances. As far as the naive Bayes classifier is concerned, the posterior estimates are simply scores from which it makes predictions by applying a decision criterion. The usual decision criterion is 'predict the class with the highest posterior probability', or in two-class problems 'predict the class whose posterior probability exceeds 0.5'. However, given that the probability estimates are inaccurate, there is no real justification for such a decision criterion. In our view, the naive Bayes decision criterion is an additional model parameter that has to be learned from the data. In the two-class case, the decision criterion is simply a threshold, which can be elegantly learned from the data using (two-class) ROC analysis. However, there is no straightforward way to upgrade this method to more than two classes, because the necessary apparatus of multi-class ROC analysis is lacking. The main contribution of this paper is a method to learn a decision criterion from the data by tuning class weights using greedy optimisation. The method is experimentally shown to lead to significant improvements in classification accuracy.

1.1. ROC analysis

ROC analysis (Received Operating Characteristic) was introduced in signal detection theory to describe how well a receiver could distinguish a signal from noise. It has a long history in medical data analysis where it is used to investigate sensitivity/specificity trade-offs of diagnostic tests. It was introduced in machine learning relatively recently (see e.g. (Provost & Fawcett, 2001)) and is quickly gaining popularity as a tool for analysing and visualising many aspects of machine learning algorithms. Some recent developments are described in (Fawcett, 2003); many of our notations are borrowed from that paper.

A two-class ROC curve is a two-dimensional curve in which the True Positive rate (TPr) is plotted on the Y axis and the False Positive rate (FPr) is plotted on the X axis. Those rates are estimated as follows:

$$TPr = \frac{\text{number of positive instances correctly classified}}{\text{total number of positive instances}}$$

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

$$FPr = \frac{\text{number of negative instances misclassified}}{\text{total number of negative instances}}$$

A discrete classifier, e.g. a decision tree, produces a single ROC point (FPr, TPr). Many classifiers, such as Bayesian classifiers or neural networks, naturally assign to each instance i a score $f(i)$ expressing the degree to which i is thought to be positive. In particular, probabilistic classifiers such as naive Bayes output posterior probability distributions over classes. In classification, it is often more convenient to work with scores as they can be manipulated without the need for re-normalisation. Probabilities can be converted into scores by the following monotonic transformation: let $f(P,i)$ (resp. $f(N,i)$) denote the estimated probabilities that i is positive (resp. negative), then $f(i) = \frac{f(P,i)}{f(N,i)} = \frac{f(P,i)}{1-f(P,i)}$. A probabilistic classifier can be turned into a categorical classifier by setting a threshold on the score, i.e. instance i is classified as positive if $f(i)$ is greater than a fixed threshold t, and negative otherwise. In the absence of any other information, this threshold is usually set to 1, which corresponds to a uniform posterior distribution in the case of a probabilistic classifier. However, we argue that this decision threshold should in fact be learned from the data in order to maximise accuracy or minimise cost.

1.2. Learning the decision threshold

ROC analysis provides an elegant way to do this. Each value of the decision threshold corresponds to an ROC point, and a piecewise linear ROC curve can be drawn by varying the threshold and plotting the corresponding points. (There is a more efficient algorithm based on ordering the instances according to their predicted scores, which will be explained in Section 2.) This curve gives an aggregated assessment of the classification power of the scoring classifier, without reference to a decision threshold. The area under the curve estimates the probability that a randomly chosen positive instance obtains a higher score than a randomly chosen negative instance (Hand & Till, 2001).

Figure 1 shows an ROC curve corresponding to the performance of a first-order naive Bayes classifier, `1BC2`, on predicting a desired property of drugs against Alzheimer's disease (see Section 4). The crossing vertical and horizontal lines indicate the point corresponding to the default threshold $f(i) = 1$ (i.e., classifying i as positive if $f(P,i) > f(N,i)$). The diagonal line through this point is the iso-accuracy line corresponding to all points in the ROC space having the same accuracy $a = p(P)TPr + p(N)(1-FPr)$ as the default threshold, where $p(P)$ and $p(N)$ are the prior class probabilities. The main point to note is that a large part of ROC curve lies above this iso-accuracy line, and consequently there are many thresholds that would achieve a higher accuracy (on the same set of instances) than the default threshold. In this particular case these thresholds are found to the right of the default point, corresponding to an increased number of positive predictions, hence a lower decision threshold. The optimal decision threshold can be found graphically by sliding the iso-accuracy line upwards until it intersects with the ROC curve in a single point.

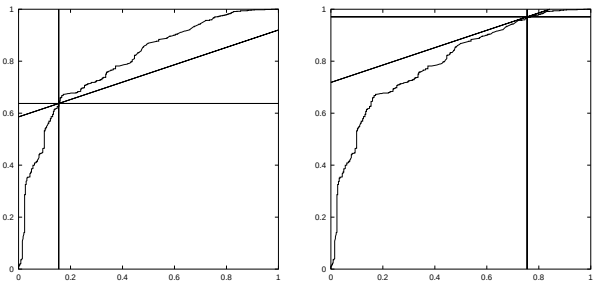

Figure 1. (Left) ROC curve obtained with a probabilistic classifier. The crosshair indicates the true and false positive rates obtained by the default threshold, and the diagonal line through this point is the iso-accuracy line connecting all points in ROC space having the same accuracy. (Right) The same ROC curve, with the crosshair identifying the point with optimal accuracy.

It is easy to factor misclassification costs into this analysis. The equation of the iso-accuracy line of all points achieving accuracy a is $TPr = \frac{p(N)}{p(P)}FPr + \frac{a-p(N)}{p(P)}$. Given an ROC curve, the optimal point on this curve is uniquely determined by the slope of the iso-accuracy lines, i.e., the class distribution. If we denote by $c(N,P)$ (resp. $c(P,N)$) the cost of misclassifying a positive (resp. negative) instance, the expected cost of applying the classifier corresponding to a point (FPr, TPr) in the ROC space is $c = p(P)(1-TPr)c(N,P) + p(N)FPrc(P,N)$. An iso-cost line could be drawn with all points having the same cost c: $TPr = \frac{c(P,N)p(N)}{c(N,P)p(P)}FPr + 1 - \frac{c}{c(N,P)p(P)}$. That is, unequal misclassification costs can be simply taken into account by modifying the class ratio and thus the slope of the iso-performance lines. We could similarly take correct classification profits into account, or use a class ratio that is different from the one in the set of instances from which the ROC curve is constructed.

1.3. Contribution of this paper

An algorithm to find the optimal point on a given two-class ROC curve is thus a fairly straightforward construction, and all essential ingredients (probabilistic ROC curves, iso-performance lines) were already present in (Provost & Fawcett, 2001). However, a generalisation to more than two classes is problematic, because there is no fully developed multi-class ROC analysis. A full n-class ROC analysis would require $n(n-1)$ dimensions, distinguishing all possible misclassifications (one-against-one). By

aggregating all misclassifications per class we obtain a n-dimensional approximation (one-against-all). We briefly review the few results in multi-class ROC analysis that we are aware of. An analysis of the special case of 3 classes (one-against-all) is given in (Mossman, 1999), concentrating on the proper statistical interpretation of the volume under the ROC surface (in analogy with the two-class area under the ROC curve). (Srinivasan, 1999) proves that, given a set of one-against-one misclassification rates, there is a unique convex polytope enclosing those points (in analogy with the two-class convex hull). (Hand & Till, 2001) propose a multi-class version of area under the ROC curve by averaging the areas of the n one-against-all curves.

None of these works addresses the issue of constructing a multi-class ROC hypersurface for a given probabilistic model. Current indications are that such a construction, which would be necessary to find a globally optimal multi-class decision criterion, is computationally intractable (we discuss this at the end of the paper). Our approach works as follows. The kind of decision criterion that we consider simply assigns the class with maximum score, taking into account positive weights for each class. The $n-1$ weights (one can be set arbitrarily) are learned using greedy hill-climbing search assuming a fixed ordering of classes. The two-class procedure for finding a global optimum outlined above is a special case of this procedure.

The outline of the paper is as follows. In Section 2 we present in detail the optimisation algorithm for two-class domains. Section 3 presents the main contribution of the paper, which is an optimisation algorithm for multi-class domains. In Section 4 we give experimental results validating our approach on a variety of propositional and first-order datasets. Section 5 discusses issues related to finding the global optimum. Section 6 concludes.

2. Optimisation on two-class domains

Given the relationships between a point on a two-class ROC curve and the accuracy or cost of the corresponding classifier, accuracy/cost can easily be optimised by considering all the points of the ROC curve. We now show how this optimisation can be achieved by a variant of the practical method for constructing an ROC curve given by (Fawcett, 2003). The basic idea of such an algorithm is to sort the instances i according to decreasing scores $f(i)$; starting in (0,0), the curve is drawn by moving $1/P$ up if the next instance is an actual positive, and moving $1/N$ to the right if it is an actual negative, until we reach (1,1) (P and N are the total number of positive and negative instances). In the case of draws, i.e. x positives and y negatives getting the same score, we draw a single diagonal segment by moving x/P steps up and y/N steps to the right at once.

Table 1 modifies this algorithm such that it returns the best threshold from a list of instances i and their scores $f(i)$. If desired, this algorithm can be merged with the previous one into a single algorithm for drawing the curve and calculating the optimal threshold simultaneously. The threshold is always chosen in between two successive scores f and f' by setting it to $\sqrt{ff'}$ (since our implementation of the naive Bayes classifier actually maintains logarithms of the scores, this corresponds to averaging the log scores).

Table 1. Algorithm to find the threshold that optimises accuracy or cost on a given ROC curve.

```
algorithm findBestThreshold
Inputs: instances i, scores f(i)
Output: threshold t resulting in optimal
        cost/accuracy
  optimum = cost/accuracy assuming all
    instances i are classified as negative
  current = optimum
  sort scores f(i) in decreasing order
  best_threshold = highest score
  for each different score f
    current = update cost/accuracy assuming
      all instances s.t. f(i) > f
      are classified positive
    if current improves on optimum then
      optimum = current
      best_threshold = sqrt(f * next(f))
  return best_threshold
```

Algorithm `findBestThreshold` is optimal in the sense that it results in the highest accuracy or lowest cost achievable with the given scores on the given set of instances. In practice, the quality of the optimisation depends on the quality of the scores, i.e. of the probabilistic model. If the model is overfitting, then what appears to be the optimal point on the ROC curve may actually lead to worse performance. On the other hand, if the model is good then the optimisation will not decrease performance.

3. Optimisation on multi-class domains

In this section we adapt the two-class optimisation method to deal with more than two classes. In our approach, a multi-class probabilistic classifier is turned into a categorical classifier by setting weights on the class scores $f(Q,i)$ for all classes Q, and assigning the class which maximises the weighted score. The main difficulty is that there is no simple algorithm tracing the ROC surface by sorting instances. We are not aware of any algorithm that, given the scores $f(Q,i)$, efficiently calculates all possible classifications of a set of instances that can be achieved by setting weights on these scores. Section 5 gives some further considerations regarding the complexity of this problem. In the absence of such a method, we develop in this section a hill-climbing algorithm that optimises the weights separately.

3.1. From ROC curves to ROC polytopes

Let $r(P,A)$ be the proportion of instances of actual class A that are predicted in class P. Then the expected accuracy is $\sum_A p(A)r(A,A)$ and the expected cost is $\sum_A \sum_{P\neq A} p(A)r(P,A)c(P,A)$, where $p(A)$ is the prior probability of class A, and $c(P,A)$ is the cost of misclassifying an instance of A as class P. Therefore, for an n-class domain, an (n^2-n)-dimensional ROC space has to be considered where points have coordinates $(r(P,A))$, for all classes $A, P \neq A$. That is, any n-class classifier produces an (n^2-n)-tuple of misclassification rates, which corresponds to a single point in ROC space. The two-dimensional ROC curve becomes an (n^2-n)-dimensional polytope.

In order to obtain a multi-class decision criterion that can be optimised, the single decision threshold used with two classes is replaced by weights w_A associated with each class A (one weight can be set to 1, since there are $n-1$ degrees of freedom). The weighted probabilistic classifier classifies instance i into class P maximising $w_P f(P,i)$, where $f(P,i)$ denotes the probabilistic classifier's estimate of the probability that instance i belongs to class P. For two classes, this reduces to classifying an instance as positive if $f(P,i) > w_N f(N,i)$, i.e. the weight on the negative class plays the role of the threshold on $\frac{f(P,i)}{f(N,i)} = f(i)$ (assuming w_P has been set to 1). In the multi-class case, however, there is no direct relation between weights and global probability thresholds. For any two classes P and Q, the only thing we can say is that an instance will not be classified in class Q, but possibly in class P, if $\frac{f(P,i)}{f(Q,i)} > \frac{w_Q}{w_P}$.

3.2. Setting the weights

The trick of the two-class algorithm in Table 1 is not to consider all possible thresholds, but only those such that the classification of a single instance changes from positive to negative. To achieve this, the instances are sorted according to $f(i)$: $f(i_1) > f(i_2) > \ldots > f(i_m)$, so considering any threshold value between $f(i_k)$ and $f(i_{k+1})$ classifies the k first instances as positive and the $m-k$ remaining instances as negative. As a result, the algorithm only considers at most $m+1$ classifications out of the 2^m possible distributions of m instances over 2 classes. Our aim is to upgrade this algorithm from two classes to n classes.

The main difficulty is that there are $\frac{n^2-n}{2}$ orderings of the m instances, according to $\frac{f(P,i)}{f(Q,i)}$ for all classes $P, Q \neq P$.

Definition 1 *Given two classes P and Q, and two instances i and j, $>_{P/Q}$ is the ordering defined by*

$$i >_{P/Q} j \text{ if and only if } \frac{f(P,i)}{f(Q,i)} > \frac{f(P,j)}{f(Q,j)}$$

Given two classes P and Q, $Q \neq P$, it would be straightforward to fix all other weights w_R to 0, w_P to 1 and tune w_Q in order to change step by step the classifications of all instances from class P to class Q, in order to find the optimal accuracy or cost. This could be repeated for all classes $P, Q \neq P$. However, it is obvious that such an approach does not consider all possible classifications of the instances, for instance those classifications corresponding to more than two weights being strictly greater than 0. At the other extreme, a blind exhaustive search algorithm could enumerate all possible classifications of the m instances. However, there are n^m ways of classifying m instances into n classes. Obviously this is untractable in most cases, for instance in the diterpene domain (cf. Section 4.2), there are $23^{1503} \sim 10^{2046}$ combinations.

Therefore we propose a hill-climbing approach. Given an ordering of the classes and assuming they are labelled $1, 2, \ldots, n$, the weights are fixed in that order, by considering only instances of the class currently being optimised against all classes whose weights already have been fixed. The optimisation consists either in minimising the expected cost, or in maximising the expected accuracy. Table 2 gives the main steps of this algorithm. It starts by fixing the first weight $w_1 = 1$. At step P, the first $P-1$ weights have been fixed, and weight w_P is tuned by calling `findBestWeight`, taking into account only predictions into classes $Q \leq P$ (since the weights of the remaining classes are still zero).

Table 2. Algorithm to set the class weights.

```
algorithm setWeights
Inputs: instances i, actual_classes,
        ordering < on classes
Outputs: weights for all classes and
         corresponding cost/accuracy
  initialise all weights w(P) to 0
  w(1) = 1
  for P=2 to n
    I = []
    for each instance i
      find class Q<P maximising w(Q)*f(Q,i)
      store predicted_classes(i) = Q
      f(i) = f(P,i) / (w(Q)*f(Q,i))
      add i to I
    w(P) = findBestWeight(I,f,P)
  return weights w(P)
end algorithm
```

The `findBestWeight` algorithm (Table 3) is actually a variant of the `findBestThreshold` algorithm (Table 1). Since there are more than 2 classes, and the misclassification costs depend on the actual and predicted classes, three changes are required. The 'predicted' class is $Q = \arg\max_{R<P}(w_R f(R,i))$, i.e., the class in which the instance would be classified if only the first classes $R < P$ were considered; this is stored for each instance before calling

`findBestWeight`. The score $f(i)$ of each instance is calculated as follows: the 'positive' class is P, and the 'negative' class is $Q = \arg\max_{R<P}(w_R f(R,i))$, therefore the score is $f(i) = \frac{f(P,i)}{\max_{R<P}(w_R f(R,i))} = \frac{f(P,i)}{w_Q f(Q,i)}$. Given the score, the predicted and actual classes of each instance, the third novelty of the `findBestWeight` algorithm is the addition of class P as input, to estimate the appropriate costs when instances become misclassified into the class P due to an increase of its weight w_P.

Table 3. Algorithm to find the best weight.

```
algorithm findBestWeight
Inputs: instances i, scores f, class P,
        actual_classes, predicted_classes
Output: weight for class P resulting in
        optimal cost/accuracy
  optimum = cost/accuracy assuming all i
    are classified in predicted_classes(i)
  current = optimum
  sort scores f in decreasing order
  best_weight = highest score
  for each different score f
    current = update cost/accuracy assuming
      all instances s.t. f(i) > f
      are classified in class P
    if current improves on optimum then
      optimum = current
      best_weight = sqrt(f * next(f))
  return best_weight
end algorithm
```

3.3. Discussion

In the three-class case, the effect of reweighting the naive Bayes scores can be visualised as follows.[1] For each instance to be classified, the naive Bayes classifier predicts a triple of probabilities, one for each class (here we assume normalised scores). These triples can be visualised as points in a probability cube; since the probabilities add up to 1, the points lie in an equilateral triangle connecting three corners of the cube. Each corner of this triangle represents a particular class to which it assigns probability 1; each side of the triangle represents a probability of zero for the class opposite that side (Figure 2). More generally, the probability for a particular class in a given point corresponds to the distance to the side opposite that class.

The decision criterion of assigning the class with maximum probability corresponds to class boundaries that are perpendicular to the sides of the triangle; in the case of equal weights these lines intersect at the triangle's centre of gravity (dotted lines in Figure 2). Our algorithm will first adjust class 2 against class 1; this fixes the vertical decision boundary. By taking class 3 into account, we find the optimal point on this vertical boundary. Notice that this last step may change some class 1 or 2 predictions into class 3 predictions, but it will never change class 1 predictions into class 2 or vice versa.

[1] This visualisation was inspired by (Mossman, 1999).

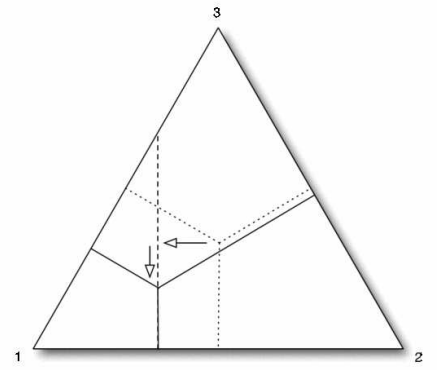

Figure 2. Three-class probability triangle with uniformly weighted decision criterion (dotted lines), increased weight of class 2 relative to class 1 (dashed line), and increased weight of class 3 relative to classes 1 and 2 (solid lines).

The algorithm is not guaranteed to find a local optimum: for instance, it might be possible to find, given the weights for classes 2 and 3 just determined, a better weight for class 1. Also, the approach depends on the order of the classes: if we start by adjusting class 3 against class 1 (i.e., moving the decision boundary perpendicular to the 1-3 side), we may end up in a different point altogether. Intuitively, it seems a good idea to start with the largest classes, since adjustments involving those classes have (potentially) the biggest impact. This has been verified experimentally by comparing with random orderings as well as the reverse ordering (smallest classes first). A more sophisticated strategy would be to take the unweighted predictions (and the cost matrix) into account.

It should be noted that, when adjusting class 3 against classes 1 and 2, it is possible to distinguish between different kinds of misclassifications (3 misclassified as 1, 3 misclassified as 2, 1 misclassified as 3, etc.) and thus take class-against-class misclassification costs into account. In fact, even when adjusting class 2 against class 1, we take class 3 instances into account, because misclassifying them as class 1 may have different cost from misclassifying them as class 2.

4. Experimental results

Experiments have been carried out to evaluate the improvement in accuracy of a naive Bayes classifier obtained with our method. Two first-order Bayesian classifiers were used: `1BC` (Flach & Lachiche, 1999) and its successor `1BC2` (Lachiche & Flach, 2002). `1BC` applies dynamic propositionalisation, while `1BC2` is a true first-order classifier which works by decomposing probability distributions over

structured terms. On propositional domains both classifiers return the same probabilistic model. We considered propositional (i.e., attribute-value) as well as relational datasets. All experiments were performed using a 10-fold cross-validation: training the probability model as well as the decision criterion on the training set, evaluating the weighted classifier thus obtained on the test set and averaging the results.

4.1. Propositional datasets

All 25 propositional datasets were taken from the UCI machine learning repository (Blake & Merz, 1998). Table 4 reports the number of classes and the accuracies of 1BC without optimisation, 1BC with optimisation, and Weka Naive Bayes (Witten & Frank, 2000), for each dataset (sometimes for different targets of a given dataset). Figures are indicated in **boldface** whenever the observed difference of the accuracy between 1BC with optimisation and the other classifier is significant using a one-sided paired t-test with a confidence of 95%.

Table 4. Accuracies on propositional datasets.

Settings	cl.	1BC	1BC opt.	NB
Audiology	24	**67.5%**	**78.5%**	**65.5%**
Bridges 2 (t-or-d)	2	85.3%	88.2%	82.4%
Bridges 2 (material)	3	86.8%	84.9%	86.8%
Bridges 2 (span)	3	67.4%	67.4%	69.6%
Bridges 2 (rel-l)	3	68.0%	68.9%	68.0%
Bridges 2 (type)	7	58.5%	59.4%	57.5%
Car	4	**85.3%**	**88.8%**	**85.3%**
Credit	2	86.5%	**85.5%**	**81.4%**
Dermatology	6	97.5%	97.3%	97.3%
Ecoli	8	83.6%	**82.1%**	**85.1%**
Flag (religion)	8	64.9%	62.4%	56.2%
Flare 2 (common)	9	**76.1%**	**82.8%**	**76.4%**
Flare 2 (moderate)	9	**91.5%**	**96.3%**	**93.1%**
Flare 2 (severe)	9	**97.5%**	**99.4%**	**97.4%**
Glass	7	67.3%	**65.4%**	**48.6%**
Horse-colic (surgical)	2	79.6%	79.6%	79.3%
Horse-colic (site)	12	**40.2%**	**45.7%**	45.4%
Horse-colic (type)	5	55.1%	56.8%	55.2%
Horse-colic (subtype)	4	**56.0%**	**63.0%**	62.5%
Horse-colic (code)	11	37.8%	38.3%	35.9%
Image segmentation	7	88.9%	**88.4%**	**85.7%**
Mushroom	2	**95.5%**	**98.1%**	**95.8%**
Nursery	5	**90.3%**	**91.5%**	**90.3%**
Post-operative	3	70.0%	71.1%	70.0%
Vote	2	**90.1%**	**88.0%**	**90.1%**

Table 5. Weka Naive Bayes and 1BC without optimisation compared to 1BC with optimisation on propositional datasets.

Measure	1BC	Weka NB
Number of wins	15 - 8	20 - 4
Number of significant wins	9 - 1	10 - 2

Table 5 summarises the significance results. The optimisation decreases accuracy significantly only once when compared with unoptimised 1BC, and twice when compared with Weka Naive Bayes (indicating that in one of these cases the lower accuracy was caused by the poorer probability model learned by unoptimised 1BC).

4.2. Relational datasets

Three relational datasets have been considered. The first dataset is about drugs against Alzheimer's disease (Boström & Asker, 1999), with four distinct targets. The second dataset concerns identifying mutagenic compounds (Srinivasan et al., 1994; Muggleton et al., 1998). We considered the 'regression-friendly' dataset of 188 molecular compounds. In these experiments, we used the atom and bond structure of the molecule as one setting, adding the lumo and logp properties to get a second setting, and finally adding boolean indicators I_a and I_1 as well. We also considered the latter propositional properties separately. The third dataset is concerned with Diterpenes, which are one of a few fundamental classes of natural products with about 5000 members known (Džeroski et al., 1998). The classification task consists of identifying types of diterpenes from NMR spectra. Table 6 reports the accuracies of 1BC without and with optimisation, and of 1BC2 without and with optimisation, for each of these datasets, targets, and settings .

Table 7. Comparison on relational domains of first-order classifiers with and without optimisation.

Measure	1BC	1BC2
Number of wins	7 - 5	6 - 6
Number of significant wins	4 - 1	2 - 2

Table 7 summarises the significance results on the relational data. The results are a bit more mixed than in the propositional case, but still there are only three significant losses out of 24 experiments. The optimisation seems to work better for 1BC, but this may be due to the fact that in many cases unoptimised 1BC2 comes already quite close to the optimised accuracy. There is one quite spectacular failure of the optimisation method for the propositional version of mutagenesis (only lumo, logp, inda and ind1). Clearly, the probability model is overfitting here so that the optimum on the training data does not correspond to the optimum on the test data.

5. On finding the global optimum

In Section 3.2, a brute-force algorithm, enumerating all possible classifications of the m instances, was discarded due to an exponential number of possible distributions of m instances over n classes. However, given a probabilistic classifier and a set of instances, many of those com-

Table 6. Accuracies on relational datasets.

Settings	cl.	1BC	1BC opt.	1BC2	1BC2 opt.
Alzheimer's (Inhibit amine reuptake)	2	**68.1%**	**79.7%**	**69.0%**	**78.3%**
Alzheimer's (Low toxicity)	2	74.4%	74.9%	74.5%	74.9%
Alzheimer's (High acetyl cholinesterase inhibition)	2	**68.7%**	**69.6%**	70.9%	70.0%
Alzheimer's (Reversal of memory defi ciency)	2	**62.8%**	**76.8%**	**66.5%**	**76.6%**
Mutagenesis: lumo and logp only	2	71.3%	69.1%	71.3%	69.1%
Mutagenesis: lumo, logp, inda and ind 1 only	2	**83.0%**	**73.9%**	**83.0%**	**73.9%**
Mutagenesis: Atoms and bonds only	2	80.3%	79.3%	81.9%	80.3%
Mutagenesis: Plus lumo and logp	2	82.4%	79.3%	81.9%	83.0%
Mutagenesis: Plus inda and ind1	2	87.2%	84.6%	82.4%	81.9%
Diterpenes: Propositional	23	78.2%	78.3%	78.2%	78.3%
Diterpenes: Relational	23	67.9%	69.1%	**70.9%**	**68.9%**
Diterpenes: Propositional and relational	23	**73.5%**	**79.9%**	79.0%	79.7%

binations are impossible using weighted maximisation as the decision criterion. In this section, we investigate some properties of multi-class ROC space in the hope of finding a more efficient algorithm to find the global optimum.

We can use the $>_{P/Q}$ orderings to detect some impossible classifications.

Theorem 1 *Given two instances i and j, if $i >_{P/Q} j$, then there are no weights w_P, w_Q such that i is classified in class Q and j is classified in class P.*

Proof: i is classified in class Q implies that $w_P f(P,i) < w_Q f(Q,i)$, so $\frac{f(P,i)}{f(Q,i)} < \frac{w_Q}{w_P}$. j is classified in class P implies that $w_P f(P,j) > w_Q f(Q,j)$, so $\frac{f(P,j)}{f(Q,j)} > \frac{w_Q}{w_P}$. So $\frac{f(P,i)}{f(Q,i)} < \frac{f(P,j)}{f(Q,j)}$, which contradicts $i >_{P/Q} j$. □

Another issue consists in finding a set of weights corresponding to a given classification. Notice that the classification of an instance i in class P corresponds to $n-1$ inequalities: $\forall Q \neq P, w_P f(P,i) > w_Q f(Q,i)$. Considering two classes P and Q, each instance i classified in class P implies the constraint $\frac{f(P,i)}{f(Q,i)} > \frac{w_Q}{w_P}$ on the weights w_P and w_Q. Since the instances are ordered according to $>_{P/Q}$, considering the instance classified in class P minimising $\frac{f(P,i)}{f(Q,i)}$, i.e. the bottom instance classified in class P according to $>_{P/Q}$, is the only necessary constraint. Symmetrically, only the top instance classified in class Q according to $>_{P/Q}$ should be considered.

Theorem 2 *Let I_P be the set of instances that are classified in class P, and let $m_{P/Q} = max_{i \in I_Q}(\frac{f(P,i)}{f(Q,i)})$ and $M_{P/Q} = min_{i \in I_P}(\frac{f(P,i)}{f(Q,i)})$, then $m_{P/Q} < \frac{w_Q}{w_P} < M_{P/Q}$.*

Proof: For all instances $i \in I_Q$, $w_P f(P,i) < w_Q f(Q,i)$, i.e. $\frac{f(P,i)}{f(Q,i)} < \frac{w_Q}{w_P}$. So $m_{P/Q} < \frac{w_Q}{w_P}$. Similarly, for all instances $i \in I_P$, $w_P f(P,i) > w_Q f(Q,i)$, i.e. $\frac{f(P,i)}{f(Q,i)} > \frac{w_Q}{w_P}$. So $\frac{w_Q}{w_P} < M_{P/Q}$. □

Suppose that the n classes are labelled $1,2,\ldots,n$. w_1 can be fixed to an arbitrary value, e.g. 1. Then the weights w_Q should satisfy the constraint: $max_{P<Q}(m_{P/Q} w_P) < w_Q < min_{P<Q}(M_{P/Q} w_Q)$. An open issue is whether this set of constraints is always satisfiable. If it is, it would mean that the orderings $>_{P/Q}$ are the only necessary tools to detect impossible classifications.

Note that this process of finding possible weights, which is linear in the number of instances m and quadratic in the number of classes n, has to be repeated for each possible classification. In other words, it does not prevent enumerating all classifications. Moreover, preliminary experiments indicated that the number of possible classifications is exponential itself. So it might be impractical to find the global optimum by upgrading the algorithm in Table 1 from two-class to multi-class domains.

As we noticed earlier, two different possible classifications of the instances might lead to the same point. So an alternative approach considering the ROC space might be more successful. This is a perspective for future work.

6. Conclusions

The probability estimates of a naive Bayes classifier are inaccurate if some of its underlying independence assumptions are violated. The decision criterion for using these estimates for classification therefore has to be learned from the data. This paper proposes the use of ROC curves to experimentally find better decision boundaries. The method can easily take non-uniform misclassification costs into account. For two classes, the algorithm is a simple adaptation of the algorithm for tracing a ROC curve by sorting the instances. There is no obvious way to upgrade this algorithm to the multi-class case. We propose a hill-climbing approach which adjusts the weights for each class in a predefined order, determining the weight for class P only on the basis of the weights of classes Q already determined.

The method starts from an initial probabilistic model and constructs an ROC curve and a set of weights determining the decision boundaries. This makes the approach applicable to any learning method which outputs class scores: for instance, our method could equally well be applied to recalibrate decision trees. Experimental evaluation on a range of propositional and relational datasets demonstrates that the method works very well in practice. For instance, when used in combination with `1BC` we achieved 13 significant wins, 22 draws and only 2 losses.

Future work includes a further study of the feasibility of finding a global optimum. The main open problem here is how a set of weights on class scores constrains the possible classifications of a test set. It might be a good idea to start from decision tree classifiers rather than naive Bayes classifiers, since a decision tree ROC curve can be obtained by ordering leaves rather than instances (Ferri et al., 2002) and therefore the constraints on possible classifications and weights will be stronger. Another topic for future work is to employ stochastic optimisation techniques.

Acknowledgements

Part of this work was supported by the EU project *Data Mining and Decision Support for Business Competitiveness: Solomon Virtual Enterprise* (IST-1999-11495). Support from National ICT Australia and the University of New South Wales, where the second author was a Visiting Research Fellow during completion of the paper, is gratefully acknowledged. Thanks are due to the anonymous reviewers for helpful comments; to Henrik Boström and Sašo Džeroski for providing us with the Alzheimer and Diterpene datasets, respectively; and to all contributors to the UCI repository.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Boström, H., & Asker, L. (1999). Combining divide-and-conquer and separate-and-conquer for efficient and effective rule induction. *Proceedings of the 9th International Workshop on Inductive Logic Programming* (pp. 33–43). Springer-Verlag.

Džeroski, S., Schulze-Kremer, S., Heidtke, K. R., Siems, K., Wettschereck, D., & Blockeel, H. (1998). Diterpene structure elucidation from ^{13}c NMR spectra with inductive logic programming. *Applied Artificial Intelligence*, *12*, 363–383. Special Issue on First-Order Knowledge Discovery in Databases.

Fawcett, T. (2003). *ROC graphs: Notes and practical considerations for data mining researchers*Tech report HPL-2003-4). HP Laboratories, Palo Alto, CA, USA. Available:http://www.purl.org/net/tfawcett/papers/HPL-2003-4.pdf.

Ferri, C., Flach, P., & Hernández-Orallo, J. (2002). Learning decision trees using the area under the roc curve. *Proceedings of the 19th International Conference on Machine Learning* (pp. 139–146). Morgan Kaufmann.

Flach, P., & Lachiche, N. (1999). 1BC: A first-order Bayesian classifier. *Proceedings of the 9th International Workshop on Inductive Logic Programming* (pp. 92–103). Springer-Verlag.

Hand, D., & Till, R. (2001). A simple generalisation of the Area Under the ROC Curve for multiple class classification problems. *Machine Learning*, *45*, 171–186.

Lachiche, N., & Flach, P. (2002). 1BC2: A true first-order Bayesian classifier. *Proceedings of the 12th International Conference on Inductive Logic Programming*. Springer-Verlag.

Mossman, D. (1999). Three-way ROCs. *Medical Decision Making*, *19*, 78–89.

Muggleton, S., Srinivasan, A., King, R., & Sternberg, M. (1998). Biochemical knowledge discovery using Inductive Logic Programming. *Proceedings of the first Conference on Discovery Science*. Berlin: Springer-Verlag.

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, *42*, 203–231.

Srinivasan, A. (1999). *Note on the location of optimal classifiers in n-dimensional ROC space* (Technical Report PRG-TR-2-99). Oxford University Computing Laboratory, Oxford.

Srinivasan, A., Muggleton, S., King, R., & Sternberg, M. (1994). Mutagenesis: ILP experiments in a non-determinate biological domain. *Proceedings of the 4th International Workshop on Inductive Logic Programming* (pp. 217–232). Gesellschaft für Mathematik und Datenverarbeitung MBH.

Witten, I., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with java implementations*. Morgan Kaufmann.