Round Robin Rule Learning

Johannes Fürnkranz JUFFI@OEFAI.AT
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Wien, Austria

Abstract

In this paper, we discuss a technique for handling multi-class problems with binary classifiers, namely to learn one classifier for each pair of classes. Although this idea is known in the literature, it has not yet been thoroughly investigated in the context of inductive rule learning. We present an empirical evaluation of the method as a wrapper around the Ripper rule learning algorithm on 20 multi-class datasets from the UCI database repository. Our results show that the method is very likely to improve Ripper's classification performance without having a high risk of decreasing it. In addition, we give a theoretical analysis of the complexity of the approach and show that its training time is within a small constant factor of the training time of the sequential class binarization technique that is currently used in Ripper.

1. Introduction

Pairwise classification is a technique for reducing a multi-class problem to multiple 2-class problems by learning a classifier for each pair of classes. It has been previously applied to various problems, mostly in the support vector machines community (see Section 7), but we are not aware of an extensive experimental study, in particular not in the context of inductive rule learning algorithms.

In this paper, we will show that this technique in fact gives significant improvements in predictive accuracy and provide a detailed analysis of the complexity of the approach, which shows that it is not much slower, but may in fact be considerably faster than conventional approaches that train each class against all other classes. In particular, we prove that the performance ratio of pairwise classification over conventional approaches goes to zero with increasing asymptotic complexity of the base algorithm. Our experimental results underline the efficiency of the approach, even though we only experimented with a proof-of-concept implementation in the form of a wrapper around the Ripper rule learning algorithm.

2. Class Binarization

Many machine learning algorithms are inherently designed for binary (two-class) decision problems. Prominent examples are neural networks with single output nodes, support vector machines and separate-and-conquer rule learning. In addition, all regression algorithms can, in principle, be used for binary decision problems, but not for multi-class problems (unless, maybe, if the class values are ordered). However, real-world problems often have multiple classes. Fortunately, there exist several simple techniques for turning multi-class problems into a set of binary problems. We will call such techniques class binarization techniques.

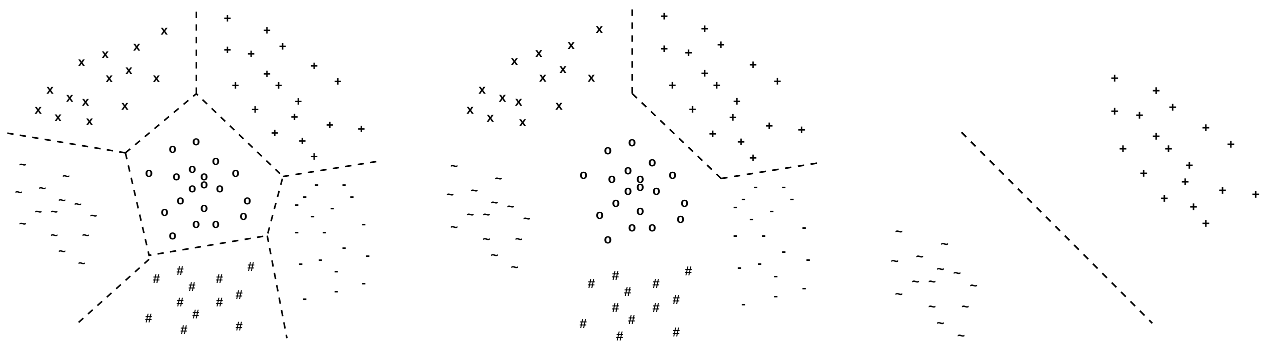
Definition 2.1 (class binarization, decoding) *A* class binarization *is a mapping of a multi-class learning problem to several 2-class learning problems in a way that allows a sensible* decoding *of the prediction, i.e., allows to derive a prediction for the multi-class problem from the predictions of the set of 2-class classifiers. The learning algorithms used for solving the 2-class problems is called the* base classifier.

The most popular class binarization rule is the unordered or one-against-all class binarization, where one takes each class in turn and learns binary concepts that discriminate this class from all other classes. It has been independently proposed for rule learning (Clark & Boswell, 1991), neural networks (Anand et al., 1995), and support vector machines (Cortes & Vapnik, 1995).

Definition 2.2 (unordered class binarization) *The* unordered class binarization *transforms a c-class problem into c 2-class problems. These are constructed by using the examples of class i as the positive examples and the examples of classes j, $j = 1 \dots c, j \neq i$ as the negative examples.*

The name "unordered" originates from Clark and Boswell (1991), who proposed this approach as an alternative to the decision-list learning approach that was originally used in CN2 (Clark & Niblett, 1989). In other fields, the strategy has different names, but as our main concern is rule learning, we stick with the terminology used there.

The rule learning algorithm Ripper (Cohen, 1995), which will be our main test object, also provides an option for


(a) *Multi-class learning* one classifi er separates all classes.

(b) *Unordered learning* c classifi ers, each separates one class from all other classes. Here: + against all other classes.

(c) *Round robin learning* $\frac{c(c-1)}{2}$ classifi ers, one for each pair of classes. Here: $+$ against $\sim$.

Figure 1. Unordered and round robin binarization for a 6-class problem.

inducing unordered rule sets. Its default mode of operation, however, is an ordered approach.

Definition 2.3 (ordered class binarization) *The* ordered class binarization *transforms a c-class problem into $(c-1)$ 2-class problems. These are constructed by using the examples of class i, $i = 1 \dots c-1$ as the positive examples and the examples of classes j, $j = i+1 \dots c$ as the negative examples.*

Note that ordered class binarization imposes an order on the induced classifiers, which has to be adhered to at classification time: the classifier learned for discriminating class 1 from classes $2 \dots c$ has to be called first, and if this classifier assigns the example to class 1, no other classifier is called. If not, the example is passed on to the next classifier. Unordered class binarization, on the other hand, has to call each of its constituent binary classifiers and requires some external criterion for *decoding* the individual predictions into a final prediction. Typical decoding rules vote the predictions of the individual classifiers, possibly by taking into account the confidences of the predictions.

For a uniform view and an empirical evaluation of several class binarization and decoding techniques, we refer the reader to (Allwein et al., 2000). In the following section, we discuss one particular technique in more detail.

3. Round Robin Classifi cation

In this section, we will discuss a more complex class binarization procedure, the pairwise classifier. The basic idea is quite simple, namely to learn one classifier for each pair of classes. In analogy to round robin tournaments in sports and games, in which each participant is paired with each other participant exactly once, we call this procedure *round robin binarization*.

Definition 3.1 (round robin, pairwise binarization) *The* round robin *or* pairwise *class binarization transforms a c-class problem into $\frac{c(c-1)}{2}$ 2-class problems, one for each unique pair of classes $<i,j>$ $(i = 1 \dots c-1$, $j = i+1 \dots c)$. The binary classifier is trained on the training examples of classes i and j. The examples of classes $\neq i,j$ are ignored for the binary classifier $<i,j>$.*

When classifying a new example, each of the learned base classifiers determines to which of its two classes the example is more likely to belong to. The winner is assigned a point, and in the end, the algorithm will predict the class that has accumulated the most points. In our version, ties are broken by preferring the class that is more frequent in the training set (or flipping a coin if the frequencies are equal), but more elaborate techniques are possible (see Section 6).

Round robin binarization is illustrated in Figure 1. For the 6-class problem shown in Figure 1(a), the round robin procedure learns 15 classifiers, one for each pair of classes. Figure 1(c) shows the classifier for the class pair $<+,\sim>$. In comparison, Figure 1(b) shows one of the classifiers for the unordered class binarization, namely the one that pairs class $+$ against all other classes. It is obvious that the round robin base classifier uses fewer examples and thus has more freedom for fitting a decision boundary between the two classes. In fact, in this problem, all binary classification problems of the round robin binarization could be solved with a simple linear discriminant, while neither the multi-class problem nor its unordered binarization have a linear solution.

Note that some examples will be forced to be classified erroneously by some of the binary base classifiers because each classifier must label all examples as belonging to one of the two classes it was trained on. Consider the classifier shown in Figure 1(c): it will arbitrarily assign all examples of class o to either $+$ or $\sim$ (depending on which side of the decision boundary they are). In principle, such "unqualified" votes may lead to an incorrect final classification. However, the votes of the five classifiers that contain examples of class o should be able to over-rule the votes of the other ten classifiers, which pick one of their two constituent classes for each o example. If the class values are independent, it is unlikely that all classifiers would unanimously vote for a wrong class. However, the likelihood of such a situation could increase if there is some similarity between the correct class and some other class value (e.g., in problems with a hierarchical class structure).[1] In any case, if the five o classifiers unanimously vote for o, no other class can accumulate four votes (because they lost their direct match against o).

In the above definition, we assume that the problem of discriminating class i from class j is identical to the problem of discriminating class j from class i. This is the case if the base learner is *class-symmetric*. Rule learning algorithms, however, need not be class-symmetric. Many of them choose one of the two classes as the default class, and learn only rules to cover the other class. In such a case, $<i, j>$ and $<j, i>$ may be two different classification problems, if j is used as the default class in the former and i in the latter.

Ripper is such an algorithm. Unless specified otherwise, it will treat the larger class of a 2-class problem as the default class and learn rules for the smaller class. Although this procedure is class-symmetric (problem $<i, j>$ is converted to $<j, i>$ if $i > j$), we felt that it would not be fair. For example, the largest class in the multi-class problem would be used as the default class in all round robin problems. This may be an unfair advantage (or disadvantage) to this class. One solution for avoiding this, is a double round robin, in which separate classifiers are learned for both problems, $<i, j>$ and $<j, i>$.[2]

Definition 3.2 (double round robin) *The* double round robin *class binarization transforms a c-class problem into $c(c-1)$ 2-class problems, one for each pair of classes $<i, j>, (i, j = \dots c,\ j \neq i)$. The examples of class i are used as the positive examples and the examples of class j as the negative examples.*

In our experiments, we used Ripper with the option `-agiven` as the base classifier, which uses the classes as given in the specification. Hence, $<i, j>$ and $<j, i>$ are two different problems, and each class is the default class in exactly half of its binary classification problems.

[1]This observation is due to one of the anonymous reviewers.

[2]Another approach to tackle this problem could be to ensure that each class is used as the default class in approximately half of the classification problems.

	Ripper				
dataset	unord.	ordered	R^3	ratio	$<$
abalone	81.03	82.18	72.99	*0.888*	++
covertype	35.37	38.50	33.20	*0.862*	++
letter	15.22	15.75	7.85	*0.498*	++
sat	14.25	17.05	11.15	*0.654*	++
shuttle	0.03	0.06	0.02	*0.375*	=
vowel	64.94	53.25	53.46	*1.004*	=
car	5.79	12.15	2.26	*0.186*	++
glass	35.51	34.58	25.70	*0.743*	++
image	4.15	4.29	3.46	*0.808*	+
lr spectrometer	64.22	61.39	53.11	*0.865*	++
optical	7.79	9.48	3.74	*0.394*	++
page-blocks	2.85	3.38	2.76	*0.816*	++
solar flares (c)	15.91	15.91	15.77	*0.991*	=
solar flares (m)	4.90	5.47	5.04	*0.921*	=
soybean	8.79	8.79	6.30	*0.717*	++
thyroid (hyper)	1.25	1.49	1.11	*0.749*	+
thyroid (hypo)	0.64	0.56	0.53	*0.955*	=
thyroid (repl.)	1.17	0.98	1.01	*1.026*	=
vehicle	28.25	30.38	29.08	*0.957*	=
yeast	44.00	42.39	41.78	*0.986*	=
average	21.80	21.90	18.52	*0.770*	

Table 1. Error rates: The first three columns show the results of Ripper (in unordered and in default, ordered mode) and R^3. The last two columns show the improvement rate of R^3 over Ripper (default), and whether R^3 is significantly better (++ if $p > 0.99$, + if $p > 0.95$) than Ripper, measured with a McNemar test. The last line shows the average of the columns above.

4. Accuracy

In this section, we will briefly present an experimental evaluation of round robin binarization in a rule learning context. We chose Ripper (Cohen, 1995) as the base classifier, which—in our view—is the most advanced member of the family of separate-and-conquer (or covering) rule learning algorithms (Fürnkranz 1997; 1999).

The unordered and ordered binarization procedures were used as implemented within Ripper. The round robin binarization was implemented as a wrapper around Ripper, which provided it with the appropriate training sets. The wrapper was implemented in `perl` and had to communicate with Ripper by writing the training sets to and reading Ripper's results from the disk. This implementation is referred to as R^3 (Round Robin Ripper).

For evaluation, we arbitrarily chose 20 datasets with ≥ 4 classes from the UCI repository (Blake & Merz, 1998).[3] The implementation of the algorithm was developed independently and not tuned on these datasets. Six datasets had a dedicated test set. On the other 14 sets, we estimated the error rate using paired, stratified 10-fold cross-validations.

[3]The restriction to 4 or more classes was made because on 3-class problems, we would expect frequent 3-way ties, which are not yet handled very cleverly. The issue of ties is discussed further below in the paper (Section 6).

Table 1 shows the accuracies of Ripper (unordered and ordered) and R^3 on the selected datasets. On half of the 20 sets, R^3 is significantly better ($p > 0.99$ on a McNemar test (Feelders & Verkooijen, 1995)) than Ripper's default mode (ordered binarization). There are only two sets (*thyroid (repl.)* and *vowel*), where R^3 is worse than Ripper, both differences being insignificant. The comparison to unordered Ripper is similar (the significance levels for this case are not shown).

We can safely conclude that round robin binarization may result in significant improvements over ordered or unordered binarization without having a high risk of decreasing performance.

5. Effi ciency

At first sight, it appears to be a questionable idea to replace $O(c)$ binary learning tasks (unordered binarization) with $O(c^2)$ binary learning tasks (round robin binarization) because the quadratic complexity seems to be prohibitive for tasks with more than a few classes. This section will illustrate that this is not the case.

5.1. Theoretical Considerations

In this section, we will see that although round robin classification turns a single c-class learning problem into $c(c-1)$ 2-class problems, the total learning effort is only linear in the number of classes, and may in some circumstances even be smaller than the effort needed for an unordered binarization. The analysis independent of the base learning algorithm used. Some of the ideas have already been sketched in a short paragraph by Friedman (1996), but we go into considerably more detail here, and, in particular, focus on the comparison to conventional class binarization techniques.

Definition 5.1 (class penalty) *If the base learner has a complexity growth function $f(n)$ (i.e., the time needed for a n-example training set is $f(n)$), and the total time needed for the class binarized problem is $\pi(c)f(n)$, we call the function $\pi(c)$ the* class penalty.

Intuitively, the class penalty $\pi(c)$ measures the performance of an algorithm on a (class binarized) c-class problem relative to its performance on a single 2-class problem of equal size.

In the following, we will compare the class penalty π_r of a single round robin class binarization to class penalty π_u of an unordered binarization. Note that the class penalty for a double round robin is twice as high as the class penalty of a single round robin. Also, unordered binarization is more expensive than ordered binarization, but it is simpler to analyze because it does not depend on the class distribution. We would estimate that the ordered approach will take about half of the training time of the unordered approach.[4] So, unless noted otherwise, the ratio of the penalty functions has to be adjusted by factor of 4 to get the results of ordered binarization versus double round robin binarization.

First, we look at the class penalty π_u of unordered class binarization:

Theorem 5.2 $\pi_u(c) = c$

Proof: There are c learning tasks, each using the entire training set of n examples. Hence the total complexity is $cf(n)$, and $\pi_u(c) = c$. $\square$

Lemma 5.3 *The sum of examples in all training sets of a single round robin class binarization is $(c-1)n$.*

Proof: Each example of the original training set will occur once in each of the $c-1$ binary tasks where its class is paired against one of the other $c-1$ classes. As there are n examples in the original training set, the total number of examples is $(c-1)n$. $\square$

Note that this number is less than the total number of training examples in the unordered class binarization (i.e., cn).

In the following analysis of the class penalty π_r of a single round robin binarization, we assume that our learner has a complexity growth function $f(n) = n^o$ with $o \geq 1$.

Theorem 5.4 *For $f(n) = n^o, o \geq 1$: $\pi_r(c) \leq c-1$. The inequality is strict for $o > 1$.*

Proof: We have c classes with n_i examples, $\sum_{i=1}^{c} n_i = n$. Without loss of generality, let us assume c is even (if c is odd, we add a dummy class with $n_{c+1} = 0$ examples). Then we can arrange the learning tasks in the form of $c-1$ rounds. Each round consists of $\frac{c}{2}$ disjunct pairings, i.e., each class occurs exactly once in each round, and it has a different partner in each round. Such a tournament schedule is always possible (think sports...).

Without loss of generality, consider a round where classes $2i$ are paired against classes $2i-1$. The complexity of this round is $\sum_{i=1}^{\frac{c}{2}} f(n_{2i} + n_{2i-1})$. As for $o \geq 1$ and $a, b \geq 0$, it holds that $a^o + b^o \leq (a+b)^o$ (with equality only in the case of $o = 1$), we have $\sum_{i=1}^{\frac{c}{2}} f(n_{2i} + n_{2i-1}) \leq f(\sum_{i=1}^{\frac{c}{2}} n_{2i} + n_{2i-1}) = f(\sum_{i=1}^{c} n_i) = f(n)$.

Anologously, we can derive the same upper bound for each of the $c-1$ rounds. Thus the total complexity of the round robin binarization is $\leq (c-1)f(n)$, where the inequality is strict for $o > 1$. $\square$

[4]The ordered approach has only $c-1$ binary tasks, and the number of training examples decreases with each learned class. Our empirical results (Section 5.2) indicate that for Ripper, the actual gain is less than 50%.

Corollary 5.5 *For algorithms with at least linear complexity, the class penalty ratio* $\frac{\pi_r(c)}{\pi_u(c)} < 1$, *i.e., single round robin is more efficient that unordered binarization.*

Proof: Follows immediately from Theorems 5.2 and 5.4. □

Theorem 5.6 *Let* $\pi(c, o)$ *denote the class penalty for an algorithm of complexity* $f(n) = n^o$. *Then*

$$\lim_{o\to\infty} \frac{\pi_r(c,o)}{\pi_u(c,o)} = 0$$

Proof: In the proof of theorem 5.2 we derived an upper bound of $(c - 1)f(n)$ for the total complexity of the round robin task. Let us denote the error we made by this approximation with $\epsilon(c, o)$. We then have

$$\frac{\pi_r(c,o)}{\pi_u(c,o)} = \frac{(c-1)f(n) - \epsilon(c,o)}{cf(n)} = \frac{c-1}{c} - \frac{\epsilon(c,o)}{cf(n)}$$

The error $\epsilon(c, o)$ is the sum of the errors $\epsilon_i(c, o)$ that were made at each of the $c - 1$ rounds of the tournament described in the proof of Theorem 5.2. Let us again, without loss of generality, look at the error $\epsilon_i(c, o)$ of a round where successive classes are paired together. Then

$$\begin{aligned}\lim_{o\to\infty} \frac{\epsilon_i(c,o)}{f(n)} &= \lim_{o\to\infty} \frac{n^o - \sum_{i=1}^{\frac{c}{2}} (n_{2i} + n_{2i-1})^o}{n^o} \\ &= 1 - \sum_{i=1}^{\frac{c}{2}} \lim_{o\to\infty} \left(\frac{n_{2i} + n_{2i-1}}{n}\right)^o = 1\end{aligned}$$

because $n_{2i} + n_{2i-1} < n$. As an analogous derivation works for all $c - 1$ values of $\epsilon_i(c, o)$, we have

$$\lim_{o\to\infty} \frac{\epsilon(c,o)}{f(n)} = \sum_{i=1}^{c-1} \lim_{o\to\infty} \frac{\epsilon_i(c,o)}{f(n)} = c - 1$$

Hence:

$$\lim_{o\to\infty} \frac{\pi_r(c,o)}{\pi_u(c,o)} = \frac{c-1}{c} - \frac{1}{c} \lim_{o\to\infty} \frac{\epsilon(c,o)}{cf(n)} = 0$$

□

In practice, this means that the more expensive a learning algorithm is, the larger will be the efficiency gain in using a round robin binarization instead of an unorderd binarization. In particular, this theorem implies that for complex algorithms even the double round robin may be faster than the ordered binarization.

For example, it is straight-forward to show that in a class-balanced eight-class problem, an algorithm of quadratic complexity has a penalty ratio of $\frac{7}{32} < \frac{1}{4}$ (Fürnkranz, 2001). Thus, under these circumstances, the single round robin is more than four times faster than the unordered approach. Considering that the double round robin is twice as slow, and assuming that the ordered approach is twice as fast as the unordered approach (see the following section for empirical values on that), a double round robin would be faster than the ordered approach. In the following section, this scenario will be empirically evaluated.

dataset	R^3	vs. unordered	vs. ordered
abalone	193.0	*4.51*	*5.73*
covertype	—	—	—
letter	1250.0	*0.51*	*1.14*
sat	143.0	*0.85*	*1.51*
shuttle	277.0	*2.10*	*3.16*
vowel	14.0	*1.83*	*4.75*
car	6.71	*1.55*	*1.47*
glass	2.03	*2.26*	*3.80*
image	25.84	*0.90*	*1.98*
lr spectrometer	489.67	*4.40*	*6.93*
optical	275.69	*0.63*	*1.34*
page-blocks	36.66	*1.43*	*1.93*
solar flares (c)	6.65	*6.03*	*7.57*
solar flares (m)	3.98	*5.62*	*7.49*
soybean	21.07	*6.29*	*13.24*
thyroid (hyper)	19.71	*2.68*	*3.46*
thyroid (hypo)	14.91	*2.39*	*3.63*
thyroid (repl.)	15.35	*2.26*	*3.33*
vehicle	7.66	*1.22*	*2.10*
yeast	16.90	*1.77*	*3.12*
average		*2.59*	*4.09*

Table 2. Runtime results: The first column shows the run-times (in CPU secs. user time) of R^3. The following columns show the ratio of R^3 over unordered and ordered Ripper. The first five lines are total run-times, i.e., training and test time, while the cross-validated results report training time only. We failed to measure the run-times for the covertype data set, where the situation was complicated because of the large test set, which had to be split into several pieces.

5.2. Empirical Evaluation

Contrary to the theoretical analysis in the previous section, where we focussed on the lenient case of pairing unordered binarization vs. single round robin, our empirical results show the performance of ordered binarization (Ripper's default mode) vs. double round robin binarization. This, as we have noted above, is the worst case. In the case of a linear algorithm complexity, the round robin should be about four times slower than the ordered binarization.

Table 2 shows the training times[5] in CPU secs. user time (measured on a Sparc Ultra-2 under Sun Solaris) of R^3 and its performance ratios against Ripper in unordered and ordered mode. On average, it is about 2.6 times slower than Ripper in unordered mode, and about 4 times slower than Ripper in default, ordered mode. Coincidentally, these

[5]Classification time is only included in the runs that had a separate test set. In general, it can be expected to be more expensive for R^3. See Section 6 for a brief discussion of this issue.

numbers also show that the ordered mode is less than twice as fast as the unordered mode, which confirms the assumptions we made at the beginning of Section 5.1.

Moreover, there are several cases where R^3 is even faster than Ripper in unordered mode and comes close to Ripper in ordered mode. This is despite the fact that R^3 is implemented as a `perl`-script that communicates to Ripper by writing the training and test sets of the new tasks to the disk. Although this is somewhat compensated by the fact that we only report user time (which ignores time for disk access and system time),[6] a tight integration of round robin binarization into Ripper's C-code would certainly be more efficient.

The good performance of R^3 does not come entirely surprising, if we consider the super-linear run-time complexity of Ripper.[7] For more expensive base learning algorithms (like support vector machines), the analysis in the previous section lets us expect even bigger savings.

6. Other Issues

R^3 as an Ensemble Method: In (Fürnkranz, 2001) we compared round robin binarization to boosting. In particular, we compared the performance improvement obtained by C5-boost over C5 to the improvement obtained by R^3 over Ripper. It turned out that both algorithms seemed to be apt for similar types of problems. Round robin binarization seemed to work well whenever boosting worked well, and vice versa. Figure 2 plots the error ratios of C5-boost/C5 and R^3/Ripper on the 20 datasets. The correlation between the improvement ratios was 0.618, which is in the same range as correlation coefficients for bagging and boosting (Opitz & Maclin, 1999). However, in terms of efficiency, C5-boost was about 8.75 times slower than C5, which is not surprising as it basically calls C5 10 times on different samples of the original dataset.

Parallel Implementations: It should be noted that—contrary to boosting, where the individual runs depend on each other and have to be performed in succession—pairwise classification can be entirely parallelized (as already noted by Friedman (1996) and Lu and Ito (1999)).

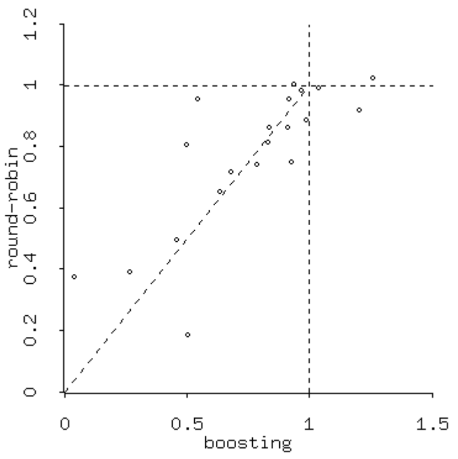


Figure 2. Error reductions ratios of boosting vs. round robin.

As each binary task will be smaller than the original task, the total training time of a multi-class problem of size n will be significantly below that of a binary problem of the same size, if each binary classifier can be trained on a separate processor.

Memory Requirements: It is also clear that each individual binary task in a round-robin binarization has less training examples than the original tasks. For multi-class tasks that are too large to be performed in memory, pairwise classification may provide a simple means to reduce the size of the learning task without resorting to subsampling. Note that this is not the case for unordered class binarization or error-correcting output codes.

Classification Efficiency: Our efficiency analysis is only concerned with training time. At testing time, we have to test a quadratic number of classifiers in order to make the final prediction. Although it might be the case that the constituent classifiers are simpler (which often means that they can make faster predictions) it can be expected that classification time would be considerably slower than in the unordered binarization case. This situation is particularly bad for lazy learners, which defer most of their training effort to the classification phase. A solution for this problem could be found in the proposal of Platt et al. (2000) who suggest to organize pairwise classifiers in a decision graph where each node represents a binary classifier. They show that this structure allows to obtain a classification for c-class problems by consulting only $c-1$ pairwise classifiers without loss of accuracy on three benchmarks problems.

Tie Breaking: We have mostly ignored the issue of tie-breaking, which is necessary when several classes have an equal number of wins against other classes. In particular for a low number of classes, ties are more likely because the ensemble is much smaller (which was the reason why we restricted our study to problems with at least four classes).

[6] For example, on the 26-class *letter* dataset, where R^3 writes $26 \times 25 = 650$ files to the disk, its total run-time is about 15% higher than the reported user time, while there is almost no difference for Ripper.

[7] The complexity of Ripper's initial rule learning and pruning phase is $O(n \log^2(n))$ (Fürnkranz, 1997; Cohen, 1995). The two phases of optimization that Ripper performs thereafter, only seem to add a constant factor to these results according to the experimental evidence shown in (Cohen, 1995). However, in very large domains, like text domains, our own experience is that these two optimization phases can slow down the algorithm considerably, and seem to dominate the run-time (but we have not performed a thorough analysis of this issue).

Our straight-forward approach of using the *a priori* more likely class, can certainly be improved upon. In addition to techniques known from the literature (see, e.g., (Hastie & Tibshirani, 1998; Allwein et al., 2000)), one could think of exploring techniques that are commonly used for breaking ties in tournament cross tables in games and sports (such as the Sonneborn-Berger ranking in chess tournaments).

Imbalanced Class Distributions: Although we have not yet evaluated this issue, we also believe that round robin binarization provides a better focus on minority classes, in particular in problems where several large classes appear next to a few small classes. The fact that separate classifiers are trained to discriminate the small classes from each other (and not only from all remaining examples as would be the case for unordered binarization or for treating the multi-class problem as a whole) may help to improve the focus in the case of imbalanced class distributions.

Comprehensibility: While boosting also provides similar gains in accuracy, the price to pay is that the learned ensemble of classifiers is no longer easy to comprehend.[8] While round robin rule learning also learns an ensemble of classifiers, we think that it has the advantage that each element of the ensemble has a well-defined semantics (separating two classes from each other). In fact, Pyle (1999) proposes a very similar technique called *pairwise ranking* in order to facilitate human decision-making in ranking problems. He claims that it is easier for a human to determine an order between n items if s/he makes pairwise comparisons between the individual items and then adds up the wins for each item, instead of trying to order the items right away.[9]

7. Related Work

Pairwise classification was introduced to the machine learning literature by Friedman (1996) but the idea seems to have been known for some time. Hastie and Tibshirani (1998) picked up the technique and introduced *pairwise coupling*, a tie-breaking technique which combines the class probability estimates from the binary classifiers into a joint probability distribution for the multi-class problem. Pairwise classification was soon applied in the support vector (Schmidt, 1996; Kreßel, 1999) and neural networks communities (Lu & Ito, 1999). In comparison to these works, our study provides a much broader empirical evaluation of round robin learning, as well as a detailed theoretical analysis. It is also the first evaluation of this technique in the context of inductive rule learning.

Unordered and ordered class binarization techniques are not the only alternatives. Error-correcting output codes (Dietterich & Bakiri, 1995) are probably the most popular alternative. It encodes a c-class problem as $\bar{c}$ binary problems ($\bar{c} > c$). As the number of required binary problems is $> c$ for c-class problems, its penalty function $\pi_{eoc} > c$, i.e., pairwise and unordered binarization are more efficient. Most recently, Allwein et al. (2000) provide a unifying view of various class binarization techniques and derive some theoretical error bounds. They also provide a thorough experimental study of five different class binarization schemes with three different techniques for combining the predictions of the binary classifiers.

Unfortunately, space restrictions prevent an in-depth discussion of the relevant literature in this paper. We have to refer the reader to (Fürnkranz, 2001).

8. Summary and Outlook

This paper has investigated the use of round robin binarization (or pairwise classification) as a technique for handling multi-class problems with separate-and-conquer rule learning algorithms (aka covering algorithms). Our experimental results show that, in comparison to conventional, ordered or unordered binarization, the round robin approach may yield significant gains in accuracy without risking a bad performance. We think that the reason for this improvement lies on the one hand in the exploitation of diverse predictions in an ensemble of classifiers and, on the other hand, in the fact that the resulting binary problems may be considerably simpler and can thus be learned more reliably (Figure 1(c)).

Moreover, we demonstrated both empirically and theoretically that the quadratic growth in the number of learning problems is compensated by the reduction in size for each of the individual problems. For algorithms with a super-linear run-time, round robin binarization is even faster than conventional unordered technique, and we have proven that this advantage increases with the complexity of the base algorithm. Our experimental results confirm the efficiency of round robin binarization, but for a true evaluation of the performance of this technique, an efficient, tight integration of the technique into a separate-and-conquer rule learning algorithm would be highly desirable.

There are several issues that still need to be addressed. First, we want to investigate whether the correlation in the performance gains of boosting and round robin binarization prohibits an effective combination of these two techniques.

[8] An exception might be a system like Slipper (Cohen & Singer, 1999) which tightly integrates boosting into the rule learning algorithm with the effect that only a single set of rules is learned and the redundancy among these rules is exploited to obtain higher accuracies. A direct comparison between Slipper and R^3 would be very interesting, not only for this reason.

[9] The aspect of being able to rank the available classifi cations for each example (as an intermediate version between predicting only a class value and providing a full probability distribution) is another interesting aspect of round robin binarization, which might be worth exploring in more depth.

To this end, we plan to evaluate round robin binarization using C5 and C5-boost as base learners. Alternatively, a direct comparison of R^3 to Slipper (Cohen & Singer, 1999) could provide evidence to answer this question.

The main disadvantage of the approach, of course, is its dependency on the number of classes. More classes result in a bigger ensemble which should produce better predictions. In particular in the limiting case, where only two or three classes are available, the benefits should be rather small. This trade-off also needs to be investigated in more detail.

Acknowledgements

I wish to thank the anonymous reviewers, William Cohen, Eibe Frank, Stefan Kramer, Johann Petrak, Lupčo Todorovski, Gerhard Widmer, and the maintainers of and contributors to the UCI collection of databases for discussions, programs, databases, and pointers to relevant literature. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. The author was supported by the ESPRIT long-term project METAL (26.357).

References

Allwein, E. L., Schapire, R. E., & Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, *1*, 113-141.

Anand, R., Mehrotra, K. G., Mohan, C. K., & Ranka, S. (1995). Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks*, *6*, 117–124.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Department of Information and Computer Science, University of California at Irvine. Irvine, CA.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. *Proceedings of the 5th European Working Session on Learning (EWSL-91)* (pp. 151–163). Porto, Portugal: Springer-Verlag.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–283.

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the 12th International Conference on Machine Learning (ML-95)* (pp. 115–123). Lake Tahoe, CA: Morgan Kaufmann.

Cohen, W. W., & Singer, Y. (1999). A simple, fast, and effective rule learner. *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)* (pp. 335–342). Menlo Park, CA: AAAI/MIT Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Feelders, A., & Verkooijen, W. (1995). Which method learns most from the data? Methodological issues in the analysis of comparative studies. *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*. Fort Lauderdale, Florida.

Friedman, J. H. (1996). *Another approach to polychotomous classification* (Technical Report). Department of Statistics, Stanford University, Stanford, CA.

Fürnkranz, J. (1997). Pruning algorithms for rule learning. *Machine Learning*, *27*, 139–171.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13*, 3–54.

Fürnkranz, J. (2001). *Round robin rule learning* (Technical Report OEFAI-TR-2001-02). Austrian Research Institute for Artificial Intelligence, Wien, Austria.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *Advances in Neural Information Processing Systems 10 (NIPS-97)* (pp. 507–513). MIT Press.

Kreßel, U. H.-G. (1999). Pairwise classification and support vector machines. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods: Support vector learning*, 255–268. Cambridge, MA: MIT Press.

Lu, B.-L., & Ito, M. (1999). Task decomposition and module combination based on class relations: A modular neural network for pattern classification. *IEEE Transactions on Neural Networks*, *10*, 1244–1256.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Platt, J. C., Cristianini, N., & Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems 12 (NIPS-99)* (pp. 547–553). MIT Press.

Pyle, D. (1999). *Data preparation for data mining*. San Francisco, CA: Morgan Kaufmann.

Schmidt, M. S. (1996). Identifying speakers with support vector networks. *Proceedings of the 28th Symposium on the Interface (INTERFACE-96)*. Sydney, Australia.