Journal of Machine Learning Research 2 (2002) 721-747 Submitted 08/01; Published 03/02

Round Robin Classification

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

Editor: Yoram Singer

Abstract

In this paper, we discuss round robin classification (aka pairwise classification), a technique for handling multi-class problems with binary classifiers by learning one classifier for each pair of classes. We present an empirical evaluation of the method, implemented as a wrapper around the Ripper rule learning algorithm, on 20 multi-class datasets from the UCI database repository. Our results show that the technique is very likely to improve Ripper's classification accuracy without having a high risk of decreasing it. More importantly, we give a general theoretical analysis of the complexity of the approach and show that its run-time complexity is below that of the commonly used one-against-all technique. These theoretical results are not restricted to rule learning but are also of interest to other communities where pairwise classification has recently received some attention. Furthermore, we investigate its properties as a general ensemble technique and show that round robin classification with C5.0 may improve C5.0's performance on multi-class problems. However, this improvement does not reach the performance increase of boosting, and a combination of boosting and round robin classification does not produce any gain over conventional boosting. Finally, we show that the performance of round robin classification can be further improved by a straight-forward integration with bagging.

Keywords: pairwise classification, inductive rule learning, multi-class problems, class binarization, ensemble techniques

1. Introduction

Although real-world problems often have multiple classes, many learning algorithms are inherently binary, i.e., they are only able to discriminate between two classes. The reasons for this may be constraints imposed by the hypothesis language (e.g., linear discriminants or support vector machines), the learning architecture (e.g., neural networks with single output nodes), or the learning framework (e.g., many rule learning algorithms are tailored towards concept learning, i.e., the problem of learning a concept description from positive and negative examples). There are two principal approaches for applying such algorithms to multi-class problems: one approach is to generalize the algorithm—as has, e.g., been done for support vector machines (Weston and Watkins, 1999; Mayoraz and Alpaydin, 1999) or boosting (Freund and Schapire, 1997)—the other is to employ class binarization techniques (Section 2), which reduce the multi-class problem into a series of binary problems.

One of the most common class binarization approaches is to learn separate concept descriptions for each individual class, i.e., to form a series of problems, one for each class, where all examples of this class are regarded as positive examples, while all other examples are regarded as negative. Pairwise classification is an alternative class binarization technique, which has lately received some attention in the neural networks and support vector machines communities (Section 8). Its basic idea is to reduce a multi-class problem to multiple two-class problems by learning one classifier for each pair of classes, using only training examples for these two classes and ignoring all others (Section 3).

In this paper, we will, on the one hand, evaluate the technique—which we name *round robin* binarization—for inductive rule learning algorithms and show that it is significantly more accurate than the ordered and unordered (one-against-all) techniques that are currently used for rule learning algorithms like Ripper (Section 4). On the other hand, we provide a detailed analysis of the complexity of the approach and show that the $c(c-1)/2$ learning problems of a single round robin can be learned more efficiently than the c learning problems of the conventional one-against-all technique (Section 5). This analysis is independent of the base learning algorithm, but we show that the advantage increases with more expensive learners. The theoretical results are confirmed by our experimental evaluation. In Section 6, we will investigate the properties of round-robin classification as a general ensemble technique with mixed results: while the technique is able to reduce C5.0's error on average, the gains do not approach those of boosting. On the other hand, a straightforward combination of round robin learning with bagging does lead to additional gains for Ripper. Finally, we will discuss a few other relevant properties of round-robin classification, including its suitability for parallel implementations, possible remedies for its inefficiency at classification time, and its comprehensibility (Section 7).

2. Class Binarization

Many machine learning algorithms are inherently designed for binary (two-class) decision problems. Prominent examples are perceptrons, support vector machines, the original AdaBoost algorithm, 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 can be ordered). On the other hand, 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 1 (class binarization, decoding, base learner) *A* class binarization *is a mapping of a multi-class learning problem to several two-class learning problems in a way that allows a sensible* decoding *of the prediction, i.e., it allows the derivation of a prediction for the multi-class problem from the predictions of the set of two-class classifiers. The learning algorithm used for solving the two-class problems is called the* base learner.

The most popular class binarization technique 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 and Boswell, 1991), neural networks (Anand et al., 1995), and support vector machines (Cortes and Vapnik, 1995).

Definition 2 (unordered/one-against-all class binarization) *The* unordered class binarization *transforms a c-class problem into c two-class problems. These are constructed by using the examples of class i as the positive examples and the examples of classes j ($j = 1 \ldots c$, $j \neq i$) 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 and Niblett, 1989; Rivest, 1987). As our main concern is rule learning, we will primarily stick to the terminology used there, but will also occasionally refer to it as *one-against-all*, which seems to be predominant in other fields.

Definition 3 (ordered class binarization) *The* ordered class binarization *transforms a c-class problem into $c-1$ binary problems. These are constructed by using the examples of class i ($i = 1 \ldots c-1$) as the positive examples and the examples of classes $j > i$ as the negative examples.*

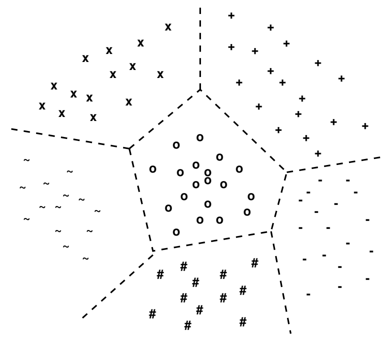
Note that ordered class binarization imposes an order on the induced classifiers, which has to be adhered to at classification time: the classifier learned for discriminating class 1 from classes $2 \ldots c$ has to be called first. If this classifier classifies the example as belonging to class 1, no 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 combining 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 (cf. Section 7).

3. Round Robin Classification

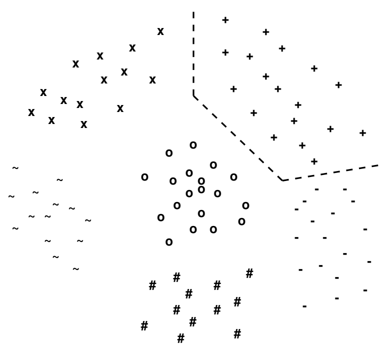
In this section, we will discuss a more complex class binarization procedure, the pairwise classifier. The basic idea is quite simple, namely to learn one classifier for each pair of classes. In analogy to round robin tournaments in sports and games, in which each participant is paired with each other participant, we call this procedure *round robin binarization*.[1]

Definition 4 (round robin/pairwise binarization) *The* round robin *or* pairwise *class binarization transforms a c-class problem into $c(c-1)/2$ two-class problems $<i,j>$, one for each set of classes $\{i,j\}, i = 1 \ldots c-1,\ j = i+1 \ldots c$. The binary classifier for problem $<i,j>$ is trained with examples of classes i and j, whereas examples of classes $k \neq i,j$ are ignored for this problem.*

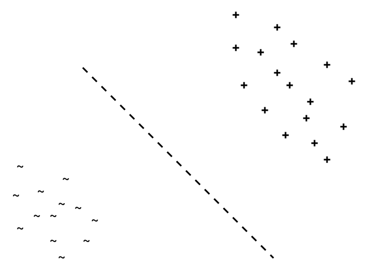
1. The etymology of this phrase is interesting and unclear. According to *The Shorter Oxford English Dictionary* (1973), the phrase has at one time or another referred to the fish *decapterus punctatus*, to the Holy Sacrament (in a blasphemous way), and to written complaints that were signed in a circular way to disguise the order in which the subscribers have signed the petition. A posting on the usenet group `uk.culture.language.english` (thanks to Foster Provost for this information) claims that its current use in games and sports goes back to these circular signatures, and that the phrase originates from the French word *ruban*, ribbon. Finally—on a Web-site that is dedicated to analyzing lyrics from the *Grateful Dead*—Psychology Professor Tom Malloy (Rhode Island College) points out that the use of round robin in the song *The Wheel* might also refer to a research design used in biology and psychology where pairwise reactions of a group of subjects to each other's behaviors (e.g., smiling) are measured.



(a) *Multi-class learning*
one classifier that separates all classes.



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



(c) *Round robin learning*
$c(c-1)/2$ classifiers, one for each pair of classes. Here: $+$ against $\sim$.

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

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 $<+, \sim>$. 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 in the round robin case, the base classifier uses fewer examples and thus has more freedom for fitting a decision boundary between the two classes. In fact, in our example, 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. The phenomenon that pairwise decision boundaries can be considerably simpler than those originating from unordered binarization has also been observed in real-world domains. For example, Knerr et al. (1992) observed that the classes of a digit recognition task were pairwise linearly separable, while the corresponding one-against-all task was not amenable to single-layer networks. The results of Hsu and Lin (2002), who obtained a larger advantage of round robin binarization over unordered binarization for support vector machines with a linear kernel than for support vector machines with a non-linear kernel on several benchmark problems, could also be explained by simpler decision boundaries in the round robin case.

A crucial point, of course, is how to decode the predictions of the pairwise classifiers to a final prediction. We implemented a simple voting technique: 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. We break ties by preferring the class that is more frequent in the training set (or flipping a coin if the frequencies are equal). 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 overrule 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). In any case, if the five o classifiers unanimously vote for o, no other class can accumulate five votes (because they lost their direct match against o). Nevertheless, this simple voting procedure is certainly suboptimal. We will discuss alternative decoding techniques in Section 7.

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. A straightforward approach for addressing this problem is to play a so-called double round robin, in which separate classifiers are learned for both problems, $<i,j>$ and $<j,i>$.[2]

Definition 5 (double round robin) *The* double round robin *class binarization transforms a c-class problem into $c(c-1)$ two-class problems $<i,j>$, one for each pair of classes $(i,j), i,j = 1 \ldots c$, $j \neq i$. The examples of class i are used as the positive examples and the examples of class j as the negative examples.*

In the following section, we will evaluate a double round robin as an alternative binarization strategy for the rule learner Ripper.

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 algorithm of the family of separate-and-conquer (or covering) rule learning algorithms (Fürnkranz, 1997; 1999b).

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.

Table 1: *Datasets used.* The first two columns show the training and test set sizes (as specified in the description of the datasets), the next three columns show the number of symbolic and numeric attributes as well as the number of classes. The last column shows the default error, i.e., the error one would get by always predicting the majority class.

name	train	test	sym	num	classes	def. error
abalone	3133	1044	1	7	29	83.5
car	1728	—	6	0	4	30.0
covertype	15,120	565,892	44	10	7	51.1
glass	214	—	0	9	7	64.5
image	2310	—	0	19	7	85.7
letter	16,000	4000	0	16	26	95.9
lr spectrometer	531	—	1	101	48	89.6
optical	5620	—	0	64	10	89.8
page-blocks	5473	—	0	10	5	10.2
sat	4435	2000	0	36	6	76.2
shuttle	43,500	14,500	0	9	7	21.4
solar flares (c)	1389	—	10	0	8	15.7
solar flares (m)	1389	—	10	0	6	4.9
soybean	683	—	35	0	19	94.1
thyroid (hyper)	3772	—	21	6	5	2.7
thyroid (hypo)	3772	—	21	6	5	7.7
thyroid (repl.)	3772	—	21	6	4	3.3
vehicle	846	—	0	18	4	74.2
vowel	528	462	0	10	11	90.9
yeast	1484	—	0	8	10	68.8

The unordered and ordered binarization procedures were used as implemented within Ripper. Ordered binarization is Ripper's default mode, and unordered binarization can be invoked using the option `-a unordered`. 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).

In principle, Ripper is a class-symmetric learner. It will treat the larger class of a two-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 $|c_i| > |c_j|$), 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.[3] For this reason, R^3 implements a double round robin binarization which calls Ripper with the option `-a given` on each binary problem $<i,j>$

3. The situation can be compared to a chess player that has to play all of his games with the same color or a football team that is allowed to play all (or none) of its games in the home stadium. This can make a decisive difference and may invalidate the final result of the tournament.

Table 2: *Error rates*: The first two column pairs show the results of Ripper (in unordered and in default, ordered mode). For both, we show the absolute error rate, and the improvement rate of R^3 over the algorithm. R^3's error rates are shown in the 5th column. The last column shows whether the difference between R^3 and default Ripper (ordered) is significant (++ if $p > 0.99$, + if $p > 0.95$, McNemar test). The first part of the table shows the results for cross-validation (17 sets). To allow a comparison with previous works, the last 6 lines show hold-out estimates for the six datasets with a designated test set (3 datasets occur in both blocks). The line in the middle shows the geometric averages of the improvement ratios.

	Ripper				R^3	McNemar
dataset	unord.	ratio	ordered	ratio		test
abalone	81.64	*0.911*	81.18	*0.916*	74.34	++
car	5.79	*0.390*	12.15	*0.186*	2.26	++
glass	35.51	*0.724*	34.58	*0.743*	25.70	++
image	4.15	*0.823*	4.29	*0.808*	3.46	+
lr spectrometer	64.22	*0.827*	61.39	*0.865*	53.11	++
optical	7.79	*0.479*	9.48	*0.394*	3.74	++
page-blocks	2.85	*0.968*	3.38	*0.816*	2.76	++
sat	13.18	*0.785*	13.04	*0.794*	10.35	++
solar flares (c)	15.91	*0.991*	15.91	*0.991*	15.77	=
solar flares (m)	4.90	*1.029*	5.47	*0.921*	5.04	=
soybean	8.79	*0.717*	8.79	*0.717*	6.30	++
thyroid (hyper)	1.25	*0.893*	1.49	*0.749*	1.11	+
thyroid (hypo)	0.64	*0.833*	0.56	*0.955*	0.53	=
thyroid (repl.)	1.17	*0.863*	0.98	*1.026*	1.01	=
vehicle	28.25	*1.029*	30.38	*0.957*	29.08	=
vowel	30.50	*0.633*	27.07	*0.690*	18.69	++
yeast	44.00	*0.949*	42.39	*0.986*	41.78	=
average		*0.787*		*0.747*		
abalone	81.03	*0.901*	82.18	*0.888*	72.99	++
covertype	35.37	*0.939*	38.50	*0.862*	33.20	++
letter	15.22	*0.516*	15.75	*0.498*	7.85	++
sat	14.25	*0.782*	17.05	*0.654*	11.15	++
shuttle	0.03	*0.667*	0.06	*0.375*	0.02	=
vowel	64.94	*0.823*	53.25	*1.004*	53.46	=

$(i, j = 1 \ldots c, i \neq j)$. This instructs Ripper to use the classes in the order specified by the user (i.e., the order in which they appear in the names file). Hence, $<i, j>$ and $<j, i>$ are two different problems, which ensures that each class is the default class in exactly half of its binary classification problems. Note, that this procedure is basically identical to the one that is employed by Ripper if it is used in unordered mode on a two-class problem except that Ripper would tie-break immediately between the theories learned for $<i, j>$ and $<j, i>$, while we first collect all votes from all $c(c-1)$ binary problems.

Table 1 shows the 20 datasets we used in this study. They were chosen arbitrarily among datasets with ≥ 4 classes available at the UCI repository (Blake and Merz, 1998).[4] The implementation of the algorithm was developed independently and not tuned on these datasets. On the six sets with a dedicated test set, we report the error rate on these test sets. On the other 14 sets, we estimated the error rate using paired, stratified 10-fold cross-validations. For *abalone*, *sat* and *vowel* we performed both a test set evaluation and a cross-validation.[5]

Table 2 shows the accuracies of Ripper (unordered and ordered) and R^3 on the selected datasets. On half of the 20 experiments (not counting the cross-validated trials of the three sets the re-appear at the bottom), R^3 is significantly better ($p > 0.99$ on a McNemar test[6]) than Ripper's default mode (ordered binarization). There were only two experiments (*thyroid (repl.)* and the test-set version of *vowel*), where R^3 is worse than Ripper, both differences being insignificant. On the 17 cross-validated problems, R^3 reduces the average error to 75% of the error of Ripper's error.[7] 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. Efficiency

At first sight, it appears to be a questionable idea to replace c binary learning tasks (unordered binarization) with $c(c-1)/2$ 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 (single) round robin classification turns a single c-class learning problem into $c(c-1)/2$ two-class problems, the total training effort is only linear in the number of classes and smaller than the effort needed for an unordered binarization. The analysis is independent of the type of base learning algorithm used, although we will show that the advantage increases with the computational complexity of the algorithm. Some of the ideas have already been sketched in a short paragraph by Friedman (1996), but we go into considerably more detail and, in particular, focus on the comparison to conventional class binarization techniques.

4. 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 7).
5. As we can see in Table 2, there are no qualitative differences between hold-out and cross-validation for *abalone* and *sat*. For *vowel* the performance of all algorithms was significantly better in the case of cross-validation, which may indicate that the 528 examples in the original training set are insufficient for learning a good concept.
6. The McNemar test (McNemar, 1947) tests for significant differences of proportions in paired sample designs. It is appropriate for comparing classifiers because it does not assume independent samples and has a comparably low Type I error (Feelders and Verkooijen, 1995; Dietterich, 1998).
7. As these are relative performance measures, we use a geometric average so that x and $1/x$ average to 1.

In the following, we assume a base learner with a time complexity function $f(n)$, i.e., the time needed for learning a classifier from a training set with n examples is $f(n)$. Note that we interpret this as an *exact* function, and not as an asymptotically tight bound as in $\Theta(f(n))$ because we are not interested in asymptotic behavior, but in the exact complexity at a given training set size n.[8] We will consider functions of the form $f(n) = \lambda n^p$ $(p \geq 1, \lambda > 0)$ and denote such a function with f_p. We use $b|f$ to denote a class binarization with algorithm b, where an base learner with complexity $f(n)$ is applied to each binary problem. Unless mentioned otherwise, all results refer to single round robin binarizations of problems with more than two classes $(c > 2)$.

Definition 6 (class penalty) *Let $g_{b|f}(c,n)$ be the total complexity for using a learner with complexity $f(n)$ on a problem with $c > 2$ classes that has been class-binarized using binarization algorithm b. We then define the* class penalty *function $\pi_{b|f}$ as*

$$\pi_{b|f}(c,n) = \frac{g_{b|f}(c,n)}{f(n)}$$

Intuitively, the class penalty $\pi_{b|f}(c,n)$ measures the performance of an algorithm on a class binarized c-class problem relative to its performance on a single two-class problem the same size n. In the following, it will turn out that in some cases the class penalty function is independent of n or f. In such cases, we will abbreviate the notation as $\pi_b(c)$.

Lemma 7 (class penalty for unordered class binarization)
$\pi_u(c) = c$

Proof There are c learning tasks, each using the entire training set of n examples. Hence the total complexity $g_{u|f}(c,n) = cf(n)$, and $\pi_u(c) = c$. ∎

Lemma 8 (class penalty for single round robin, linear base algorithm)
For a base learner with a linear run-time complexity $f_1(n) = \lambda n$: $\pi_{r|f_1}(c) = c - 1$.

Proof Each example of the original training set will occur exactly 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 $(c-1)n$.

As f is linear, the sum of the complexities on all individual training sets is equal to the complexity of the algorithm on the sum of all training set sizes, i.e., $\sum f_1(n_i) = f_1(\sum n_i)$. Hence,

$$g_{r|f_1}(c,n) = f_1((c-1)n) = \lambda(c-1)n = (c-1)(\lambda n) = (c-1)f_1(n)$$

8. This is a very strong assumption that will in general not hold up in practice. Strictly speaking, the time needed for training a classifier does not only depend on the sample size, but also on the domain and the "difficulty" of the sample for the learner. Even in the same domain, samples of equal sizes may yield theories of different complexities, and more complex theories will in general require longer training times. One way to incorporate this aspect could be to interpret our complexity functions as average run-times over all possible subsets of this size (in a given domain). However, we believe that making the simplifying assumption of an exact complexity function does not invalidate our claims, which are also backed up by empirical results (Section 5.2).

Therefore $\pi_{r|f_1}(c) = c - 1$. ■

This analysis ignores a possible constant overhead of the algorithm, which potentially affects $c(c-1)/2$ function calls in the round robin case, while it only affects c function calls in the unordered case. However, some significant overhead costs, like reading in the training examples (and similar initialization steps) need, of course, only be performed once if the round robin procedure is performed in memory (which was not the case in the implementation which we used for the experimental results reported in the next section). If there is an overhead μ to be considered (i.e., $f(n) = \lambda n + \mu$), the total costs will be increased by $\mu c(c-1)/2$. For very large values of c, these quadratic costs may outweigh the savings, but under reasonable assumptions (e.g., $c^2 < n$) these additive costs should not matter, in particular—as we shall see in the following—not in the case of super-linear base algorithms.

Lemma 9 (class penalty for single round robin, super-linear base algorithm)
For $p > 1$: $\pi_{r|f_p}(c,n) < \begin{cases} c-1 & \textit{if } c \textit{ is even} \\ c & \textit{if } c \textit{ is odd} \end{cases}$

Proof
Assume we have c classes, class i has n_i examples, $\sum_{i=1}^{c} n_i = n$.

c is even:
In this case, we can arrange the learning tasks in the form of $c-1$ rounds. Each round consists of $c/2$ disjoint 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.[9] Without loss of generality, consider a round where classes $2i$ are paired against classes $2i-1$. The complexity of this round is $\sum_{i=1}^{c/2} f_p(n_{2i} + n_{2i-1})$. As for $p > 1$ and $a_i > 0$, $i = 1 \ldots N$ it holds that $\sum_i a_i^p < (\sum_i a_i)^p$, and because we assumed $c > 2$, we get

$$\sum_{i=1}^{\frac{c}{2}} f_p(n_{2i} + n_{2i-1}) < f_p(\sum_{i=1}^{\frac{c}{2}} n_{2i} + n_{2i-1}) = f_p(\sum_{i=1}^{c} n_i) = f_p(n)$$

Analogously, we can derive the same upper bound for each of the $c-1$ rounds. Thus the total complexity of the round robin binarization $g_{r|f_p}(c,n) < (c-1)f_p(n)$ and $\pi_{r|f_p}(c,n) < c-1$.

c is odd:
we add a dummy class with $n_{c+1} = 0$ examples, and perform a tournament as above. As this tournament has c rounds, $\pi_{r|f_p}(c,n) < c$.

■

Note that the upper bounds in Lemma 9 are not tight (see also Theorems 11 and 12 below).

9. A simple algorithm for constructing such a tournament is to fix one player, say the one with index c (assume c is even, add a dummy player if it is not). Then construct a tournament schedule for the first round by pairing player i with player $(c+1) - i$, $i = 1 \ldots c/2$. Subsequent rounds are played with the same pairings, but before each round, the first $c-1$ entries rotate places, i.e., player i takes over the place of $i+1$ $(i = 1 \ldots c-2)$ and $c-1$ moves to 1. It is both, fairly trivial to see that this is correct, and quite easy to put into practice when organizing a round robin chess tournament for kids (let one stick to its seat and all others move around one seat at a time).

In particular, the bound of $c-1$ should also hold for odd numbers but the proof for that case seems to be considerably more tricky.[10] However, the current version suffices to prove the following theorem:

Theorem 10 (efficiency of round-robin and unordered binarization)
For algorithms with at least linear complexity ($p \geq 1$): $\pi_{r|f_p}(c,n) < \pi_{u|f_p}(c,n)$, i.e., single round robin is more efficient than unordered binarization.

Proof Follows immediately from Lemmas 7, 8 and 9. ■

As mentioned above, the bounds used for proving the lemmas are certainly not tight. This becomes obvious, if we look at problems with a uniform class distribution.

Theorem 11 (class penalty for class-balanced problems)
For a class-balanced problem, $\pi_{r|f_p}(c) = (c-1)(\frac{2}{c})^{p-1}$

Proof In the (single) round robin case, we have $c(c-1)/2$ problems with $2n/c$ examples each. Hence the total complexity is

$$\begin{aligned} g_{r|f_p}(c,n) &= \frac{c(c-1)}{2} f_p\left(2\frac{n}{c}\right) = (c-1)\frac{c}{2}\lambda\left(2\frac{n}{c}\right)^p = (c-1)\left(\frac{2}{c}\right)^{p-1}\lambda n^p = \\ &= (c-1)\left(\frac{2}{c}\right)^{p-1} f_p(n) \end{aligned}$$

Therefore $\pi_{r|f_p}(c) = (c-1)(\frac{2}{c})^{p-1}$. ■

From this result, it is easy to see that $\pi_{r|f_p}(c,n)$ decreases with increasing complexity order p of the base learner (assuming $c > 2$). Likewise, for $p > 2$, $\pi_{r|f_p}(c,n)$ can be made arbitrarily small by increasing the number of classes c. While the latter property is hard to generalize for arbitrary class distributions—it is not the case that every problem with more than c classes has a smaller class penalty than an arbitrary c-class problem (and vice versa)—it is easy to prove the following theorem:

Theorem 12 (class penalty ratio for increasing order of base algorithm)
For $c > 2$, the class penalty ratio $\frac{\pi_{r|f_p}(c,n)}{\pi_{u|f_p}(c,n)}$ is monotonically decreasing with p and

$$\lim_{p\to\infty} \frac{\pi_{r|f_p}(c,n)}{\pi_{u|f_p}(c,n)} = 0$$

10. The straightforward proof idea of deriving an upper bound for all regular pairings of each round (excluding the pairing with the dummy class) does not go through because the inequality $\sum_{i=1}^{c}(n-n_i)^p < (c-1)n^p$ does not hold in general.

Proof The total effort for the single round robin is $\sum_{i=1}^{c-1}\sum_{j=i+1}^{c} f_p(n_i+n_j)$. Hence the class penalty ratio is:

$$\frac{\pi_{r|f_p}(c,n)}{\pi_{u|f_p}(c,n)} = \frac{g_{r|f_p}(c,n)}{g_{u|f_p}(c,n)} = \frac{\sum_{i=1}^{c-1}\sum_{j=i+1}^{c} f_p(n_i+n_j)}{cf_p(n)} = \frac{1}{c}\sum_{i=1}^{c-1}\sum_{j=i+1}^{c}\frac{f_p(n_i+n_j)}{f_p(n)}$$

Consequently,

$$\lim_{p\to\infty}\frac{\pi_{r|f_p}(c,n)}{\pi_{u|f_p}(c,n)} = \lim_{p\to\infty}\frac{1}{c}\sum_{i=1}^{c-1}\sum_{j=i+1}^{c}\frac{\lambda(n_i+n_j)^p}{\lambda n^p} = \frac{1}{c}\sum_{i=1}^{c-1}\sum_{j=i+1}^{c}\lim_{p\to\infty}\left(\frac{n_i+n_j}{n}\right)^p = 0$$

The last equation holds because $c > 2$ and $0 < n_i + n_j < n \;\; (i,j = 1\ldots c;\; i \neq j)$, hence $\frac{n_i+n_j}{n} < 1$. For the same reasons $\left(\frac{n_i+n_j}{n}\right)^p > \left(\frac{n_i+n_j}{n}\right)^{p+\epsilon}$ for all $\epsilon > 0$ and $\frac{\pi_{r|f_p}(c,n)}{\pi_{u|f_p}(c,n)}$ is monotonically decreasing with p. ■

As the decrease with increasing order p is strictly monotonic, this theorem implies that the more expensive a learning algorithm is, the larger will be the efficiency gain for using round robin binarization instead of unordered binarization. In particular, it may be the case that for expensive algorithms, even a *double* round robin is faster than unordered binarization (in fact, it is easy to see that Theorem 12 also holds for double round robin binarization) or may be even faster than ordered binarization. Assume, e.g., an algorithm with a quadratic complexity on a class-balanced eight-class problem (i.e., $p = 2$ and $c = 8$). According to Theorem 11 and Lemma 7:

$$\frac{\pi_{r|f_p}(c)}{\pi_{u|f_p}(c)} = \frac{(c-1)(\frac{2}{c})^{p-1}}{c} = \frac{7(\frac{1}{4})^1}{8} = \frac{7}{32} < \frac{1}{4}$$

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 as the single round robin, and assuming that the ordered approach is twice as fast as the unordered approach (see the following section for empirical values on that), the double round robin may in this case be faster than the ordered approach. This scenario will be empirically evaluated in the following section.

5.2 Empirical Evaluation

Contrary to the theoretical analysis in the previous section, where we focussed on the "friendly" case of pairing unordered binarization vs. single round robin, our empirical results compare the performance of a double round robin binarization with Ripper as a base learner against both ordered binarization (Ripper's default mode) and unordered binarization (Ripper with the parameters `-a unordered`). In the case of a linear algorithm complexity, the double round robin should be about two times slower than the unordered binarization and four times slower than the ordered binarization. As discussed in the previous

Table 3: *Runtime results:* The first column shows the average run-times (in CPU secs. user time) of R^3. The subsequent columns show the ratio of R^3 over unordered and ordered Ripper. The reported run-times are average training time over all folds of a 10-fold cross-validation. The line at the bottom shows the geometric average of the 17 cross-validated performance ratios.

dataset	R^3	vs. unordered	vs. ordered
abalone	140.28	*3.14*	*3.27*
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*
sat	186.89	*0.69*	*1.25*
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*
vowel	16.22	*0.87*	*2.16*
yeast	16.90	*1.77*	*3.12*
average		*2.02*	*3.19*

section, Ripper's super-linear run-time[11] might improve the relative performance of round robin learning.

Table 3 shows the run times 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. The reported times are average training times,[12] i.e., the performance ratios can be interpreted as empirical estimates of the class penalty ratios $\frac{\pi_r}{\pi_u}$ and $\frac{\pi_r}{\pi_o}$. On average, R^3 is about two times slower than Ripper in unordered mode, and about three times slower than Ripper in default, ordered mode, while Ripper's ordered mode is about 1.5 times faster than unordered mode (as opposed to the factor 2 we were assuming in the theoretical analysis at the end of Section 5.1). This means that despite the inefficient implementation, the empirical values are fairly close to the estimates we made at the end of the previous section: for a linear algorithm, we expected the double round-robin procedure to be about twice as slow as the unordered approach and about four times as slow than the ordered approach.

11. The complexity of Ripper's initial rule learning and pruning phase is $O(n \log^2(n))$ (Fürnkranz and Widmer, 1994; Cohen, 1995). Thereafter Ripper performs two phases of optimization, which—according to the experimental evidence shown in (Cohen, 1995)—only add a constant factor to the overall complexity.
12. Note that the performance ratios for testing times would in general be considerably worse; in some cases, we observed increases in the factors of up to 50%. We will briefly discuss the problem of classification efficiency (and some proposals for solving it) in Section 7.

Apparently, the additional savings—based on the fact that simpler concepts are learned for the pairwise tasks and that Ripper's run-time is super-linear—make up for the losses due to the sub-optimal implementation as a wrapper. For more expensive base learning algorithms (like support vector machines), the analysis in the previous section lets us expect bigger savings.

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, while unordered and ordered binarization are native in Ripper's efficient implementation in `C`. Clearly, a tight integration of round robin binarization into Ripper's `C`-code would be more efficient.[13]

6. Round Robin Learning as an Ensemble Technique

Ensemble techniques have received considerable attention within the recent machine learning literature (Dietterich, 1997; 2000a; Opitz and Maclin, 1999; Bauer and Kohavi, 1999). The idea to obtain a diverse set of classifiers for a single learning problem and to vote or average their predictions is both simple and powerful, and the obtained accuracy gains often have a sound theoretical foundation (Freund and Schapire, 1997; Breiman, 1996). Averaging the predictions of these classifiers helps to reduce the variance and often increase the reliability of the predictions. There are several techniques for obtaining a diverse set of classifiers. The most common technique is to use subsampling to diversify the training sets as in bagging (Breiman, 1996) and boosting (Freund and Schapire, 1997). Other techniques include the use of different feature subsets (Bay, 1999), to exploit the randomness of the base algorithms (Kolen and Pollack, 1991), possibly by artificially randomizing their behavior (Dietterich, 2000b), or to use multiple representations of the domain objects, for example by using information originating from different hyperlinks pointing to a web page (Fürnkranz, 1999a; 2002). Finally, classifier diversity can be ensured by modifying the output labels, i.e., by transforming the learning tasks into a collection of related learning tasks that use the same input examples but a different assignments of the class labels. Error-correcting output codes are the most prominent example for this type of ensemble method (Dietterich and Bakiri, 1995).

Clearly, round robin classification may also be interpreted as a member of this last group, and its performance gain may be seen in this context. Obviously, the final prediction is made by exploiting the redundancy provided by multiple models, each of them being constructed from a subset of the original data. However, contrary to subsampling approaches like bagging and boosting, these datasets are constructed deterministically.[14] In this respect pairwise classification shares more similarities with error-correcting output codes, but differs from it through the fixed procedure for setting up the new binary problems and the fact

13. The effect is somewhat compensated by the fact that we only report user time (which ignores time for disk access and system time). 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.
14. Boosting is also deterministic if the base learner is able to use weighted examples. Often, however, the example weights are interpreted as probabilities which are used for drawing a random sample as the training set for the next boosting iteration.

Table 4: *Boosting:* A comparison between round robin binarization and boosting, both with C5.0 as a base learner. The first column shows the results of C5.0, while the next three column pairs show the results of round robin learning, boosting, and the combination of both, all with C5.0 as a base learner. For these, we give both the absolute error rate and the performance ratio relative to the base learner C5.0. The last line shows the geometric average of these ratios.

dataset	C5.0	round robin		boosting		both	
abalone	78.48	75.08	*0.957*	77.88	*0.992*	74.67	*0.951*
car	7.58	5.84	*0.771*	3.82	*0.504*	1.85	*0.244*
glass	35.05	24.77	*0.707*	27.57	*0.787*	22.90	*0.653*
image	3.20	2.90	*0.905*	1.60	*0.500*	1.73	*0.541*
lr spectrometer	51.22	51.79	*1.011*	46.70	*0.912*	51.98	*1.015*
optical	9.20	5.04	*0.547*	2.46	*0.267*	2.54	*0.277*
page-blocks	3.09	2.98	*0.964*	2.58	*0.834*	2.78	*0.899*
sat	13.82	13.16	*0.953*	9.32	*0.675*	9.00	*0.651*
solar flares (c)	15.77	15.69	*0.995*	16.41	*1.041*	16.70	*1.059*
solar flares (m)	4.90	4.90	*1.000*	5.90	*1.206*	5.83	*1.191*
soybean	9.66	6.73	*0.697*	6.59	*0.682*	6.44	*0.667*
thyroid (hyper)	1.11	1.14	*1.024*	1.03	*0.929*	1.33	*1.190*
thyroid (hypo)	0.58	0.69	*1.182*	0.32	*0.545*	0.53	*0.909*
thyroid (repl.)	0.72	0.74	*1.037*	0.90	*1.259*	0.90	*1.259*
vehicle	26.24	29.20	*1.113*	24.11	*0.919*	23.17	*0.883*
vowel	21.72	19.49	*0.898*	8.89	*0.409*	14.75	*0.679*
yeast	43.26	40.63	*0.939*	41.85	*0.967*	40.77	*0.942*
average			*0.909*		*0.735*		*0.757*

that each of the new problems is smaller than the original problem. In particular the latter fact may often cause the sub-problems in pairwise classification to be conceptually simpler than the original problem (as illustrated in Figure 1).

In previous work (Fürnkranz, 2001), we observed that the improvements in accuracy obtained by R^3 over Ripper were quite similar to those obtained by C5.0-boost over C5.0 on the same problems. Round robin binarization seemed to work whenever boosting worked, and vice versa. The correlation coefficient r^2 of the the error ratios of C5.0-boost/C5.0 and R^3/Ripper on the 20 datasets was about 0.618, which is in the same range as correlation coefficients for bagging and boosting (Opitz and Maclin, 1999). We interpreted this as weak evidence that the performance gains of round robin learning may be comparable to that of other ensemble methods and that it could be used as a general method for improving a learner's performance on multi-class problems. We will further investigate this question in this section and will in particular focus upon a comparison of round robin learning with boosting (Section 6.1) and bagging (Section 6.2), and upon the potential of combining it with these techniques.

6.1 Boosting

As a first step, we performed a direct comparison of the performance of C5.0 and C5.0-boost (C5.0 called with the parameter -b, i.e., 10 iterations of boosting) to C5.0-rr, a single round robin procedure with C5.0 as the base learning algorithm. Table 4 shows the results of a 10-fold cross-validation on 17 datasets.

The first thing to note is that the performance of C5.0 does indeed improve by about 10% on average if round robin binarization is used as a pre-processing step for multi-class problems. This is despite the fact that C5.0 can handle multi-class problems and does not depend on a class binarization routine. However, the gain is not as consistent and not as big as the gain for Ripper (Table 2), possibly because Ripper's average error on the multi-class problems in our study is in general above that of C5.0 (by a factor of 1.122), and therefore allows for larger improvements. A possible explanation for this is that the unordered and ordered binarization schemes used by Ripper are not very good. This is confirmed by the fact that in a direct comparison (which can easily be computed from Tables 2 and 4), R^3 decreases the average error of C5.0 by a factor of 0.838, and the error of C5.0-rr by a factor of 0.923. From this, we can conclude that robin binarization helps Ripper to outperform C5.0 on multi-class problems.

The direct comparison between round robin classification and boosting shows that the improvement of C5.0-rr over C5.0 is not as large as the improvement provided by boosting: although there are a few cases where round robin outperforms boosting, C5.0-boost is much more reliable than C5.0-rr, producing an average error reduction of more than 26% on these 17 datasets. The correlation between the error reduction rates of C5.0-boost and C5.0-rr is very weak ($r^2 = 0.276$), which refutes our earlier hypothesis and brings up the question whether there is a fruitful combination of boosting and round robin classification. The last column of Table 4 answers this question negatively: the results of using round robin classification with C5.0-boost as a base learner does—on average—not lead to performance improvements over boosting.

These results are analogous to the results of Schapire (1997) who compared AdaBoost.OC (error-correcting output codes as a binarization scheme for conventional two-class AdaBoost) with AdaBoost.M1 (Freund and Schapire, 1997), AdaBoost's straightforward adaptation for multi-class base learners (a version of which is presumably also implemented in C5.0; Quinlan 1996), and found no significant differences for the base learner C4.5 (Quinlan, 1993), C5.0's predecessor. Similar to our comparison between C5.0-boost and round robin binarization, Schapire (1997) also found that boosting outperformed binarization via error-correcting output codes. In subsequent work, Allwein et al. (2000) showed that the performance gain of pairwise classification using AdaBoost as a base learner is on average indiscernible from the performance gain of alternative binarization schemes, including some employing error-correcting output codes (such as AdaBoost.OC).

6.2 Bagging

So far we were only concerned with single and double round robins. A natural extension to this procedure is to consider cases where more than two classifiers are trained for each binary problem. For algorithms with random components (such as Ripper's internal split of the training examples or the random initialization of back-propagation neural networks)

Table 5: *Bagging:* A comparison of round robin learning versus bagging and of the combination of both using Ripper as the base classifier. At the bottom, the average error ratios of the ensemble techniques over the respective base learner are shown for the base learners Ripper, C5.0, and C5.0-boost (we omitted the detailed results for the latter two). Note that the average performance ratios are relative to the base learner (i.e., they are only comparable within a line not between lines).

Ripper	base	round robin		bagging		both	
abalone	81.18	74.34	*0.916*	78.36	*0.965*	73.14	*0.901*
car	12.15	2.26	*0.186*	9.38	*0.771*	1.79	*0.148*
glass	34.58	25.70	*0.743*	29.44	*0.851*	25.70	*0.743*
image	4.29	3.46	*0.808*	2.51	*0.586*	2.99	*0.697*
lr spectrometer	61.39	53.11	*0.865*	57.82	*0.942*	52.92	*0.862*
optical	9.48	3.74	*0.394*	2.86	*0.302*	2.81	*0.296*
page-blocks	3.38	2.76	*0.816*	2.65	*0.784*	2.54	*0.751*
sat	13.04	10.35	*0.794*	10.18	*0.781*	8.92	*0.684*
solar flares (c)	15.91	15.77	*0.991*	15.91	*1.000*	15.69	*0.986*
solar flares (m)	5.47	5.04	*0.921*	5.26	*0.961*	5.18	*0.947*
soybean	8.79	6.30	*0.717*	8.20	*0.933*	6.00	*0.683*
thyroid (hyper)	1.49	1.11	*0.749*	1.41	*0.945*	1.09	*0.731*
thyroid (hypo)	0.56	0.53	*0.955*	0.58	*1.050*	0.42	*0.764*
thyroid (repl.)	0.98	1.01	*1.026*	0.98	*0.999*	0.85	*0.864*
vehicle	30.38	29.08	*0.957*	26.12	*0.860*	26.83	*0.883*
vowel	27.07	18.69	*0.690*	16.26	*0.601*	18.79	*0.694*
yeast	42.39	41.78	*0.986*	40.63	*0.959*	39.89	*0.941*
average (Ripper)			*0.747*		*0.811*		*0.685*
average (C5.0)			*0.909*		*0.864*		*0.838*
average (C5.0-boost)			*1.029*		*0.977*		*1.019*

this could simply be performed by running the algorithm on the same dataset with different random seeds. For other algorithms there are two options: randomness could be injected into the algorithm's behavior (Dietterich, 2000b) or random subsets of the available data could be used for training the algorithm. The latter procedure is more or less equivalent to bagging (Breiman, 1996). We will evaluate this option in this section.

Table 5 shows the results of a comparison of round robin learning, bagging, and a combination of both. Bagging was implemented by drawing 10 samples with replacement from the available data. Ties were broken in the same way as for the round robin binarization, i.e., by simple voting using the *a priori* class probability as a tie breaker. Similarly, bagging was integrated with round robin binarization by drawing 10 independent samples of each pairwise classification problem. Thus we obtained a total of $10c(c-1)$ predictions for each c-class problem, which again were simply voted. The number of 10 iterations was chosen arbitrarily (to conform to C5.0-boost's default number of iterations) and is certainly not optimal (in both cases).

The results show clearly that the performance of the simple round robin (second column) can be improved considerably by integrating it with bagging (last column). The bagged round robin procedure reduces Ripper's error on the datasets to about 68.5% of the original error (third line from the bottom). For comparison, we also show the results of bagging only, which seems to give the least improvement. The results of bagging are not included to compare it to the round robin, but to show that the reduction in error rate for the bagged round robin robin outperforms both its constituents.

We also repeated these experiments using C5.0 and C5.0-boost as the base learners. We only show the average performance for these cases. Again, the advantage of the use of round robin learning is less pronounced for the multi-class learner C5.0 (it is even below the improvement given by our simple bagging procedure), and the combination of C5.0-boost and round robin learning does not produce an additional gain. It is worth mentioning that the combination of boosting and bagging outperforms boosting, which confirms previous good results with such algorithms (Pfahringer, 2000; Krieger et al., 2001).

In order to compare the absolute performances of the algorithms we can normalize all relative results on the performance of one algorithm (say Ripper). C5.0's performance was better than Ripper's by a factor of about 0.891. Multiplying this with the improvement of 0.735 of boosting (Table 4) and of an additional 0.977 for adding bagging (Table 5) yields that bagged C5.0-boost has about 64% of the error rate of basic Ripper, which makes it the best performing combination. In comparison, the combination of round robin and bagging for Ripper (68.5%) is relatively close behind, in particular if we consider the bad performance of Ripper in comparison to C5.0. An evaluation of a boosting variant of Ripper (such as Slipper; Cohen and Singer, 1999) would be of interest.

7. Other Properties and Open Questions

In the following, we briefly discuss further important aspects of round robin binarization.

Decoding: We have mostly ignored the issue of *decoding* techniques for combining the predictions of the pairwise classifiers into a final prediction for the multi-class problem. The technique we used in this paper (simple voting using the *a priori* probability of the class as a tie breaker), is quite likely to be suboptimal. First, one could improve the tie breaking by 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). A further step ahead would be to weight each vote with a confidence estimate provided by the base classifier, or to allow a classifier only to vote for a class if it has a certain minimum confidence in its prediction. Several studies in various contexts have compared different voting techniques for combining the predictions of the individual classifiers of an ensemble (e.g., Mayoraz and Moreira, 1997; Allwein et al., 2000; Fürnkranz, 2002). Although the final word on this issue remains to be spoken, it seems to be the case that techniques that include confidence estimates into the computation of the final predictions are preferable (cf. also Schapire and Singer, 1999). Along similar lines, there have been several proposals for combining the class probability estimates of the pairwise classifiers into class probability distributions for the multi-class problems (Price et al., 1995; Hastie and Tibshirani, 1998). More elaborate proposals suggest learning separate classifiers for deciding whether a given example belongs to one of the two classes used to train a certain member of the pairwise ensemble (Moreira and Mayoraz, 1998), or organizing the classifiers into an efficient graph structure that can derive a prediction in at most $c - 1$ steps (Platt et al., 2000).

Classification Efficiency: Our efficiency analysis is only concerned with training time. At testing time, one has to test a quadratic number of classifiers in order to make the final prediction. Even though the constituent classifiers are quite likely to be simpler (which often means that they can make faster predictions) it can be expected that classification takes considerably longer for a round robin ensemble than for unordered binarization. This situation is particularly bad for lazy learners, which defer most of their training effort to the classification phase.

Next to the straightforward solution of testing all theories in parallel (see below), a solution for this problem could be found in the proposal of Platt et al. (2000) who suggest organizing pairwise classifiers in a decision graph where each node represents a binary classifier. They show that this structure allows obtainment of a prediction for a c-class problem by consulting only $c - 1$ pairwise classifiers without loss of accuracy on three benchmarks problems. In some sense, this technique may be viewed as the "ordered" version of round robin classification.

Finally, we could once more take a look at sports and game tournaments, where elaborate pairing schemes allow determination of a reliable ranking in cases where a round robin tournament is infeasible due to the high number of players. The frequently used knock-out tournament (where players are paired randomly and only the winner advances into the next round) is probably too brittle. An interesting alternative might be provided by swiss system tournaments, which are frequently used in competitive chess. In this type of tournament, all players play a fixed number of rounds, typically of the order $\log(c)$. The trick is that in each round players of about equal strength (i.e., players that are ranked close to each other in the current standings of the tournament) are paired against each other. Such schemes could improve classification time for problems with very high numbers of classes, in particular if classification is very expensive (e.g., in the case of lazy learners).

Comprehensibility: While boosting seems to provide larger gains in accuracy, the price to pay is that the learned ensemble of classifiers is no longer easy to comprehend. 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, p.16) 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 one 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.[15]

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 easily parallelized by assigning the binary classification problems to different processors, as already noted by Friedman (1996) and Lu and Ito (1999). As each binary task will be smaller than the original task, the total training time of a multi-class

15. The aspect of being able to rank the available classifications 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.

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. Naturally, a parallel implementation would also provide a trivial solution to the problem with classification efficiency discussed above.

Memory Requirements: It is also clear that each individual binary task in a round-robin binarization has fewer 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.

Imbalanced Class Distributions: It would also be interesting to investigate the effect of round robin binarization on minority classes, in particular in problems where several large classes appear next to a few small classes. We think that 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. On the other hand, if the base learner tends to prefer large classes, one dominant large class will tend to win against all minority classes and will be more frequently predicted. The evidence from Table 4 seems to confirm this: it is primarily sets with skewed class distributions where round robin classification does not perform well (consider, e.g., the three thyroid datasets). However, this evidence is certainly not conclusive and we believe that a closer investigation of this issue is a rewarding topic for future research.

8. Related Work

The idea of pairwise classification has been known in the neural networks and statistics communities for some time. For example, Witten and Frank (2000, p.113) refer to it as a technique for making linear regression applicable to multi-class problems but do not cite a source. The earliest reference we could find is Knerr et al. (1990) who propose a stepwise procedure for linearizing non-linear multi-class problems by first trying to identify classes that can be solved by a one-against-all approach, then a pairwise approach, and finally a piece-wise linear technique. The motivation behind this and other works in the neural networks community is that it is often better to have a modular network, i.e., a network that consists of several simpler and independently trained sub-networks, rather than a single, complex multi-layer neural network, which usually requires a large hidden layer and significant training times. Unordered (Anand et al., 1995) and pairwise (Lu and Ito, 1999) binarization techniques have also been investigated in this context.

Friedman (1996) evaluates pairwise classification on two versions of CART (Breiman et al., 1984) and a nearest neighbor algorithm on 50 randomly generated problems. He observed improvements for the CART version which uses a linear function for splitting a node and for the nearest neighbor rule. For CART with axis parallel splits, the performance of the pairwise class binarization was similar to that of the standard techniques. The author also provided a brief discussion of the complexity of the approach.

Naturally, pairwise classification was also investigated within the support vector machine community. A comparison of unordered and round robin binarization on a speaker identification problem can be found in (Schmidt, 1996; Schmidt and Gish, 1996), without providing a clear conclusion about the preferable approach. Kreßel (1999) discusses the technique in some detail and presents empirical results on a digit recognition task, where the author notes the unexpected efficiency of the approach without providing an explanation for it. Recently, Hsu and Lin (2002) conducted an excellent empirical comparison between unordered binarization, pairwise classification with both the simple voting technique we used and with the more efficient proposal for organizing the classifiers into an efficient DAG (Platt et al., 2000), as well as two approaches for directly generalizing the support vector algorithm to multi-class problems. In their experiments, round robin binarization performed best, both in terms of accuracy and training time. Interestingly, the advantage over competing methods was more pronounced for a linear kernel than for a non-linear RBF kernel. We interpret this as evidence that round robin binarization simplifies the individual binary decision problems as motivated in Figure 1.

Angulo and Català (2000) suggest a related technique where multi-class problems are mapped to 3-class problems. Like with pairwise classification, the idea is to generate one training set for each pair of classes, but in addition to encoding the two class values with target values $+1$ and -1, examples of all other classes are added with a target value of 0, which gives up some of the advantages that result from the reduction of the training set sizes on the binary problems. They do not evaluate their approach. A similar idea was used by Kalousis and Theoharis (1999) for the meta-learning task of predicting the most suitable learning algorithm(s) for a given dataset. A nearest-neighbor learner was trained to predict the better algorithm for each pair of learning algorithms, where each of these pairwise problems had three classes: one for each algorithm and a third class "tie" if both algorithms performed indistinguishably.

Error-correcting output codes (Dietterich and Bakiri, 1995) are a popular and powerful class binarization technique. The basic idea is to encode a c-class problem as $\bar{c}$ binary problems ($\bar{c} > c$), where each binary problem uses a *subset* of the classes as the positive class and the remaining classes as a negative class. It may thus be viewed as a generalization of unordered binarization, where only single classes are used as positive examples. As a consequence, each original class is encoded as a $\bar{c}$-dimensional binary vector, one dimension for each prediction of a binary problem (by convention $+1$ for positive and -1 for negative). The resulting matrix of the form $\{-1, +1\}^{c \times \bar{c}}$ is called the *coding matrix*. New examples are classified by determining the row in the matrix that is closest to the binary vector obtained by submitting the example to the $\bar{c}$ classifiers. If the binary problems are chosen in a way that maximizes the distance between the class vectors, the reliability of the classification can be significantly increased. Error-correcting output codes can also be easily parallelized, but each subtask requires the total training set. As typically $\bar{c} > c$, the penalty function $\pi_{ecoc} > c$, i.e., pairwise and unordered binarization are more efficient.

Allwein et al. (2000) show that a generalization of error-correcting output codes can be used as a general framework for class binarization techniques. Their basic idea is to generalize the coding matrices in a way that allows examples to be ignored in the binary problems, i.e., to allow the coding matrices to be of the form $\{-1, 0, +1\}^{c \times \bar{c}}$. Clearly, pairwise classification is a special case in this framework. For example, the coding matrix for a double round robin has $\bar{c} = c(c-1)$ columns, where the column corresponding to the binary problem $<i, j>$ has $+1$ in the i-th row, -1 in the j-th row and 0 in all other rows. Thus, each row is a vector of the form $\{-1, 0, +1\}^{\bar{c}}$. Note, however, that the vector of predictions is of the form $\{-1, +1\}^{\bar{c}}$ because every binary classifier will make a prediction (either $+1$ or -1) for every example. Nevertheless, the simple voting procedure we used is equivalent to finding the row that is most similar to the prediction vector (if similarity is measured with the Hamming distance), which is equivalent to the decoding technique suggested by Dietterich and Bakiri (1995). Allwein et al. (2000) criticize this simple Hamming decoding and suggest the use of loss-based decoding techniques that take into account the confidence of the base learner into its own predictions. In a theoretical analysis, which relates the training error of decoding methods to the error on the binary problems and to the minimum distance between entries in the coding matrix, the authors derive upper bounds for the training error of loss-based decoding that are lower than those for Hamming decoding. In an experimental study with five different class binarization techniques and three decoding techniques (two of them loss-based and one Hamming decoding), the loss-based techniques seemed to produce lower generalization errors. Their results also showed that for support vector machines unordered binarization is inferior to all other techniques, among them pairwise classification. Among these alternatives, no clear winner emerged. However, for boosted decision trees, unordered binarization performed on the same level as all other approaches.

Finally, we note the relation of round robin classification to comparison training (Tesauro, 1989; Utgoff and Clouse, 1991), which has been proposed as a training procedure in evaluation function learning. In this framework, the learner is not trained with the target values of the evaluation function in certain states, but instead is trained on state pairs where the preferable state is marked. Thus, this training procedure is somewhere between supervised learning (where the function is trained on examples of target values) and reinforcement learning (where it only receives indirect feedback about the value of states). Tesauro (1989) demonstrated a particularly interesting technique, where he enforced a symmetric neural network architecture and showed that with this architecture, one only has to perform n network evaluations to determine the best of n moves. It is an interesting open question, whether a similar technique could be employed for speeding up pairwise classification.

9. Conclusions

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. In particular, round robin binarization helps Ripper to outperform C5.0 on multi-class problems, whereas C5.0 outperforms the original version of Ripper on the same problems. 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).

Moreover, we demonstrated both empirically and theoretically that the resulting quadratic growth in the number of learning problems is compensated by the reduction in size for each of the individual problems. For base algorithms with linear or super-linear run-time, round robin binarization is faster than the conventional unordered technique. Furthermore, we have proven that this advantage increases with the complexity of the base algorithm. Note that these theoretical results are not restricted to rule learning, but are also applicable to perceptrons, support vector machines, boosting, and other learning algorithms that need binarization techniques for solving multi-class problems. Our experimental results confirm the efficiency of round robin binarization for rule learning, but for a true evaluation of the performance of this technique, an efficient, tight integration into a separate-and-conquer rule learning algorithm would be needed.

Finally, we investigated the properties of round robin learning as a general ensemble technique, in particular in comparison to bagging and boosting. For C5.0 and even less so for C5.0-boost, round robin classification did not seem to work as well as it did for Ripper, which is consistent with previous similar results on error-correcting output codes. Similarly, a straightforward integration of pairwise classification with bagging outperformed both constituent techniques, when using Ripper (and to some extent when using C5.0) as a base learner, but not in combination with boosting. It remains to be seen whether round robin learning does not work well for boosting in general, or whether this is a specific phenomenon for C5.0 and its boosting procedure. In any case, we can conclude that round robin classification provides a more efficient and more accurate alternative to the class binarization procedures that are currently in use in inductive rule learning.

Acknowledgments

I wish to thank William Cohen, Eibe Frank, Alexandros Kalousis, Stefan Kramer, Rich Maclin, Dragos Margineantu, Johann Petrak, Foster Provost, Lupčo Todorovski, Gerhard Widmer, and the maintainers of and contributors to the UCI collection of databases for discussions, programs, databases, and/or pointers to relevant literature. Thanks are also due to the anonymous reviewers and the editor, in particular for the careful comments on the theoretical section of this paper.

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. The author is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF.

References

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

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

C. Angulo and A. Català. *K*-SVCR. A multi-class support vector machine. In R. López de Mántaras and E. Plaza (eds.) *Proceedings of the 11th European Conference on Machine Learning (ECML-2000)*, pp. 31–38. Springer-Verlag, 2000.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–169, 1999.

S. D. Bay. Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, 3(3):191–209, 1999.

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

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA, 1984.

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

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

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

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

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T. G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4): 97–136, Winter 1997.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli (eds.) *First International Workshop on Multiple Classifier Systems*, pp. 1–15. Springer-Verlag, 2000a.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000b.

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

A. Feelders and W. Verkooijen. Which method learns most from the data? Methodological issues in the analysis of comparative studies. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, pp. 219–225, Fort Lauderdale, Florida, 1995.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

J. H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, Stanford, CA, 1996.

J. Fürnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–171, 1997.

J. Fürnkranz. Exploiting structural information for text classification on the WWW. In D. Hand, J. N. Kok, and M. Berthold (eds.) *Advances in Intelligent Data Analysis: Proceedings of the 3rd International Symposium (IDA-99)*, pp. 487–497, Amsterdam, Netherlands, 1999a. Springer-Verlag.

J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1): 3–54, 1999b.

J. Fürnkranz. Round robin rule learning. In C. E. Brodley and A. P. Danyluk (eds.) *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pp. 146–153, Williamstown, MA, 2001b. Morgan Kaufmann Publishers.

J. Fürnkranz. Hyperlink ensembles: A case study in hypertext classification. *Information Fusion*. Special Issue on Fusion of Multiple Classifiers, 2002. To appear.

J. Fürnkranz and G. Widmer. Incremental Reduced Error Pruning. In W. Cohen and H. Hirsh (eds.) *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pp. 70–77, New Brunswick, NJ, 1994. Morgan Kaufmann.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In M. I. Jordan, M. J. Kearns, and S. A. Solla (eds.) *Advances in Neural Information Processing Systems 10 (NIPS-97)*, pp. 507–513. MIT Press, 1998.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 2002. To appear.

A. Kalousis and T. Theoharis. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337, 1999.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In F. Fogelman Soulié and J. Hérault (eds.) *Neurocomputing: Algorithms, Architectures and Applications*, volume F68 of *NATO ASI Series*, pp. 41–50. Springer-Verlag, 1990.

S. Knerr, L. Personnaz, and G. Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3(6):962–968, 1992.

J. F. Kolen and J. B. Pollack. Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems 3 (NIPS-90)*, pp. 860–867. Morgan Kaufmann, 1991.

U. H.-G. Kreßel. Pairwise classification and support vector machines. In B. Schölkopf, C. Burges, and A. Smola (eds.) *Advances in Kernel Methods: Support Vector Learning*, chapter 15, pp. 255–268. MIT Press, Cambridge, MA, 1999.

A. Krieger, A. J. Wyner, and C. Long. Boosting noisy data. In C. E. Brodley and A. P. Danyluk (eds.) *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pp. 274–281. Williamstown, MA, 2001. Morgan Kaufmann Publishers.

B.-L. Lu and M. Ito. Task decomposition and module combination based on class relations: A modular neural network for pattern classification. *IEEE Transactions on Neural Networks*, 10(5):1244–1256, 1999.

E. Mayoraz and E. Alpaydin. Support vector machines for multi-class classification. In J. Mira and J. V. Sánchez-Andrés (eds.) *Engineering Applications of Bio-Inspired Artificial Neural Networks: Proceedings of the International Work-Conference on Artificial and Natural Neural Networks (IWANN-99), Volume II*, pp. 833–842, Alicante, Spain, 1999. Springer-Verlag.

E. Mayoraz and M. Moreira. On the decomposition of polychotomies into dichotomies. In D. H. Fisher (ed.) *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pp. 219–226, Nashville, TN, 1997. Morgan Kaufmann.

Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947.

M. Moreira and E. Mayoraz. Improved pairwise coupling classification with correcting classifiers. In C. Nédellec and C. Rouveirol (eds.) *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, pp. 160–171, Chemnitz, Germany, 1998. Springer-Verlag.

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

B. Pfahringer. Winning the KDD99 classification cup: Bagged boosting. *SIGKDD explorations*, 1(2):65–66, 2000.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Müller (eds.) *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pp. 547–553. MIT Press, 2000.

D. Price, S. Knerr, L. Personnaz, and G. Dreyfus. Pairwise neural network classifiers with probabilistic outputs. In G. Tesauro, D. Touretzky, and T. Leen (eds.) *Advances in Neural Information Processing Systems 7 (NIPS-94)*, pp. 1109–1116. MIT Press, 1995.

D. Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann, San Francisco, CA, 1999.

J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 725–730. AAAI/MIT Press, 1996.

R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

R. E. Schapire. Using output codes to boost multiclass learning problems. In D. H. Fisher (ed.) *Proceedings fo the 14th International Conference on Machine Learning (ICML-97)*, pp. 313–321, Nashville, TN, 1997. Morgan Kaufmann.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

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

M. S. Schmidt and H. Gish. Speaker identification via support vector classifiers. In *Proceedings of the 21st IEEE International Conference Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*, pp. 105–108, Atlanta, GA, 1996.

G. Tesauro. Connectionist learning of expert preferences by comparison training. In D. Touretzky (ed.) *Advances in Neural Information Processing Systems 1 (NIPS-88)*, pp. 99–106. Morgan Kaufmann, 1989.

P. E. Utgoff and J. Clouse. Two kinds of training information for evaluation function learning. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, pp. 596–600, Anaheim, CA, 1991. AAAI Press.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In M. Verleysen (ed.) *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN-99)*, pp. 219–224, Bruges, Belgium, 1999.

I. H. Witten and E. Frank. *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann Publishers, 2000.