Round Robin Ensembles*

Johannes Fürnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
juffi@oefai.at

OEFAI-TR-2002-37

Abstract

In this paper we investigate the performance of pairwise (or round robin) classification, originally a technique for turning multi-class problems into two-class problems, as a general ensemble technique. In particular, we show that the use of round robin ensembles will also increase the classification performance of decision tree learners, even though they can directly handle multi-class problems. The performance gain is not as large as for bagging and boosting, but on the other hand round robin ensembles have a clearly defined semantics. Furthermore, we investigate whether confidence estimates can be used to improve the accuracy of the predictions of the ensemble. Finally, we show that the advantage of pairwise classification over direct multi-class classification and one-against-all binarization increases with the number of classes, and that round robin ensembles form an interesting alternative for problems with ordered class values.

1 Introduction

In a recent paper (Fürnkranz, 2001), we analyzed the performance of pairwise classification (which we call *round robin* learning) for handling multi-class problems in rule learning. Most rule learning algorithms (and many other concept learning algorithms, which are restricted to learning binary classes) handle multi-class problems by converting them into a series of two-class problems, one for each class. Each of these problems uses the examples of the corresponding class as positive examples, and all others as negative examples. This procedure is known as *one-against-all* class binarization. Round robin binarization, on the other hand, converts a c-class problem into a series of two-class problems by learning one classifier for each pair of classes, using only training examples for these two classes and ignoring all others. A new example is classified by submitting it to each of the $c(c-1)/2$ binary classifiers, and combining their predictions via simple voting. The most important result of the previous study was that this procedure not only increases predictive accuracy, but that it is also no more expensive than the more commonly used one-against-all approach.

Obviously, a round robin classifier may also be interpreted as an ensemble classifier that, similar to error-correcting output codes (Dietterich and Bakiri, 1995), constructs an ensemble by transforming the learning problem into multiple two-class problems and learning a classifier for each of them. In fact, Allwein et al. (2000) show that pairwise classification (and other class binarization techniques)

*A draft of this paper has previously appeared as (Fürnkranz, 2002a). The main changes are that Sections 3 and 4 have been completely rewritten and complemented with a few new results, and that Section 5 is an original contribution in this paper.

are a special case of a generalized version of error-correcting output codes, which allows to specify that certain classes should be ignored for some problems (in addition to assigning them to a positive or a negative class, as conventional output codes do). In this paper, we will investigate several properties of round robin learning as a general ensemble technique. We will start with a brief recapitulation of our previous results on round robin learning (Section 2). In Section 3, we will investigate the performance of round-robin binarization as a general ensemble technique and show that it may also lead to performance improvements for multi-class base learning algorithms, in our case c5.0. The comparison of round robin ensembles to bagging and boosting (Section 4) will show that its performance gain may match the gain for bagging and boosting if ripper is used as base learner, but will lack behind boosting for c5.0. We will also evaluate various combinations of bagging, boosting and round robin. In Section 5, we discuss the use of confidence estimates for improving the prediction quality of a round robin ensemble. Finally, as more classes result in a larger ensemble of classifiers, it is reasonable to expect that the performance of round robin ensembles depends crucially on the number of classes of the problem. In Section 6, we will investigate this relation on classification problems with identical attributes but varying numbers of classes, which we obtain by discretizing the target variables of regression problems. Our results will show that round robin learning can indeed improve the performance of the c4.5 and c5.0 decision tree learners, and that a higher number of classes increases its performance, in particular in comparison to a one-against-all binarization. The same results will also show that round robin learning is a viable alternative for ordered classification (Section 7).

2 Round Robin Classification

In this section, we will briefly review round robin learning (aka pairwise classification) in the context of our previous work in rule learning (Fürnkranz, 2001; 2002b). Separate-and-conquer rule learning algorithms (Fürnkranz, 1999) are typically formulated in a concept learning framework. The goal is to find an explicit definition for an unknown concept, which is implicitly defined via a set of positive and negative examples. Within this framework, multi-class problems, i.e., problems in which the examples may belong to (exactly) one of several categories, are usually addressed by defining a separate concept learning problem for each class. Thus the original learning problem is transformed into a set of binary concept learning problems, one for each class, where the positive training examples are those belonging to the corresponding class and the negative training examples are those belonging to all other classes. Clark and Boswell (1991) proposed this technique for dealing with multi-class rule learning problems, but is also well-known in other areas such as neural networks (Anand et al., 1995), support vector machines (Cortes and Vapnik, 1995), or statistics (cf. multi-response linear regression). A variant of the technique, in which classes are first *ordered* (e.g., according to their relative frequencies in the training set) is used in the ripper rule learning algorithm (Cohen, 1995).

On the other hand, the basic idea of round robin classification is to transform a c-class problem into $c(c-1)/2$ binary problems, one for each pair of classes. Note that in this case, the binary decision problems not only contain fewer training examples (because all examples that do not belong to the pair of classes are ignored), but that the decision boundaries of each binary problem may also be considerably simpler than in the case of one-against-all binarization. In fact, in the example shown in Figure 1, each pair of classes can be separated with a linear decision boundary, while more complex functions are required to separate each class from all other classes. Evidence that the decision boundaries of the binary problems are in fact simpler can also be found in practical applications: 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. Similarly, Hsu

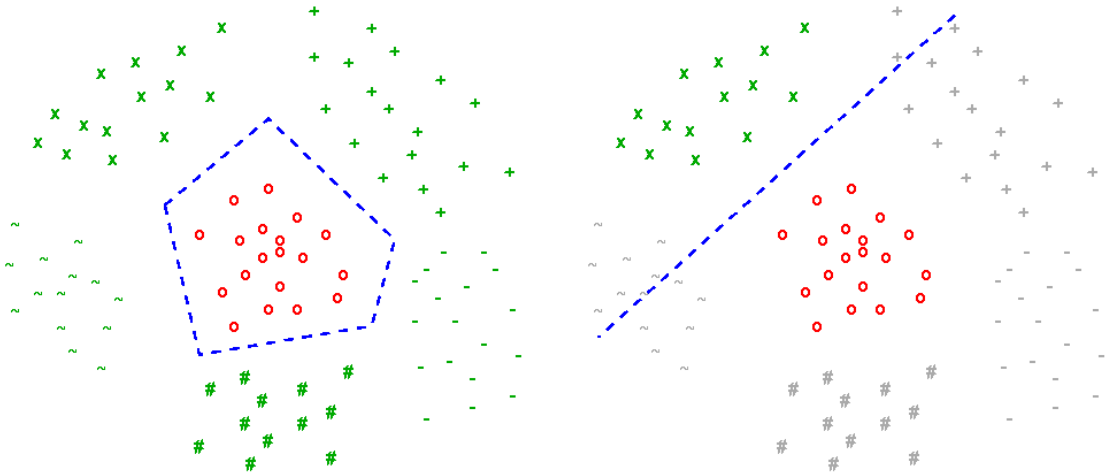

Figure 1: *One-against-all class binarization* (left) transforms each c-class problem into c binary problems, one for each class, where each of these problems uses the examples of its class as the positive examples (here o), and all other examples as negatives. *Round robin class binarization* (right) transforms each c-class problem into $c(c-1)/2$ binary problems, one for each pair of classes (here 0 and x) ignoring the examples of all other classes.

and Lin (2002) 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.

The basic idea of pairwise classification is fairly well-known from the literature. It can be found in the areas of statistics (Bradley and Terry, 1952; Friedman, 1996), neural networks (Knerr et al., 1990; 1992; Price et al., 1995; Lu and Ito, 1999), support vector machines (Schmidt and Gish, 1996; Hastie and Tibshirani, 1998; Kreßel, 1999; Hsu and Lin, 2002), and others. We refer to (Fürnkranz, 2002b, Section 8) for a brief survey of the literature on this topic.

The main contributions of our previous work (Fürnkranz, 2001; 2002b) were on the one hand to empirically evaluate the technique for rule learning algorithms and to show that it is preferable to the one-against-all technique that is used in most rule learning algorithms. More importantly, however, we analyzed the computational complexity of the approach, and demonstrated that despite the fact that its complexity is quadratic in the number of classes, the algorithm is no slower than the conventional one-against-all technique. It is easy to see this, if we consider that in the one-against-all case, each training example is used c times (namely in each of the c binary problems), while in the round robin approach, each examples is only used $c-1$ times, namely only in those binary problems, where its own class is paired against one of the other $c-1$ classes. Furthermore, the advantage of pairwise classification increases for computationally expensive (super-linear) learning algorithms. The reason is that expensive learning algorithms learn many small problems much faster than a few large problems. For details we refer to (Fürnkranz, 2002b).

3 Round Robin Ensembles

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 multiple classifiers reduces the variance and often increases the reliability of the classifier. There are several techniques for obtaining a diverse set of classifiers. The most common technique is the use of subsampling to diversify the training sets as in bagging (Breiman, 1996) and boosting (Freund and Schapire, 1997). Other techniques include exploiting the randomness of the base algorithms (Kolen and Pollack, 1991), possibly by artificially randomizing their behavior (Dietterich, 2000b), or the use of different feature subsets (Bay, 1999) or multiple representations of the domain objects, for example by using information originating from different hyperlinks pointing to a web page (Fürnkranz, In press). 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 (Dietterich and Bakiri, 1995) are the most prominent example for this type of ensemble method.

In this section we suggest that round robin classification may also be interpreted as an ensemble technique, and its performance gain may be viewed in this context. Like with conventional ensemble techniques, 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.[1] In this respect pairwise classification is quite similar to error-correcting output codes, but differs from them through its fixed procedure for setting up the new binary problems, and the fact that each of the new problems is smaller than the original problem.

The first and foremost question we are interested in is whether round robin learning may also boost the performance for learning algorithms that are in principle capable of dealing with multi-class problems. To that end, we performed a direct comparison of the performance of c5.0 to c5.0-rr, a round robin procedure with c5.0 as the base learning algorithm. c5.0-rr transforms each c-class problem into $c(c-1)/2$ binary problems and uses c5.0 to learn a decision tree for each of them. For predicting its class, a test example is submitted to all $c(c-1)/2$ classifiers and their predictions are combined via unweighted voting. Ties are broken in favor of larger classes.

The left half of Table 1 shows the results on 18 datasets with 4 or more classes from the UCI repository (Blake and Merz, 1998). For all datasets we estimated error rates with a 10-fold stratified cross-validation, except for *letter*, where we used the standard 4000 examples hold-out set.[2] Because of this difference, we did not include the *letter* dataset into the computation of averages. Shown are both the error rates of c5.0 and its round robin version, as well as their ratio, which tells us the relative improvement (a performance ratio of 0.9 means that the round robin version reduces the error rate of the base learner by 10%). For comparison, the right half of the table shows the same results for ripper.

[1] Boosting is also deterministic if the base learner is able to directly use weighted examples. Often, however, the example weights are interpreted as probabilities which are used for drawing the sample for the next boosting iteration.

[2] Some other domains also have dedicated test sets (*abalone*, *sat*, *vowel*), but as these are considerably smaller than *letter*, we decided to pool the training and test examples and evaluated them via 10-fold cross-validation as well. Besides, *letter* is a frequently used benchmark for class binarization techniques. The use of the dedicated hold-out set makes our results in this domain comparable to previous results.

Table 1: A comparison between c5.0 and a round robin ensemble using c5.0 as the base learner (left half of the table), and between ripper and its round robin version (right half). Shown are the estimated error rates, and the ratio of the performance of the round robin version over the base learner. The last line shows the mean error rates and the geometric average of the performance ratios. Results for *letter* were performed in a different setup and are not included into the averages.

dataset	c5.0	round robin		ripper	round robin	
letter	12.48	8.80	*0.705*	15.75	7.85	*0.498*
abalone	78.48	75.08	*0.957*	81.18	74.34	*0.916*
car	7.58	5.84	*0.771*	12.15	2.26	*0.186*
glass	35.05	24.77	*0.707*	34.58	25.70	*0.743*
image	3.20	2.90	*0.905*	4.29	3.46	*0.808*
lr spectrometer	51.22	51.79	*1.011*	61.39	53.11	*0.865*
optical	9.20	5.04	*0.547*	9.48	3.74	*0.394*
page-blocks	3.09	2.98	*0.964*	3.38	2.76	*0.816*
sat	13.82	13.16	*0.953*	13.04	10.35	*0.794*
solar flares (c)	15.77	15.69	*0.995*	15.91	15.77	*0.991*
solar flares (m)	4.90	4.90	*1.000*	5.47	5.04	*0.921*
soybean	9.66	6.73	*0.697*	8.79	6.30	*0.717*
thyroid (hyper)	1.11	1.14	*1.024*	1.49	1.11	*0.749*
thyroid (hypo)	0.58	0.69	*1.182*	0.56	0.53	*0.955*
thyroid (repl.)	0.72	0.74	*1.037*	0.98	1.01	*1.026*
vehicle	26.24	29.20	*1.113*	30.38	29.08	*0.957*
vowel	21.72	19.49	*0.898*	27.07	18.69	*0.690*
yeast	43.26	40.63	*0.939*	42.39	41.78	*0.986*
average	19.15	17.69	*0.909*	20.74	17.36	*0.747*

The last line shows the arithmetic mean of the error rates, and the geometric mean of the performance ratios (so that x and $1/x$ average to 1). Note that both summary measures have to be interpreted with care: the average error rate is dominated by results with large variations among the algorithms (particularly so for the run-time results discussed below), while the performance ratios are somewhat influenced by the fact that the default accuracy of the problems decreases with an increasing number of classes. Consequently, error differences for problems with more classes receive a lower weight (assuming there is some correlation of the performance of the algorithms and the default accuracy of the problem).

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. This is despite the fact that c5.0 can directly handle multi-class problems by itself, and does not depend on a class binarization routine. Presumably the simpler decision boundaries of the binary problems allow c5.0 to learn the class-binarized problems fairly well, so that the combination of their binary predictions produces more reliable multi-class predictions.

However, the relative improvement over c5.0 is not as large as the relative improvement over ripper: the improvement ratios are in general somewhat lower (although there are some cases with large gains such as *optical* and *soybean*), and the average reduction of the error rate is only about 10% (as opposed to $\approx 25\%$ for ripper). In absolute terms, on the other hand, the performance of the round robin versions of ripper and c5.0 are very similar: the error rates are always in approximately the same range, and their averages (which are dominated by the larger error rates as discussed above) are approximately the same. Apparently, both ensembles reach the same peak performance although the

performance of their respective base learners is very different (ripper's average performance on the multi-class problems in our study is in general considerably below that of c5.0, for which we blame the inadequacy of ripper's class binarization scheme.[3]).

In any case, these results indicate that round robin ensembles may lead to considerable reductions in error rate. In the next section, we try to quantify the size of this improvement in comparison to bagging and boosting.

4 Comparison to Bagging and Boosting

In this section, we will compare the performance of round robin ensembles to bagging and boosting (Section 4.1), investigate the potential of combining these techniques with each other (Section 4.2), and compare several other properties of round robin ensembles to those of bagging and boosting (Section 4.3).

4.1 Experimental Comparison

For doing a comparison of round robin, bagging and boosting, we implemented a simple bagging algorithm that draws 10 samples with replacement from the available data, trains a base learning algorithm on each of them, and combines the predictions in the same way as for round robin binarization, i.e., by simple voting using the *a priori* class probability as a tie breaker. For boosting, we relied on the built-in strategies of the algorithms: c5.0 has an option `-b` for boosting, and ripper has a separate utility `boost` that performs boosting. By default, both algorithms limit the number of iterations with 10, and we did not change this setting. Also, our choice of 10 iterations for bagging was arbitrarily based on these default values for boosting, and is certainly not optimal.

Table 2 shows the results of the comparison of round robin learning, bagging, and boosting using c5.0 (top half) and ripper (bottom half) as base learners. In the case of c5.0, the round robin ensembles do not reach the performance of boosted c5.0. Although there are a few cases where round robin outperforms boosting, the performance gains obtained by boosting are in general larger than those of round robin learning. The average error reduction of boosting is more than 26% on these 17 datasets. The situation is somewhat different when ripper is used as a base learner. ripper's built-in boosting procedure does not quite reach the same performance level as round robin learning, although it is able to outperform it significantly in some cases (e.g., *vowel*). In general, ripper's boosting method does not seem to be as robust as c5.0's (which is also witnessed by its crash on the *abalone* dataset). We do not know whether this is an artifact of the implementation, or whether this is due to some fundamental differences between rule and decision tree learning algorithms. In any case, round robin ripper also outperforms bagged ripper, while bagged c5.0 outperforms the round robin version of c5.0, which could be interpreted as evidence that the quality of the base learner (c5.0 performs much better than ripper on our set of multi-class problems) has a strong impact of the performance of bagging and boosting. Again, we interpret this as evidence that a large part of the gain that round robin learning produces for ripper is due to the inferiority of the one-against-all class binarization technique.

It is worth to have a look at the training times of each algorithm, which are shown in the right half of Table 2. For this comparison two things should be kept in mind: first, it should be noted that boosting had a bit of an advantage because the boosting strategies are directly implemented in the algorithms' C-code, whereas round robin and bagging were implemented in Perl in the form of

[3]These results used ripper in default mode, i.e., with an ordered binarization. The results using unordered, one-against-all binarization are approximately the same (Fürnkranz, 2002b).

Table 2: A comparison of round robin learning, bagging and boosting using c5.0 (top half) and ripper (bottom half) as the base learner. Shown are error rates and performance ratios over the base learner (both in the left half of the table) as well as the average training time (CPU secs. user time) of each algorithm. For bagging and boosting we also show the ratio of their run-time over round robin's run-time. ripper's implementation of boosting crashed on the *abalone* dataset.

	error rates and error reduction rates						training times				
dataset	round robin		bagging		boosting		rr	bagging		boosting	
letter	8.80	*0.705*	7.90	*0.633*	5.78	*0.463*	237.34	107.04	*0,451*	72.66	*0.306*
abalone	75.08	*0.957*	76.87	*0.980*	77.88	*0.992*	38.53	26.49	*0.688*	28.02	*0.727*
car	5.84	*0.771*	6.89	*0.908*	3.82	*0.504*	1.39	3.09	*2.227*	0.48	*0.347*
glass	24.77	*0.707*	26.17	*0.747*	27.57	*0.787*	0.90	1.21	*1.337*	0.41	*0.454*
image	2.90	*0.905*	2.90	*0.905*	1.60	*0.500*	6.18	12.53	*2.027*	5.26	*0.851*
lr spectrometer	51.79	*1.011*	48.78	*0.952*	46.70	*0.912*	93.24	45.02	*0.483*	58.61	*0.629*
optical	5.04	*0.547*	4.57	*0.497*	2.46	*0.267*	44.58	81.30	*1.824*	44.32	*0.994*
page-blocks	2.98	*0.964*	2.63	*0.852*	2.58	*0.834*	6.56	17.47	*2.663*	8.65	*1.319*
sat	13.16	*0.953*	9.88	*0.715*	9.32	*0.675*	22.00	70.06	*3.185*	44.67	*2.031*
solar flares (c)	15.69	*0.995*	15.69	*0.995*	16.41	*1.041*	2.52	3.05	*1.212*	0.18	*0.073*
solar flares (m)	4.90	*1.000*	4.90	*1.000*	5.90	*1.206*	2.35	3.75	*1.594*	0.26	*0.113*
soybean	6.73	*0.697*	8.20	*0.848*	6.59	*0.682*	9.24	3.90	*0.421*	0.79	*0.085*
thyroid (hyper)	1.14	*1.024*	1.19	*1.071*	1.03	*0.929*	6.71	14.04	*2.093*	2.72	*0.406*
thyroid (hypo)	0.69	*1.182*	0.50	*0.864*	0.32	*0.545*	5.25	13.06	*2.487*	1.96	*0.373*
thyroid (repl.)	0.74	*1.037*	0.90	*1.259*	0.90	*1.259*	5.26	12.91	*2.455*	2.15	*0.409*
vehicle	29.20	*1.113*	25.41	*0.968*	24.11	*0.919*	1.60	4.61	*2.875*	2.69	*1.677*
vowel	19.49	*0.898*	11.31	*0.521*	8.89	*0.409*	3.85	5.89	*1.532*	5.44	*1.416*
yeast	40.63	*0.939*	41.85	*0.967*	41.85	*0.967*	4.00	5.50	*1.377*	3.88	*0.971*
average (c5.0)	17.69	*0.909*	16.98	*0.864*	16.35	*0.735*			*1.562*		*0.523*
letter	7.85	*0.498*	7.20	*0.457*	8.70	*0.552*	1335.22	12662.54	*9.483*	8565.95	*6.415*
abalone	74.34	*0.916*	78.36	*0.965*	—	—-	174.56	996.19	*5.707*	—	—
car	2.26	*0.186*	9.38	*0.771*	5.09	*0.419*	7.93	49.58	*6.255*	18.45	*2.328*
glass	25.70	*0.743*	29.44	*0.851*	27.10	*0.784*	2.11	4.91	*2.330*	4.21	*2.000*
image	3.46	*0.808*	2.51	*0.586*	1.86	*0.435*	26.99	127.29	*4.716*	132.18	*4.897*
lr spectrometer	53.11	*0.865*	57.82	*0.942*	59.32	*0.966*	836.84	512.42	*0.612*	665.91	*0.796*
optical	3.74	*0.394*	2.86	*0.302*	3.61	*0.381*	285.01	1818.76	*6.381*	1635.90	*5.740*
page-blocks	2.76	*0.816*	2.65	*0.784*	2.89	*0.854*	38.66	231.89	*5.997*	158.55	*4.101*
sat	10.35	*0.794*	10.18	*0.781*	10.52	*0.807*	194.85	1982.23	*10.173*	1114.40	*5.719*
solar flares (c)	15.77	*0.991*	15.91	*1.000*	15.91	*1.000*	7.15	16.59	*2.322*	4.43	*0.620*
solar flares (m)	5.04	*0.921*	5.26	*0.961*	5.47	*1.000*	5.55	9.42	*1.697*	4.55	*0.819*
soybean	6.30	*0.717*	8.20	*0.933*	8.35	*0.950*	30.47	21.66	*0.711*	11.43	*0.375*
thyroid (hyper)	1.11	*0.749*	1.41	*0.945*	1.43	*0.963*	23.10	69.38	*3.004*	60.91	*2.637*
thyroid (hypo)	0.53	*0.955*	0.58	*1.050*	0.90	*1.623*	15.93	51.37	*3.226*	55.70	*3.497*
thyroid (repl.)	1.01	*1.026*	0.98	*0.999*	1.33	*1.350*	16.52	59.38	*3.594*	67.17	*4.066*
vehicle	29.08	*0.957*	26.12	*0.860*	26.71	*0.879*	8.07	44.36	*5.496*	22.03	*2.729*
vowel	18.69	*0.690*	16.26	*0.601*	12.32	*0.455*	18.30	83.46	*4.562*	63.51	*3.471*
yeast	41.78	*0.986*	40.63	*0.959*	41.92	*0.989*	18.21	88.66	*4.867*	23.63	*1.297*
average (ripper)	17.36	*0.747*	18.15	*0.811*	[14.05	*0.801*]			*3.405*		*[2.154]*

a wrapper that writes multiple training sets for the base algorithm to the disk. Second, bagging and boosting always perform a fixed number of up to 10 iterations (ripper's boosting algorithm sometimes stopped before it reached 10 iterations), whereas round robin learning performs $c(c-1)/2$ iterations, where c is the number of classes. There are only 3 problems with less than five classes (*car*, *vehicle*,

Table 3: Pairwise correlation coefficients of the relative improvements of boosting, bagging and round robin over c5.0.

r^2	round robin	boosting	bagging
round robin	1	0.269	0.333
boosting	0.269	1	0.669
bagging	0.333	0.669	1

thyroid (repl.)); in all other cases round robin performs 10 or more iterations. The *letter* domain, for example, has 26 classes, one for each letter in the alphabet. In this domain, the round robin algorithm wrote $25 \times 26/2 = 325$ training sets to the disk. Not surprisingly, c5.0-rr is the slowest algorithm in this domain, but it is only about 3 times slower than 10 iterations of boosting. This illustrates the fact that although the number of members in a round robin ensembles grows quadratically with the number of classes, its run-time complexity only grows linearly (Fürnkranz, 2002b). In other domains, (e.g., the 5-class domain *page-blocks* or the 6-class domain *sat*) round robin is even the fasted algorithms. For ripper, round robin learning is considerably faster than both bagging and boosting. This is no surprise if we consider that both bagging and boosting have to perform internal class binarizations in each iteration, and our theoretical results show that round robin binarization is faster than the unordered class binarization that is used by ripper (Fürnkranz, 2002b). Again, this can be nicely seen at the *letter* dataset, where bagging is almost 10 times as slow as round robin, i.e., a single iteration of bagging was almost as expensive as the entire round robin procedure.

4.2 Integration with Bagging and Boosting

One of the motivations for this study were earlier results (Fürnkranz, 2001) where we observed that the improvements in accuracy obtained by round robin ripper over ripper were quite similar to those obtained by boosted c5.0 over c5.0 on the same problems. Round robin binarization seemed to work well for ripper whenever boosting worked well for c5.0. The correlation coefficient r^2 of the performance ratios ripper-rr/ripper and c5.0-boost/c5.0 was about 0.618, which is in the same range as the correlation coefficient 0.69 that reported for the error reduction rates of bagging and boosting decision trees (Table 3 of Opitz and Maclin, 1999). Table 3 shows the pairwise correlation coefficients of the results for c5.0 of Table 2. Although the correlation between bagging and boosting is consistent with the above-mentioned results of Opitz and Maclin (1999), round robin does not seem to correlate well with bagging and boosting when c5.0 is used as the base learner for all three. On the one hand, this refutes our earlier hypothesis, but on the other hand this throws up an important question: given the weakness of the correlation between round robin and other ensemble methods, it could be the case that they exploit different domain characteristics for getting their respective performance improvements. If this is the case, an integration of round robin learning with bagging and boosting could result in additional performance gains. We try to answer this question in the following.

The integration of bagging and round robin learning can be realized as a straight-forward extension of the basic algorithm, in which multiple classifiers are trained for each pair of classes (analogous to round robin tournaments in sports and games where each team plays each other team several times). In order to guarantee a variety of the individual results, one could either use algorithms with random (Kolen and Pollack, 1991) or randomized (Dietterich, 2000b) components, or use 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). Thus, we realized the integration of bagging and round robin learning by drawing with replacement 10 independent samples of size n of each pairwise classification

Table 4: A comparison of various ways of combining bagging, boosting, and round robin learning. The number show the accuracies and the relative improvement over the base learner c5.0.

dataset	bag + rr		boost + rr		bag + boost		all three	
letter	8.35	*0.669*	5.45	*0.437*	3.98	*0.319*	5.58	*0.447*
abalone	73.45	*0.936*	74.67	*0.951*	75.51	*0.962*	73.76	*0.940*
car	5.38	*0.710*	1.85	*0.244*	5.50	*0.726*	2.78	*0.367*
glass	25.23	*0.720*	22.90	*0.653*	21.03	*0.600*	23.83	*0.680*
image	2.90	*0.906*	1.73	*0.541*	2.21	*0.691*	2.34	*0.731*
lr spectrometer	52.54	*1.026*	51.98	*1.015*	42.94	*0.838*	51.22	*1.000*
optical	3.15	*0.342*	2.54	*0.276*	1.98	*0.215*	2.17	*0.236*
page-blocks	2.58	*0.835*	2.78	*0.900*	2.41	*0.780*	2.54	*0.822*
sat	9.43	*0.682*	9.00	*0.651*	8.72	*0.631*	8.58	*0.621*
solar flares (c)	15.69	*0.995*	16.70	*1.059*	16.70	*1.059*	16.41	*1.041*
solar flares (m)	4.90	*1.000*	5.83	*1.190*	5.54	*1.131*	5.33	*1.088*
soybean	7.61	*0.788*	6.44	*0.667*	5.42	*0.561*	6.44	*0.667*
thyroid (hyper)	1.30	*1.171*	1.33	*1.198*	1.22	*1.099*	1.19	*1.072*
thyroid (hypo)	0.50	*0.862*	0.53	*0.914*	0.34	*0.586*	0.42	*0.724*
thyroid (repl.)	0.77	*1.069*	0.90	*1.250*	0.87	*1.208*	0.93	*1.292*
vehicle	25.30	*0.964*	23.17	*0.883*	24.23	*0.923*	25.30	*0.964*
vowel	17.17	*0.791*	14.75	*0.679*	6.97	*0.321*	11.82	*0.544*
yeast	38.61	*0.893*	40.77	*0.942*	38.68	*0.894*	38.95	*0.900*
average	16.85	*0.838*	16.35	*0.757*	15.31	*0.718*	16.12	*0.749*

problem. Thus we obtained a total of $10c(c-1)/2$ predictions for each c-class problem. Like with the regular round robin procedure, the final prediction was computed by simple voting among these $10c(c-1)/2$ classifiers.

The integration of boosting and round robin learning was simply realized by using c5.0-boost as the base learner inside the round robin procedure. Similarly, boosting and bagging were combined by combining the predictions of ten c5.0-boost classifiers that were trained on ten bootstrap samples of the data. Finally, the combination of all three ensembles was realized by using c5.0-boost as the base learner for the combination of bagging and round robin that is described in the previous paragraph.

Table 4 shows the results. While the combination with bagging or boosting with round robin learning did not produce substantially different results than bagging or boosting alone, the combination of bagging and boosting emerged as a clear winner. In the light of the results of Table 3, which showed that the predictions of round robin learning and both bagging or boosting are less correlated than the predictions of bagging and boosting, this came as a surprise. We would have expected that the classifiers that differ more from each other have a better chance of complementing each other into an even better classifier that exploits the different strengths of each of its constituents. On the other hand, this only confirms previous good results with combinations of bagging and boosting (Pfahringer, 2000; Krieger et al., 2001).

For both bagging and boosting, the average improvement ratios of their combination with round robin learning are not very different from their individual results (compare the shown average improvement ratios to those of table 2). The same pattern could also be found in our previous results with bagging and round robin learning using ripper as a base learner (Fürnkranz, 2002b), but there the trend was much clearer. Integrating bagging and round robin learning in fact produced average results that were very comparable to those of the combination of bagging and boosting with c5.0 as a

base learner, even though c5.0 appears to be a considerably better base learner (at least on the original multi-class problems).

On the other hand, although there are some cases where the combination performs better than both of its constituents, round robin classification with c5.0-boost as a base learner does on average not lead to performance improvements over boosting, and may indeed even hurt performance. In some sense, these results are analogous to those of Schapire (1997) who compared AdaBoost.OC (error-correcting output codes as a binarization scheme for conventional 2-class AdaBoost) with AdaBoost.M1 (Freund and Schapire (1997)'s straight-forward adaptation of AdaBoost for multi-class base learners, a version of which is presumably also implemented in c5.0; Quinlan 1996), and found no significant differences between the two (using c4.5 (Quinlan, 1993) as a base learner). Analogous to our comparison of c5.0-boost and round robin ensembles, 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).

4.3 Discussion

Our results are mixed. On the one hand, we did show that round robin ensembles do improve performance and can be considered as a pre-processing method that reduces the complexity of the decision boundary by mapping a problem to a set of smaller subproblems. On the other hand, our results show that with respect to the gain in accuracy, round robin learning is not *en par* with boosting, and quite likely also not with bagging if we consider that we only tried one setting for the crucial parameter, the number of iterations. The good results of ripper seem to be mostly caused by the inferiority of ripper's one-against-all class binarization strategy. ripper's previous good results in comparison to C4.5 should probably be re-investigated with respect to the dimension of the number of classes in the problem. For example, only 11 of the 37 problems studied by Cohen (1995) were multi-class problems, and we would expect that ripper's good results were mostly on these datasets. Further investigations are necessary for a definite answer on this question.

However, even though they do not reach the same performance level as alternative ensemble methods, we believe that round robin ensembles nevertheless deserve attention. 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. Round robin ensembles on the other hand have the advantage that each element of the ensemble has a well-defined semantics (separating two classes from each other), so that questions like "Why class A and not class B?" can be answered by showing the corresponding theory (or part of it).[4] In many cases, this may be enough for justifying a prediction. 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 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. 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, to which we will briefly return in section 7.

[4] Of course, this assumes that the corresponding binary classifier actually voted for class A. In case it voted for B, but was over-ruled by all other classifiers that did vote for A, one would potentially have to present the all $2(c-1)-1$ classifiers of the ensemble that involve A and B. Nevertheless, we think it should be possible to exploit the clear semantics of each binary classifier for a better approach to explaining a classifiers decisions.

It should also 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. Moreover, each binary task will be smaller than the original task, so that the total training time of a round robin ensemble of size n will be significantly below that of a parallelized version of bagging if each binary classifier can be trained on a separate processor. For the same reason, round robin ensembles may provide a memory-efficient alternative for multi-class tasks, because for problems with a large number of classes, the training sets for each pair of classes may still fit into memory when the entire dataset does not.

5 Weighted Voting

A crucial component of each ensemble technique is the voting procedure that is used for combining the predictions of each member of the ensemble into a final prediction. Proposals range from simple voting to meta-learning techniques like stacking (Wolpert, 1992), arbiters and combiners (Chan and Stolfo, 1995) or grading (Seewald and Fürnkranz, 2001). Meta-learning techniques are not straightforwardly applicable to our approach, as they typically depend on having a fixed number of classifiers in the ensembles (for constructing the meta training set). Several studies have compared voting techniques for combining the predictions of the individual classifiers of an ensemble in various contexts (e.g., Mayoraz and Moreira, 1997; Allwein et al., 2000). Specifically tailored to pairwise classification, there were proposals for combining the class probability estimates of the pairwise classifiers into class probability distributions for the multi-class problems (Hastie and Tibshirani, 1998; Price et al., 1995). More elaborate suggestions aimed at learning separate classifiers for deciding when a given example is of one of the two classes that were used to train a classifier of the pairwise ensemble (Moreira and Mayoraz, 1998), and at organizing the classifiers into an efficient graph structure that can derive a prediction in at most $c - 1$ steps (Platt et al., 2000).

In this section, we will evaluate several ways of using confidence estimates for improving the performance of round robin ensembles. Previous results on other ensemble techniques, most notably boosting (Schapire and Singer, 1999), seem to indicate that techniques that include confidence estimates of the ensemble predictors into the computation of the final predictions are preferable. The basic idea is to generalize the basic voting procedure, which gives equal weight (say 1) to the vote of each binary classifier, in a way that uses a value based on the confidence estimate as a weight for a prediction's vote: if a classifier is quite confident in its classification, its vote will have a higher weight, and low-confidence votes will have lower weight.

For estimating c5.0's confidence in its predictions, we used c5.0's add-on utility that writes out predictions and confidence estimates for a test file.[5] As c5.0 is a commercial product, it is not known how these confidence estimates are computed but it can be assumed that the estimates are based on the purity of the leaf that is used to classify the example (i.e., the largest proportion of training examples in the leaf that were of the same class).

ripper does not directly provide confidence estimates, but its implementation comes with a utility `predict` that is quite similar to the above-mentioned tool of c5.0. Instead of delivering confidence estimates, `predict` writes out the number of positive and negative training examples that were classified by the rule that classified the test example. These numbers can be used to compute straightforward confidence estimates, in particular *purity* $p/(p+n)$ or the *Laplace-estimate* $(p+1)/(p+n+2)$. The latter is also used internally in ripper for breaking ties in case an example is covered by more than one rule.

[5] This utility is freely available from `http://www.rulequest.com/download.html`.

Table 5: The impact of weighted voting. Shown are the results of round robin using unweighted and weighted voting for both c5.0 and ripper, in the form of error rates and improvement ratios of weighted voting over unweighted voting.

	c5.0			ripper				
dataset	unweighted	c5.0 weights		unweighted	purity weights		Laplace weights	
letter	8.80	9.15	*1.040*	7.85	7.90	*1.006*	8.05	*1.025*
abalone	75.08	74.34	*0.990*	74.34	73.69	*0.991*	73.86	*0.994*
car	5.84	5.73	*0.980*	2.26	2.14	*0.949*	2.43	*1.077*
glass	24.77	25.23	*1.019*	25.70	25.70	*1.000*	25.70	*1.000*
image	2.90	2.77	*0.955*	3.42	3.16	*0.924*	3.25	*0.949*
lr spectrometer	51.79	52.73	*1.018*	53.11	54.05	*1.018*	55.74	*1.050*
optical	5.04	5.18	*1.028*	3.74	3.58	*0.957*	3.63	*0.971*
page-blocks	2.98	2.94	*0.988*	2.76	2.72	*0.987*	2.70	*0.980*
sat	13.16	13.21	*1.004*	10.35	10.55	*1.020*	10.54	*1.018*
solar flares (c)	15.69	15.69	*1.000*	15.77	15.69	*0.995*	15.69	*0.995*
solar flares (m)	4.90	4.90	*1.000*	5.04	4.90	*0.971*	4.90	*0.971*
soybean	6.73	6.88	*1.022*	5.86	5.86	*1.000*	5.56	*0.950*
thyroid (hyper)	1.14	1.14	*1.000*	1.11	1.19	*1.071*	1.09	*0.976*
thyroid (hypo)	0.69	0.66	*0.962*	0.53	0.45	*0.850*	0.48	*0.900*
thyroid (repl.)	0.74	0.72	*0.964*	1.01	0.98	*0.974*	0.98	*0.974*
vehicle	29.20	28.84	*0.988*	29.08	28.61	*0.984*	28.96	*0.996*
vowel	19.49	19.19	*0.984*	19.29	18.18	*0.942*	19.09	*0.990*
yeast	40.63	40.30	*0.992*	41.78	41.44	*0.992*	41.58	*0.995*
average	17.69	17.67	*0.994*	17.36	17.23	*0.977*	17.42	*0.987*

Table 5 shows the results of using weighted voting for c5.0, and the two above-mentioned versions of weighted voting for ripper (using purity and Laplace estimates). It can be seen that weighted voting does improve the performance, in particular for ripper, where it lead to improvements for 13 of 18 datasets, thereby reducing the error rate of round robin classification by a factor of 0.977 for purity weights and 0.987 for Laplace weights. The gain for c5.0 is not as large and not as consistent.

We also tested several other versions of weighted voting for ripper, which are not shown here. First, we tried what we call "generalized Laplace estimates" of the form $(p+1)/(p+n+l)$, where l is a user-settable parameter. We could not find any other values that lead to a systematic improvement over $l = 2$. In particular, the option where we set l to the number of classes c lead to considerably worse results. Second, we also tried a different voting procedure in which only predictions that do not originate from a default rule are eligible to vote. The reasoning behind this is that in our setting ripper always learns a pair of theories for each pair of classes, one theory where it learns rules for the first class and classifies all uncovered examples as belonging to the second class, and one theory where the roles of the classes are reversed. We felt that these default rules have entirely different characteristics than regularly learned rules and tried to ignore predictions originating from them. However, this always led to a bad performance. We suspect the reason for this is that if both theories agree, one of them makes a default prediction, which is ignored. Presumably (hopefully) this case happens more often than a disagreement, so that good predictions are quite frequently ignored. This may eventually lead to situations where erroneous predictions get more and more weight.

One problem with the discussed voting procedure is that there is no guarantee that the size of the confidence estimates corresponds to the relative weight that the corresponding vote should have. To investigate this question, we looked at different ways of rescaling the weights. In particular, we transformed the original weights for c5.0's predictions and our weights for the predictions of ripper

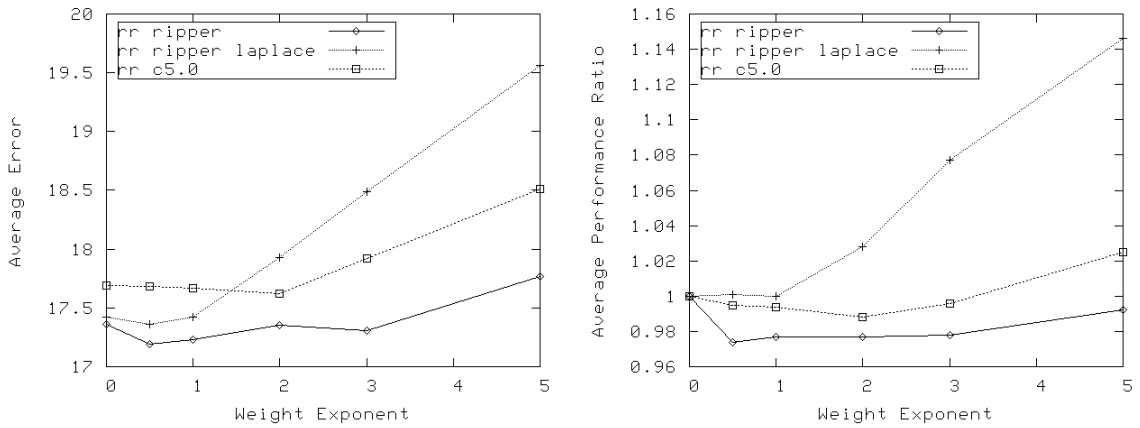

Figure 2: Using different exponents t for weights of the form w^t. The left graph shows the average errors of c5.0, ripper and ripper with Laplace correction, whereas the right graph shows the average improvement ratio of the weighted version of the respective algorithm over the unweighted version (i.e., the version with $t = 0$).

by raising them to the t-th power, i.e., each prediction is weighted with w^t instead of w. This may be viewed as a generalization of the trade-off between weighted and unweighted voting: $t = 0$ is equivalent to unweighted voting, while $t = 1$ corresponds to regular voting as discussed above. Values between 0 and 1 trade off these two extremes, whereas values above 1 successively increase the penalty that is given to weights below 1 (i.e., they increase the importance of a weight being close to 1).

Figure 2 shows the results for values of $t = 0, 0.5, 1, 2, 3, 5$. We show both the average error rates (left graph) and the average performance ratios over unweighted voting (right graph). The graph shows the performance of $t = 1$ is very good throughout, although not optimal. c5.0 achieves better results with $t = 2$, while ripper's results seem to be better around $t = 1/2$. Further experiments would be needed to determine the optimal value, but the results around $t = 1$ seem to be relatively stable. Choosing values $t > 3$ always leads to a degrade in performance (the increase in error continues for $t = 7$ and $t = 10$, which we did not include in the graphs). The results also show that higher exponents seem to penalize the use of the Laplace correction for estimating ripper's votes.

In summary, the results of this section showed that weighted voting may lead to a fairly consistent but moderate additional performance gain for round robin ensembles.

6 Ensemble Size

The size of a round robin ensemble depends on the number of classes in the problem. In this section, we will analyze the behavior of round-robin learning when varying the number of classes. In particular, we were interested in answering one of the open questions of our previous research (Fürnkranz, 2002b), namely whether the advantage of round robin binarization over one-against-all binarization increases with the number of classes.

With this goal in mind, we decided to follow the experimental set-up described by Frank and Hall (2001). They used a set of regression problems and transformed each of them into a series of classification problems, each with a different number of classes. The transformation was performed using equal-frequency discretization on the target variable. Thus the resulting problems were class-

Table 6: Comparison of the error rates of j48 and j48-rr on 29 regression datasets, which were class-discretized to classification problems with 3, 5, and 10 class values.

	3 classes			5 classes			10 classes		
dataset	j48	j48-rr		j48	j48-rr		j48	j48-rr	
Abalone	36.10	35.37	+	53.66	49.37	+	73.16	68.83	+
Ailerons	25.21	24.87	+	43.02	41.83	+	63.35	61.17	+
Delta Ailerons	19.67	19.61	+	44.46	42.54	+	58.73	54.80	+
Elevators	37.76	35.30	+	52.24	47.80	+	71.38	66.29	+
Delta Elevators	30.13	29.17	+	52.31	49.00	+	63.09	57.64	+
2D Planes	13.39	13.39	+	24.63	24.66	−	46.95	45.75	+
Pole Telecom	4.37	4.40	−	4.95	5.08	−	9.16	9.38	−
Friedman Artificial	19.73	19.52	+	35.15	34.31	+	58.96	56.79	+
MV Artificial	0.47	0.48	−	0.81	0.82	−	1.82	1.91	−
Kinematics	36.29	35.74	+	56.37	53.70	+	75.70	72.92	+
CPU Small	21.54	21.01	+	36.19	34.32	+	57.81	54.83	+
CPU Act	19.25	18.82	+	33.23	31.46	+	54.27	51.63	+
House 8L	30.46	29.53	+	49.75	47.32	+	69.59	65.81	+
House 16H	31.79	30.52	+	50.98	47.65	+	70.72	65.97	+
Auto MPG	21.98	24.10	−	40.50	41.58	−	60.40	62.74	−
Auto Price	14.97	14.40	+	37.92	34.53	+	63.21	66.29	−
Boston Housing	25.10	24.11	+	40.38	38.44	+	61.05	58.16	+
Diabetes	48.84	53.26	−	74.42	64.19	+	77.44	75.12	+
Pyrimidines	50.54	46.35	+	58.24	60.81	−	75.81	78.51	−
Triazines	46.72	46.88	−	61.08	63.17	−	83.06	83.17	−
Machine CPU	28.04	25.79	+	42.87	39.86	+	63.44	60.96	+
Servo	24.31	19.88	+	44.67	39.52	+	65.33	57.72	+
Wisconsin Breast C.	63.20	63.35	−	76.60	74.43	+	88.87	86.65	+
Pumadyn 8NH	34.02	33.43	+	53.96	49.22	+	76.18	71.72	+
Pumadyn 32H	22.44	21.54	+	37.35	35.17	+	58.12	54.93	+
Bank 8FM	13.98	14.16	−	26.86	26.25	+	50.29	48.33	+
Bank 32NH	44.21	43.53	+	62.59	60.84	+	75.74	71.04	+
California Housing	20.97	20.68	+	36.66	35.45	+	57.30	56.41	+
Stocks	8.75	8.51	+	13.09	13.42	−	27.40	28.05	−

balanced. We use exactly the same datasets for our evaluation, and compare j48 (the c4.5 clone implemented in the Weka data mining library; Witten and Frank 2000) to j48-rr, a version that uses pairwise classification with j48 as a base learner. The implementation of j48-rr was provided by Richard Kirkby of the Weka team, which gave us the opportunity to check our previous findings with an independent implementation of the algorithm.

Table 6 shows the 10-fold cross-validation error rates of each algorithm on the 29 problems, together with a sign that indicates whether j48 ($-$) or the round robin version ($+$) had the higher estimated accuracy. No significance test was used to compute these individual differences, but in all three settings, j48-rr outperformed j48 in 22 out of 29 datasets. Even with the conservative sign test, which has a comparably high Type II error, we can reject the null hypothesis that the overall performance of j48 and j48-rr is identical on these 29 datasets with 99% confidence. However, four of the datasets (*Pole Telecom*, *MV Artificial*, *Auto MPG*, and *Triazines*) seem to be completely unamenable to pairwise classification, i.e., j48 performs better in all three classification settings.

It is hard to estimate the relative size of the improvement, which is our main concern when we want to show that the performance of round robin ensembles improves with the number of classes.

Table 7: Error, training time, and testing time for a round robin version of j48, a one-against-all version of j48, regular j48, and the binarization technique for ordered classification of Frank and Hall (2001).

	j48-rr	j48-1a		j48		j48-ORD
error						
3	26.82	26.57	*0.99*	27.39	*1.02*	26.30
5	40.92	42.48	*1.04*	42.93	*1.04*	41.43
10	58.40	63.83	*1.10*	60.63	*1.03*	58.92
training time						
3	17.65	35.34	*1.66*	15.99	*0.82*	
5	27.90	53.52	*1.53*	24.58	*0.78*	
10	45.47	84.78	*1.38*	35.76	*0.64*	
testing time						
3	0.66	0.37	*0.50*	0.22	*0.27*	
5	1.64	0.50	*0.30*	0.30	*0.17*	
10	6.81	0.67	*0.14*	0.48	*0.09*	

Inspection of a few cases in Table 6 reveals that on several datasets the advantage of j48-rr over j48 seems to increase with the number of classes, at least for the step from three to five classes (cf., e.g., *Abalone*). In an attempt to make this observation more objective, we summarized the results of these two algorithms in Table 7, and also included the results of j48-1a, a version of j48 that uses a one-against-all binarization. We show the average performance of all algorithms, and the geometric averages of the performance ratios of j48-rr over j48, and j48-1a over j48.

The results show that the performance improvement of round robin over a one-against-all approach increases steadily by both measures. The performance improvement over j48 also increases in absolute terms, but stays about the same in relation to the error rate of j48 (the improvement is always approximately 3% of j48's error rate). This seems to indicate that the one-against-all class binarization becomes more and more dangerous for larger numbers of classes. A possible reason could be that the class distributions of the binary problems in the one-against-all case become more and more skewed for an increasing number of classes (because the number of examples for each class decreases).

We also used these experiments to get the confirmation of an independent implementation for round robin's favorable run-time results over one-against all. The lower two parts of Table 7 shows the summaries for the training and test times. As expected, round robin binarization is considerably faster than a one-against-all approach, despite the fact that round robin binarization generates $c(c-1)/2$ binary problems for a c-class problem, while the one-against-all technique generates only c problems. However, the advantage seems to decrease with an increasing number of classes. This is consistent with our theoretical predictions (Fürnkranz, 2002b, Theorem 11) in case j48's run-time complexity is less than quadratic. Should j48 be slower (i.e., its run-time grows faster than with the square of the training set size) we would have expected that the performance advantage over one-against-all binarization would increase with the number of classes.

For completeness, Table 7 also shows the results for testing time, which are clearly the worst for the round robin case. Each example has to be tested on $c(c-1)/2$ theories for the round robin case, on c theories for the one-against-all case, and on 1 theory for regular j48, so round robin's testing time will grow with the square of the number of classes. If this is a problem, one can consider the use of a technique proposed by Platt et al. (2000), who suggested to organize the classifiers into an efficient graph structure that can derive a prediction in at most $c-1$ steps.

7 A Note on Ordered Classification

As noted above, we basically chose the same experimental setup as Frank and Hall (2001). The only difference between our evaluation procedure and theirs is that we only used a single 10-fold cross-validation, while Frank and Hall (2001) averaged ten 10-fold cross-validation runs. However, these differences are negligible: in the six experiments that we both performed—those using j48 and j48-1a—their average accuracy estimates and our estimates differed by at most 0.05.

This allows us to evaluate the performance of round robin learning in domains with ordered classification. In particular, we can directly compare our results to the results of the to j48-ORD, the algorithm that Frank and Hall (2001) suggested in their work, which is based on an adaptation of a one-against-all class binarization which forms groups of classes based on their order. The performance of j48-ORD (computed from the results shown in Frank and Hall (2001)) is shown in the last column of Table 7. The interesting result is that there is almost no difference between their approach and round robin learning. Apparently general round robin learning is as good for ordered classification as their tailored modification of one-against-all learning. This opens up the question whether a similar adaptation of round robin learning could further improve these results, which we leave open for future work.

In some sense, these results parallel psychological studies which found that it is easier to rank n items by making pairwise comparisons between them than to try to directly construct a ranking (Pyle, 1999, p.16–19).

8 Conclusions

The results of our study are mixed: On the one hand they show that round robin ensembles improve performance not only for learners that depend on class binarization techniques. Thus, round robin binarization can be considered as an interesting pre-processing technique for increasing performance on multi-class problems. On the other hand, however, our results also demonstrate that the performance gain is not as large as the gain for bagging or boosting. Should optimizing accuracy be the only concern, round robin binarization is no competitor to these. However, contrary to bagging and boosting, the systematic way for generating the ensemble members (one for each pair of classes) maintains a certain amount of interpretability, certainly more than bagging and boosting. Thus it may form an interesting trade-off between performance gain and interpretability of the results.

Second, we investigated the use of confidence estimates for combining the predictions of the individual classifiers into an ensemble prediction. We showed that simply using the purity of the rule that made the prediction as a weight for the classifier's vote leads to a small but consistent performance improvement over unweighted voting. This procedure can be further optimized by choosing an appropriate exponent for the weight, but 1 seemed to work fairly well in our experiment. We also looked at using (generalized) Laplace estimates and at the possibility of ignoring predictions that originate from the default rules that ripper learns for the default class, but these techniques did not seem to be helpful.

Our third main result shows that the performance improvements of round robin ensembles increase with the number of classes in the problem (at least for ordered classes). While the improvement over j48 grows approximately linearly with j48's error rate, the growth of the performance increase over one-against-all class binarization is even more dramatic. We believe that this illustrates that handling many classes is a major problem for the one-against-all binarization technique, possibly because the resulting binary learning problems have increasingly skewed class distributions.

Finally, we also showed that round robin binarization is a valid alternative to learning from ordered classification.

Acknowledgments

I would like to thank Eibe Frank and Mark Hall for providing the regression datasets (which were originally collected by Luis Torgo), Richard Kirkby for providing his implementation of pairwise classification in Weka, Johann Petrak for his experimentation scripts and help using them, and the maintainers of and contributors to the UCI collection of machine learning databases.

The author is supported by an APART stipend (no. 10814) of the *Austrian Academy of Sciences*. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture.

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.

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.

R. A. Bradley and M. E. Terry. The rank analysis of incomplete block designs — I. The method of paired comparisons. *Biometrika*, 39:324–345, 1952.

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

P. K. Chan and S. J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pages 90–98. Morgan Kaufmann, 1995.

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

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

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. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems*, pages 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.

E. Frank and M. Hall. A simple approach to ordinal classification. In L. D. Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pages 145–156, Freiburg, Germany, 2001. Springer-Verlag.

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

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

J. Fürnkranz. Pairwise classification as an ensemble technique. In T. Elomma, H. Mannila, and H. Toivonen, editors, *Proceedings of the 13th European Conference on Machine Learning (ECML-02)*, volume 2430 of *Lecture Notes in Artificial Intelligence*, pages 97–110, Helsinki, Finland, 2002a. Springer-Verlag.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002b.

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

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS-97)*, pages 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*, 13(2):415–425, March 2002.

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, editors, *Neurocomputing: Algorithms, Architectures and Applications*, volume F68 of *NATO ASI Series*, pages 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)*, pages 860–867. Morgan Kaufmann, 1991.

U. H.-G. Kreßel. Pairwise classification and support vector machines. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, chapter 15, pages 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, editors, *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pages 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, September 1999.

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

M. Moreira and E. Mayoraz. Improved pairwise coupling classification with correcting classifiers. In C. Nédellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, 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, editors, *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pages 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, editors, *Advances in Neural Information Processing Systems 7 (NIPS-94)*, pages 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)*, pages 725–730. AAAI/MIT Press, 1996.

R. E. Schapire. Using output codes to boost multiclass learning problems. In D. H. Fisher, editor, *Proceedings fo the 14th International Conference on Machine Learning (ICML-97)*, pages 313–321, Nachville, 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 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)*, pages 105–108, Atlanta, GA, 1996.

A. K. Seewald and J. Fürnkranz. An evaluation of grading classifiers. In F. Hoffmann, H. D.J, N. Adams, D. Fisher, and G. Guimaraes, editors, *Advances in Intelligent Data Analysis: Proceedings of the 4th International Conference (IDA-01)*, pages 115–124, Cascais, Portugal, 2001. Springer-Verlag.

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

D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.