Technische Universität Darmstadt
Knowledge Engineering Group
Hochschulstrasse 10, D-64289 Darmstadt, Germany

http://www.ke.informatik.tu-darmstadt.de

Technical Report TUD–KE–2007–03

Sang-Hyeun Park, Johannes Fürnkranz

Efficient Pairwise Classification and Ranking

A short version of this paper appeared as:
Sang-Hyeun Park, Johannes Fürnkranz. Efficient Pairwise Classification,
Proceedings of the 18th European Conference on Machine Learning (ECML-07),
Springer-Verlag, 2007

Efficient Pairwise Classification and Ranking

Sang-Hyeun Park PARK@KE.INFORMATIK.TU-DARMSTADT.DE
Johannes Fürnkranz JUFFI@KE.INFORMATIK.TU-DARMSTADT.DE
Knowledge Engineering Group
Department of Computer Science
TU Darmstadt, Germany

Abstract

Pairwise classification is a class binarization procedure that converts a multi-class problem into a series of two-class problems, one problem for each pair of classes. While it can be shown that for training, this procedure is more efficient than the more commonly used one-against-all approach, it still has to evaluate a quadratic number of classifiers when computing the predicted class for a given example. In this paper, we propose a method that allows a faster computation of the predicted class when weighted or unweighted voting are used for combining the predictions of the individual classifiers. While its worst-case complexity is still quadratic in the number of classes, we show that even in the case of completely random base classifiers, our method still outperforms the conventional pairwise classifier. For the more practical case of well-trained base classifiers, its asymptotic computational complexity seems to be almost linear. We also propose a method for approximating the full class ranking, based on the Swiss System, a common scheme for conducting multi-round chess tournaments. Our results indicate that this adaptive scheme offers a better trade-off between approximation quality and number of performed comparisons than alternative, fixed schemes for ordering the evaluation of the pairwise classifiers.

1. Introduction

Many learning algorithms can only deal with two-class problems. For multi-class problems, they have to rely on *class binarization* procedures that transform the original learning problem into a series of binary learning problems. A standard solution for this problem is the *one-against-all* approach, which constructs one binary classifier for each class, where the positive training examples are those belonging to this class and the negative training examples are formed by the union of all other classes.

An alternative approach, known as *pairwise classification* or *round robin classification* has recently gained attention (Fürnkranz, 2002; Wu et al., 2004). Its basic idea is to transform a c-class problem into $c(c-1)/2$ binary problems, one for each pair of classes. This approach has been shown to produce more accurate results than the one-against-all approach for a wide variety of learning algorithms such as support vector machines (Hsu & Lin, 2002) or rule learning algorithms (Fürnkranz, 2002). Moreover, Fürnkranz (2002) has also proved that despite the fact that its complexity is quadratic in the number of classes, the algorithm can in fact be *trained* faster than the conventional one-against-all technique.[1] However, in order to obtain a final prediction, we still have to combine the predictions of all $c(c-1)/2$ classifiers, which can be very inefficient for large values of c.

The main contribution of this paper is a novel solution for this problem. Unlike previous proposals (such as (Platt et al., 2000); cf. Section 3.2) our approach is not heuristic but is guaranteed to produce exactly the same prediction as the full pairwise classifier, which in turn has been shown to optimize the Spearman rank correlation with the target labels (Hüllermeier & Fürnkranz, 2004b). In essence,

1. It is easy to see this, if one considers 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 example 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.

the algorithm selects and evaluates iterative pairwise classifiers using a simple heuristic to minimize the number of used pairwise classifiers that are needed to determine the correct *top rank* class of the complete (weighted) voting. We will describe and evaluate this algorithm in Section 3.

In Section 4, we will then investigate the case when we are not only interested in the prediction of a single class, but in a ranking of all possible classes. For this case, we propose the so-called *Swiss System*, an algorithm that is commonly used for organizing multi-round chess tournaments. Its key idea is to always evaluate classifiers that have similar positions in the current, incomplete ranking. Our results show that this algorithm offers a good trade-off between the number of evaluated classifiers and the quality of the approximation of the complete ranking.

2. Pairwise Classification

In the following, we assume that a multi-class problem has c classes, which we denote with $c_1, \ldots, c_c$. A pairwise or round robin classifier trains a set of $c(c-1)/2$ *binary classifiers* $C_{i,j}$, one for each pair of classes $(c_i, c_j), i < j$. We will refer to the learning algorithm that is used to train the classifiers $C_{i,j}$ as the *base classifier*. Each binary classifier is only trained on the subset of training examples that belong to the classes c_i and c_j, all other examples are ignored for the training of $C_{i,j}$.

Typically, the binary classifiers are class-symmetric, i.e., the classifiers $C_{i,j}$ and $C_{j,i}$ are identical. However, for some types of classifiers this does not hold. For example, rule learning algorithms will always learn rules for the positive class, and classify all uncovered examples as negative. Thus, the predictions may depend on whether class c_i or class c_j has been used as the positive class. As has been noted in (Fürnkranz, 2002), a simple method for solving this problem is to average the predictions of $C_{i,j}$ and $C_{j,i}$, which basically amounts to the use of a so-called *double round robin* procedure, where we have two classifiers for each pair of classes. We will use this procedure for our results with Ripper.

At classification time, each binary classifier $C_{i,j}$ is queried and issues a vote (a prediction for either c_i or c_j) for the given example. This can be compared with sports and games tournaments, in which each player plays each other player once. In each game, the winner receives a point, and the player with the maximum number of points is the winner of the tournament. In our case, the class with the maximum number of votes is predicted (ties are broken arbitrarily for the larger class). In this paper, we will assume binary classifiers that return class probabilities $p(c_i|c_i \vee c_j)$ and $p(c_j|c_i \vee c_j)$. These can be used for *weighted voting*, i.e., we predict the class that receives the maximum number of votes:

$$c' = \arg\max_{i=1\ldots c} \sum_{j=1}^{c} p(c_i|c_i \vee c_j)$$

This procedure optimizes the Spearman rank correlation with the target ranking (Hüllermeier & Fürnkranz, 2004b). Other algorithms for combining votes exist (cf. *pairwise coupling* (Hastie & Tibshirani, 1998; Wu et al., 2004)), but are not subject of this paper. [2]

Note that weighted or unweighted voting produce a *ranking* of all classes. For prediction problems, one is typically only interested in the *top ranked class*, but in some applications one might also be interested in the complete ranking of classes. We will focus on classification in the next section, ranking will be considered in Section 4.

2. Furthermore, comparisons in (Hüllermeier & Fürnkranz, 2004a) with more advanced methods showed that the simple voting method is competitive with more complex methods.

3. Efficient Pairwise Classification

3.1 The QWeighted Algorithm

Weighted or unweighted voting predicts the top rank class by returning the class with the highest accumulated voting mass after evaluation of all pairwise classifiers. During such a procedure there exist many situations where particular classes can be excluded from the set of possible top rank classes, even if they reach the maximal voting mass in the remaining evaluations. Consider following simple example: Given c classes with $c > j$, if class a has received more than $c - j$ votes and class b lost j votings, it is impossible for b to achieve a higher total voting mass than a. Thus further evaluations with b can be safely ignored.

To increase the reduction of evaluations we are interested in obtaining such exploitable situations frequently. Pairwise classifiers will be selected depending on a *loss* value, which is the amount of potential voting mass that a class has *not* received. More specifically, the loss l_i of a class i is defined as $l_i := p_i - v_i$, where p_i is the number of evaluated incident classifiers of i and v_i is the current vote amount of i. Obviously, the loss will begin with a value of zero and is monotonically increasing.[3] The class with the current minimal loss is one of the top candidates for the top rank class.

Algorithm 1: QWeighted (Quick Weighted Voting)

```
begin
    while c_top not determined do
        c_a ← class c_i ∈ K with minimal l_i;
        c_b ← class c_j ∈ K\{c_a} with minimal l_j & classifier C_{a,b} has not yet been evaluated;
        if no c_b exists then
            c_top ← c_a;
        else
            v_ab ← Evaluate(C_{a,b});
            l_a ← l_a + (1 − v_ab);
            l_b ← l_b + v_ab;
end
```

First the pairwise classifier $C_{a,b}$ will be selected for which the losses l_a and l_b of the relevant classes c_a and c_b are minimal, provided that the classifier $C_{a,b}$ has not yet been evaluated. In the case of multiple classes that have the same minimal loss, there exists no further distinction, and we select a class randomly from this set. Then, the losses l_a and l_b will be updated based on the evaluation returned by $C_{a,b}$ (recall that v_{ab} is interpreted as the amount of the voting mass of the classifier $C_{a,b}$ that goes to class c_a and $1 - v_{ab}$ is the amount that goes to class c_b). These two steps will be repeated until all classifiers for the class c_m with the minimal loss has been evaluated. Thus the current/estimated loss l_m is the correct loss for this class. As all other classes already have a greater loss, c_m is the correct *top rank* class.

Theoretically, a minimal number of comparisons of $c - 1$ is possible (*best case*). Assuming that the incident classifiers of the correct top rank c_{top} always returns the maximum voting amount ($l_{top} = 0$), c_{top} is always in the set $\{c_j \in K | l_j = \min_{c_i \in K} l_i\}$. In addition, c_{top} should be selected as the first class in step 1 of the algorithm among the classes with the minimal loss value. It follows that exactly $c - 1$ comparisons will be evaluated, more precisely all incident classifiers of c_{top}. The algorithm terminates and returns c_{top} as the correct top rank.

The *worst case*, on the other hand, is still $c(c - 1)/2$ comparisons, which can, e.g., occur if all pairwise classifiers classify randomly with a probability of 0.5. In practice, the number of comparisons

3. This loss is essentially identical to the voting-against principle introduced by Cutzu (2003a; 2003b), which we will discuss later on in Section 3.2.

will be somewhere between these two extremes, depending on the nature of the problem. The next section will evaluate this trade-off.

3.2 Related Work

Cutzu (2003a; 2003b) recognized the importance of the voting-against principle and observed that it allows to reliably conclude a class when not all of the pairwise classifiers are present. For example, Cutzu claims that using the voting-against rule one could correctly predict class i even if none of the pairwise classifiers C_{ik} $(k = 1 \dots c, k \neq i)$ are used. However, this argument is based on the assumption that all base classifiers classify correctly. Moreover, if there is a second class j that should ideally receive $c-2$ votes, voting-against could only conclude a tie between classes i and j, as long as the vote of classifier C_{ij} is not known. The main contribution of his work, however, is a method for computing posterior class probabilities in the voting-against scenario. Our approach builds upon the same ideas as Cutzu's, but our contribution is the algorithm that exploits the voting-against principle to effectively increase the prediction efficiency of pairwise classifiers without changing the predicted results.

The voting-against principle was already used earlier in the form of DDAGs Platt et al. (2000), which organize the binary base classifiers in a decision graph. Each node represents a binary decision that rules out the class that is not predicted by the corresponding binary classifier. At classification time, only the classifiers on the path from the root to a leaf of the tree (at most $c-1$ classifiers) are consulted. While the authors empirically show that the method does not lose accuracy on three benchmark problems, it does not have the guarantee of our method, which will always predict the same class as the full pairwise classifier. Intuitively, one would also presume that a fixed evaluation routine that uses only $c-1$ of the $c(c-1)/2$ base classifiers will sacrifice one of the main strengths of the pairwise approach, namely that the influence of a single incorrectly trained binary classifier is diminished in large ensemble of classifiers (Fürnkranz, 2003).

3.3 Evaluation

We compare the QWeighted algorithm with the full pairwise classifier and with DDAGs Platt et al. (2000) on seven arbitrarily selected multi-class datasets from the UCI database of machine learning databases (Hettich et al., 1998). We used four commonly used learning algorithms as base learners (the rule learner Ripper, a Naive Bayes algorithm, the C4.5 decision tree learner, and a support vector machine) all in their implementations in the Weka machine learning library (Witten & Frank, 2005). Each algorithm was used as a base classifier for QWeighted, and the combination was run on each of the datasets. As QWeighted is guaranteed to return the same predictions as the full pairwise classifier, we are only interested in the number of comparisons needed for determining the winning class.[4] These are measured for all examples of each dataset via a 10-fold cross-validation except for *letter*, where the supplied testset was used. Table 1 shows the results.

With respect to accuracy, there is only one case in a total of 28 experiments (4 base classifiers $\times$ 7 datasets) where DDAGs outperformed the QWeighted, which, as we have noted above, optimizes the Spearman rank correlation. This and the fact that, to the best of our knowledge, it is not known what loss function is optimized by DDAGs, confirm our intuition that QWeighted is a more principled approach than DDAGs. It can also be seen that the average number of comparisons needed by QWeighted is much closer to the best case than to the worst case.

Next to the absolute numbers, we show the trade-off between best and worst case (in brackets). A value of 0 indicates that the average number of comparisons is $c-1$, a value of 1 indicates that the value is $c(c-1)/2$ (the value in the last column). As we have ordered the datasets by their respective number of classes, we can observe that this value has a clear tendency to decrease with

4. As mentioned above, we used a double round robin for Ripper for both, the full pairwise classifier and for QWeighted. In order to be comparable to the other results, we, in this case, divide the observed number of comparisons by two.

Table 1: Comparison of QWeighted and DDAGs with different base learners on seven multi-class datasets. The right-most column shows the number of comparisons needed by a full pairwise classifier ($c(c-1)/2$). Next to the average numbers of comparisons for QWeighted we show their trade-off $\frac{n-(c-1)}{\max-(c-1)}$ between best and worst case (in brackets).

			Accuracy		∅ Comparisons		
dataset	c	learner	QWeighted	DDAG	QWeighted	DDAG	full
vehicle	4	NB	45.39	44.92	4.27 (*0.423*)	3	6
		SMO	75.06	75.06	3.64 (*0.213*)		
		J48	71.99	70.92	3.96 (*0.320*)		
		JRip	73.88	72.46	3.98 (*0.327*)		
glass	7	NB	49.07	49.07	9.58 (*0.238*)	6	21
		SMO	57.01	57.94	9.92 (*0.261*)		
		J48	71.50	69.16	9.69 (*0.246*)		
		JRip	74.77	74.30	9.75 (*0.250*)		
image	7	NB	80.09	80.09	9.03 (*0.202*)	6	21
		SMO	93.51	93.51	8.29 (*0.153*)		
		J48	96.93	96.75	8.55 (*0.170*)		
		JRip	96.62	96.41	8.75 (*0.183*)		
yeast	10	NB	57.55	57.21	15.86 (*0.191*)	9	45
		SMO	57.68	57.41	15.52 (*0.181*)		
		J48	58.56	57.75	15.48 (*0.180*)		
		JRip	58.96	58.09	15.87 (*0.191*)		
vowel	11	NB	63.84	63.64	17.09 (*0.158*)	10	55
		SMO	81.92	81.52	15.28 (*0.117*)		
		J48	82.93	78.28	17.13 (*0.158*)		
		JRip	82.42	76.67	17.42 (*0.165*)		
soybean	19	NB	92.97	92.97	27.70 (*0.063*)	18	171
		SMO	94.14	93.41	28.36 (*0.068*)		
		J48	93.56	91.80	29.45 (*0.075*)		
		JRip	94.00	93.56	27.65 (*0.063*)		
letter	26	NB	63.08	63.00	44.40 (*0.065*)	25	325
		SMO	83.80	82.58	42.26 (*0.058*)		
		J48	91.50	86.15	47.77 (*0.076*)		
		JRip	92.33	88.33	45.01 (*0.068*)		

the number of the classes. For example, for the 19-class *soybean* and the 26-class *letter* datasets, only about $6-7\%$ of the possible number of additional pairwise classifiers are used, i.e., the total number of comparisons seems to grow only linearly with the number of classes. This can also be seen from Figure 1a, which plots the datasets with their respective number of classes together with a curve that indicates the performance of the full pairwise classifier.

Finally, we note that the results are qualitatively the same for all base classifiers. QWeighted does not seem to depend on a choice of base classifiers. For a more systematic investigation of the complexity of the algorithm, we performed a simulation experiment. We assume classes in the form of numbers from $1 \dots c$, and, without loss of generality, 1 is always the correct class. We further assume pairwise base pseudo-classifiers $i \prec_\epsilon j$, which, for two numbers $i < j$, return *true* with a probability $1-\epsilon$ and *false* with a probability ϵ. For each example, the QWeighted algorithm is applied to compute a prediction based on these pseudo-classifiers. The setting $\epsilon = 0$ (or $\epsilon = 1$)

Table 2: Average number n of pairwise comparisons for various number of classes and different error probabilities ϵ of the pairwise classifiers, and the full pairwise classifier. Below, we show their trade-off $\frac{n-(c-1)}{\max-(c-1)}$ between the best and worst case, and an estimate of the growth ratio $\frac{\log(n_2/n_1)}{\log(c_2/c_1)}$ of successive values of n.

c	$\epsilon = 0.0$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.5$	full
5	5.43	5.72	6.07	6.45	6.90	7.12	10
	0.238 | —	*0.287* | —	*0.345* | —	*0.408* | —	*0.483* | —	*0.520* | —	
10	14.11	16.19	18.34	21.90	25.39	28.74	45
	0.142 | **1.378**	*0.200* | **1.501**	*0.259* | **1.595**	*0.358* | **1.764**	*0.455* | **1.880**	*0.548* | **2.013**	
25	42.45	60.01	76.82	113.75	151.19	198.51	300
	0.067 | **1.202**	*0.130* | **1.430**	*0.191* | **1.563**	*0.325* | **1.798**	*0.461* | **1.974**	*0.632* | **2.109**	
50	91.04	171.53	251.18	422.58	606.74	868.25	1,225
	0.036 | **1.101**	*0.104* | **1.515**	*0.172* | **1.709**	*0.318* | **1.893**	*0.474* | **2.005**	*0.697* | **2.129**	
100	189.51	530.17	900.29	1,684.21	2,504.54	3,772.45	4,950
	0.019 | **1.058**	*0.089* | **1.628**	*0.165* | **1.842**	*0.327* | **1.995**	*0.496* | **2.045**	*0.757* | **2.119**	

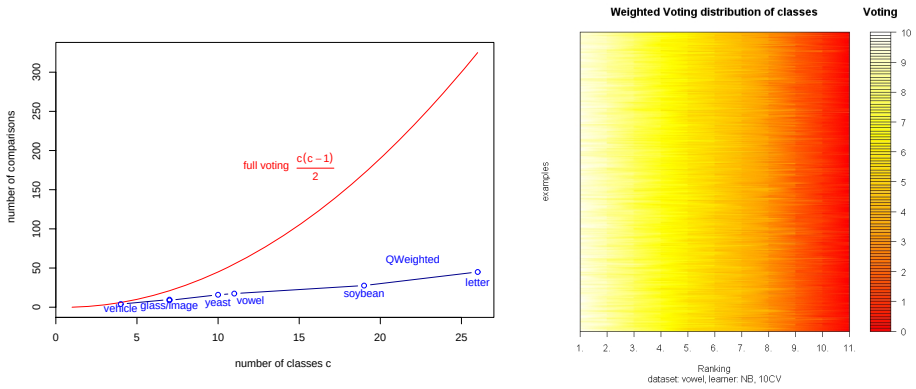
corresponds to a pairwise classifier where all predictions are consistent with a total order of the possible class labels, and $\epsilon = 0.5$ corresponds to the case where the predictions of the base classifiers are entirely random.

Table 2 shows the results for various numbers of classes ($c = 5, 10, 25, 50, 100$) and for various settings of the error parameter ($\epsilon = 0.0, 0.05, 0.1, 0.2, 0.3, 0.5$). Each data point is the average outcome of 1000 trials with the corresponding parameter settings. We can see that even for entirely random data, our algorithm can still save about 1/4 of the pairwise comparisons that would be needed for the entire ensemble. For cases with a total order and error-free base classifiers, the number of needed comparisons approaches the number of classes, i.e., the growth appears to be linear.

To shed more light on this, we provide two more measures below each average: the lower left number (in italics) shows the trade-off between best and worst case, as defined above. The result confirms that for a reasonable performance of the base classifiers (up to about $\epsilon = 0.2$), the fraction of additional work reduces with the number of classes. Above that, we start to observe a growth. The reason for this is that with a low number of classes, there is still a good chance that the random base classifiers produce a reasonably ordered class structure, while this chance is decreasing with increasing numbers of classes. On the other hand, the influence of each individual false prediction of a base classifier decreases with an increasing number of classes, so that the true class ordering is still clearly visible and can be better exploited by the QWeighted algorithm.

This is illustrated in Figure 1b, which shows the distribution of the votes produced by the SVM base classifier for the dataset *vowel*. As shown on the scale to right, different color codes are used for encoding different numbers of received votes. Each line in the plot represents one example, the left shows the highest number of votes, the right the lowest number of votes. If all classes receive the same number of votes, the area should be colored uniformly. However, here we observe a fairly clear change in the color distribution, the bright areas to the left indicating the the top-rank class often receives nine or more votes, and the areas to the right indicating that the lowest ranking class typically receives less than one vote (recall that we use weighted voting).

We tried to directly estimate the exponent of the growth function of the number of comparisons of QWeighted, based on the number of classes c. The resulting exponents, based on two successive measure points, are shown in bold font below the absolute numbers. For example, the exponent of the growth function between $c = 5$ and $c = 10$ is estimated (for $\epsilon = 0$) as $\frac{\log(14.11/5.43)}{\log(10/5)} \approx 1.378$. We



(a) (b)

Figure 1: a) Efficiency of QWeighted in comparison to a full pairwise classifier, b) Distribution of votes for *vowel* (11-class problem, base learner NB). The x-axis describes the ranking positions.

can see that in the growth rate starts almost linearly (for a high number of classes and no errors in the base classifiers) and approaches a quadratic growth when the error rate increases.[5]

In addition to the small datasets from Table 1, we evaluated the QWeighted algorithm on three more real world datasets with a relative high number of classes:

Uni-label RCV1-v2

RCV1-v2 (Lewis et al., 2004) is a dataset consisting of over 800,000 categorized news articles from Reuters, Ltd. For the category *topic* multiple labels from a total of 103 hierarchically organized labels are assigned to the instances. We transformed this original multi-label dataset to uni-label to be suitable for the QWeighted algorithm. For each instance, the label with the greatest depth in the hierarchical tree among the assigned labels was selected as true class label. In case, that multiple labels have the same greatest depth, the respective instance was removed. We applied this procedure on the provided trainset and testset no. 0 by Lewis et al. resulting to a multi-classification dataset with 100 classes, 23,149 train- and 199,328 test-instances with at least one positive example for each of the 100 classes.[6] We will refer to this created dataset as *urcv1-v2*.

ASTRAL 2 & 3

These datasets describe protein sequences retrieved from the SCOP 1.71 protein database (Murzin et al, 1995). We used ASTRAL (Brenner et al., 2000) to filter these sequences so that no two sequences share greater than 95% identity. The class labels are organized in a 3-level hierarchy, consisting of protein folds, superfamilies and families (in descending order). *astral3* consists of 1,588 classes and contains the original hierarchy. To fill the gap between datasets *urcv1-v2* and *astral3* in terms of number of classes, we constructed a second dataset *astral2* by limiting the hierarchical depth to 2. So, two instances which previously shared the same superfamily x are now assigned to superfamily x as new class label. By decreasing the depth, the number of classes were reduced to 971. Both datasets have 13,006 instances and 21 numeric attributes (20 amino acids plus selenocysteine).

5. At first sight, it may be surprising that some of the numbers are greater than 2. This is a result of the fact that $c(c-1)/2 = c^2 - c/2$ is quadratic *in the limit*, but for low values of c, the subtraction of the linear term $c/2$ has a more significant effect. Thus, e.g., the estimated growth of the full pairwise classifier from $c = 5$ to $c = 10$ is $\frac{\log(45/10)}{\log(10/5)} \approx 2.17$.
6. The amount of train and test-instances is identical to their unmodified versions. So, there was in no case an instance with multiple possible candidates as most specific label. Furthermore, only one of the originally 101 assigned labels was removed by the described procedure.

Table 3: a) Average number of pairwise comparisons for *urcv1-v2* under various number of feature set sizes s (base learner J48). b) Results of QWEIGHTED for datasets with a relative high number of classes. Below, we show their trade-off $\frac{n-(c-1)}{\max-(c-1)}$ between the best and worst case (base learner J48).

(a)

	Accuracy		∅ Comparisons	
s	1-vs-1	Direct	QW	full
250	60.8986	53.4260	297.33	4,950
500	62.2828	54.4108	306.39	4,950
1,000	61.5674	54.1043	310.89	4,950
2,000	62.0801	54.5458	312.62	4,950
4,000	62.1107	53.7747	312.60	4,950
8,000	62.1072	53.4350	312.29	4,950
14,000	62.0901	53.9894	312.21	4,950
20,000	61.3220	53.6568	312.35	4,950

(b)

dataset	c	QW	full	Acc.
urcv1-v2	100	312.62 (*0.0440*)	4,950	62.1
astral2	971	9,490.81 (*0.0018*)	470,935	24.8
astral3	1,588	28,476.20 (*0.0213*)	1,260,078	20.8

Dataset *urcv1-v2* uses more than $40,000$ features (stemmed words). To reduce the complexity of the multi-classification problem, χ^2-based feature selection (Yang & Pedersen, 1997) was applied. After computing the ranking of features with decreasing χ^2 scores, we selected various numbers s of the top ranked features ($s = 250, 500, 1000, 2000, 4000, 8000, 14000, 20000$). The decision tree learner *J48* was used as base learner, because of its fast computational speed and for validation, we used the transformed testset. The results are shown in Table 3a. It is surprising that the accuracy value is only minimally influenced by the feature set size. All accuracy values lies within $\simeq$ 61-62 percent for the pairwise approach and approx. 54 without any binarization. In fact, the pairwise approach reaches its maximum accuracy when using only 500 features. As a side note, we can see that the pairwise approach outperforms the direct multi-class capability (3rd column) of J48 in terms of accuracy.

The maximal number of pairwise comparisons (at $s = 2000$) from Table 3a is listed together with the results of the QWEIGHTED algorithm for *astral2* and *astral3* in Table 3b. For *astral2* and *astral3* 66 percent of all instances were used for training and the rest for testing. Once again, trade-off values were estimated for the average number of pairwise comparisons. As these values show, QWEIGHTED uses only a fairly small amount compared to a full voting aggregation and tends clearly to the best case than to the worst case ($c(c-1)/2$ comparisons). One can see an increasing growth of the trade-off values between *astral2* and *astral3*. However, this effect can be explained with the general poor classification accuracy of protein sequences. According to the simulation results, there exist a correlation between performance of QWEIGHTED and performance of the underlying base classifiers. The decreased accuracy on *astral3* compared to *astral2* (right-most column) indicates weaker base classifiers, which leads to a increasing number of needed pairwise comparisons.

In summary, our results indicate that the QWEIGHTED algorithm always increases the efficiency of the pairwise classifier: for high error rates in the base classifiers, we can only expect improvements by a constant factor, whereas for the practical case of low error rates we can also expect a significant reduction in the asymptotic algorithmic complexity.

4. Efficient Pairwise Ranking

Pairwise classification not only delivers a single class label but a ranking of all classes, and in many applications this is of interest. An example might be in speech recognition to combine the ranking of possible recognized words with further background information like context or grammar to gain

a higher accuracy. However, the QWeighted algorithm in the previous section focuses entirely on finding the top rank. In this section, we will briefly discuss an algorithm for efficient computation of a complete ranking.

4.1 The Swiss Algorithm

Our algorithm is based on the *swiss system* that is commonly used in chess tournaments (FIDE, 2006). Its fundamental ideas are that all players play a fixed number of rounds, where in each round players of similar strength are paired with each other. After each round, a ranking is determined based upon the points collected so far, and the next pairings are drawn based on the current ranking (players with the same number of points play each other). Given n entrants, a limitation of $\log(n)$ rounds is typical, resulting in $n \cdot \log(n)$ games, as opposed to the quadratic amount in the usual round-robin system. In addition, no pairing will be evaluated twice.

Algorithm 2: Swiss system

```
begin
    for round= 1 to max_round do
        pair classes that have a similar score, starting with classes with byes;
        pair the remaining classes among each other;
        remaining classes receive a bye;
        evaluate the corresponding classifiers and update the loss values for each class;
end
```

We adopted this system for use in pairwise classification, so that it computes a ranking based on weighted voting (Algorithm 2). Contrary to the regular Swiss system, this algorithm does not try to optimize the pairings, but is greedily producing pairings in various stages. First, it pairs classes that had received a bye in the last round, then classes with a similar score, and then all remaining classes. As a result of the greedy pairing, we may have situations, where for certain classes no valid pairings can be found, e.g., when all remaining classes were already previously paired with the specific class. In this situation the class receives a *bye*, which guarantees a pairing in the beginning of the next round, even if there is no similarly rated class available. The main advantage of this greedy approach is that we can also easily linearize it, i.e., we can evaluate the ranking quality not only after complete rounds, but after each individual pairing.

4.2 Evaluation

Unfortunately, most datasets including those from the UCI Database provide only the correct top rank class but no reference ranking of classes. The main reasons may be that the main purpose of those datasets were aimed at classification, and that even with external expert knowledge it is difficult to provide a full ranking of classes.[7] For this reason, we used the ranking that has been predicted from a full voting procedure as our reference ranking.

For all datasets Ripper (Cohen, 1995), more precisely JRIP, its implementation in Weka, was used as base-learner. Ordinary round-based evaluations of the swiss-system leads to a small number of data points.[8] For the purpose of smoother curves, we plotted the estimated rankings after each single evaluation of a binary classifier. There was no upper limit to the number of rounds, so we could continue until all pairwise comparisons have been performed.

7. Pyle (1999) acknowledges this difficulty and, interestingly, proposes a pairwise technique for ranking a list of options.

8. $\leq \frac{(c(c-1)/2)}{\lfloor c/2 \rfloor}$, the number is not constant because of possible bye-situations within Swiss-system, please refer to (FIDE, 2006) for details.

To measure the accuracy two error values were deployed:

- the *normalized ranking error*, where the error of each position between predicted ranking τ_p and true ranking τ_t is counted.

$$D_R = \frac{1}{k_c} \cdot \sum_{1}^{c} (\tau_t(i) - \tau_p(i))^2$$

  The normalizing constant $k_c = c(c^2 - 1)/3$ is the maximal possible ranking error. Note that this is a linear transformation of the Spearman rank correlation into a loss function with value range $[0, 1]$.

- the *position error*, which uses the position of the true class $\hat{c}$ in the predicted ranking

$$D_P = \tau_p^{-1}(\hat{c}) - 1$$

  It may be viewed as a simple generalization of $\{0, 1\}$-loss for problems in which one is interested in finding the correct label with a minimum amount of trials (Hüllermeier & Fürnkranz, 2005).

All values were estimated with 10-fold cross-validation except for dataset *letter*, where the supplied testset was used.

First, we investigated the ranking error, i.e. the deviation of the ranking obtained by Swiss from the ranking of the full pairwise classifier. Figure 2a shows the ranking error curve for *vowel*, but all datasets have similar characteristics (Park, 2006). It shows that the ranking error decreases exponentially with increasing numbers of used pairwise classifiers, i.e., in the beginning, the ranking improves substantially with each additional comparisons until a certain performance level is reached. After that, each additional comparison yields only small improvements, but the approximation is never 100% accurate.

In order to scale this performance, we compared it to various other ordered methods:

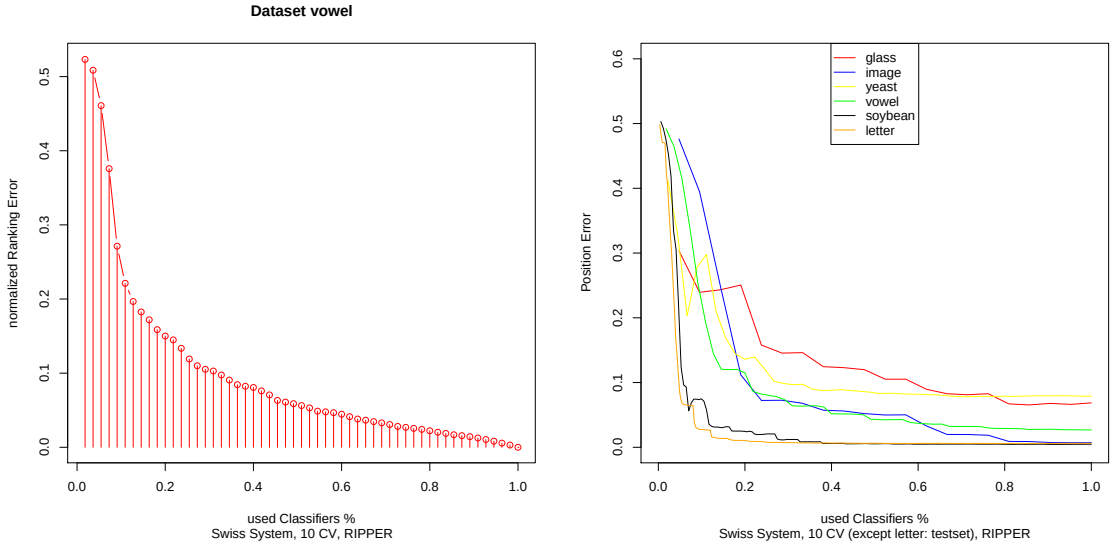
- ascending ($C_{1,2}$,$C_{1,3}$,...,$C_{1,n}$,$C_{2,3}$,$C_{2,4}$,...)
- descending ($C_{n,n-1}$,...,$C_{n,1}$,$C_{n-1,n-2}$,$C_{n-1,n-3}$,...)
- diagonal ($C_{2,1}$,$C_{3,1}$,$C_{3,2}$,$C_{4,3}$,$C_{4,2}$,$C_{4,1}$,$C_{5,4}$,...)
- and a random order.

Class 1 is the first class that was listed in the dataset, 2 for the second and so on. Table 4 shows the average performance ratios $r = \frac{D_{i,s}}{D_{i,\text{SWISS}}}$ where $D_{i,s}$ is the ranking error of one of the four ordered methods, and $D_{i,\text{SWISS}}$ is the performance of the Swiss algorithm. Thus, a value greater than 1 indicates a superior performance of the Swiss system. The Swiss system always outperformed the *diagonal* and *random* methods, typically by more than 10%. The Swiss system also wins in the majority of the cases against *ascending* and *descending*, but here we can find two exceptions, where one of these two wins by a comparably small margin (about 4%). A possible explanation could be that in these datasets, the given fixed class order could contain information that holds for most of the datasets (e.g., the majority class is first in the list). The *ascending* and *descending* ordering schemes could exploit such informations (also note that in these cases a good performance of one is also accompanied with a bad performance of the other), whereas the other two methods distribute the pairwise classifiers for each class fairly evenly across the sequence. Nevertheless, it seems safe to conclude that the Swiss system outperforms all fixed schemes.

The ascending order for datasets *image* and *yeast* and the descending order for *vowel* seem to break ranks, while most values demonstrate the superior performance of the Swiss system. These exceptions may be based on specific characteristics of these datasets and can be probably ignored in the general case.

Table 4: Comparison of the ranking error between Swiss system and four class-ordered systems. The values show the average performance ratio, where a value of 1 means identical to the Swiss system and a value greater 1 indicates a better performance of the Swiss algorithm.

dataset	ascending	descending	diagonal	random
vehicle	1.102	1.183	1.316	1.170
glass	1.122	1.328	1.782	1.413
image	0.968	1.491	1.181	1.106
yeast	1.123	1.667	1.792	1.486
vowel	1.285	0.965	1.381	1.106
soybean	1.053	1.193	1.262	1.028
letter	1.084	1.419	1.263	1.193



(a) (b)

Figure 2: a) Ranking error curve for *vowel* (base learner Ripper), b) Position error curves using Swiss system for all datasets (base learner Ripper)

Figure 2b shows the results for the position error. As you can see the position error converges faster than the ranking error to the position error of the full pairwise classifier. Contrary to the ranking error, not all pairwise comparisons are needed to predict the winning class. This observation is consistent with the consideration that if all incident classifiers of the correct top rank are evaluated during such a sequence, the result is a position error of zero and further pairwise comparisons do not affect it anymore.

However, the saturation point, for which we can say that a reasonable position error has been obtained, differs between datasets. Table 5 takes a closer look at this issue, summarizing the results ordered by the number of classes. Here we computed a *relative position error* D_R, which is the position of the top-ranked class of the full ranking in the incomplete ranking. We assume that the ratings have converged, when $D_R < 0.02$ (third column) or $D_R = 0.0$ (last column). For $D_R < 0.02$, we can see Since the datasets in the legend are ordered by their number of classes a relation between the number of classes and the convergence of the position error. The higher the number of classes

Table 5: Position error of all datasets. The last two columns describe the average number of pairwise comparisons for a position error D_P within the threshold $D_P < 0.02$ and $D_P = 0$.

dataset	c	$D_R < 0.02$	$D_R = 0$
vehicle	4	5 (83%)	6 (100%)
glass	7	17 (81%)	19 (90%)
image	7	14 (67%)	20 (95%)
yeast	10	16 (36%)	44 (98%)
vowel	11	27 (49%)	52 (95%)
soybean	19	39 (23%)	145 (85%)
letter	26	34 (10%)	315 (97%)

of a dataset, the smaller the percentage of pairwise comparisons are needed to achieve a acceptable position error. This trend is less clear for a perfect position error, as can be seen in the fourth column. Here it can also be seen that in most cases almost all pairwise comparisons were needed for a perfect agreement with the predictions of the full pairwise classifier. Note however that, contrary to the QWeighted algorithm, the Swiss algorithm still attempts to maximize the full ranking, and does not focus on the top ranking only (as QWeighted does).

5. Conclusions

In this paper, we have proposed two novel algorithms that allow to speed up the prediction phase for pairwise classifiers. The QWeighted algorithm will always predict the same class as the full pairwise classifier, but the algorithm is close to linear in the number of classes, in particular for large numbers of classes, where the problem is most stringent. For very hard problems, where the performance of the binary classifiers reduces to random guessing, its worst-case performance is still quadratic in the number of classes, but even there practical gains can be expected.

The Swiss algorithm does not focus on the top-ranked class, but tries to approximate the complete ranking of the classes. While we could not directly evaluate the ranking ability of this algorithm (because of a lack of datasets with complete class ranking information for each example), we could demonstrate that this adaptive algorithm offers a better trade-off than alternative, fixed evaluation orders for the pairwise classifiers, and therefore allows for a better trade-off between the number of evaluated pairwise classifiers and the quality of the approximation of the full class ranking.

A restriction of our approaches is that they are only applicable to combining predictions via voting or weighted voting. There are various other proposals for combining the class probability estimates of the base classifiers into an overall class probability distribution (this is also known as *pairwise coupling* (Hastie & Tibshirani, 1998; Wu et al., 2004)). Nevertheless, efficient alternatives for other pairwise coupling techniques are an interesting topic for further research.

Acknowledgments

This research was supported by the *German Science Foundation (DFG).*

References

Brenner, S. E., Koehl, P., & Levitt, M. (2000). The ASTRAL compendium for sequence and structure analysis. *Nucleic Acids Research*, *28*, 254–256.

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

Cutzu, F. (2003a). How to do multi-way classification with two-way classifiers. *Proceedings of the International Conference on Artificial Neural Networks (ICANN-03)* (pp. 375–384). Springer-Verlag.

Cutzu, F. (2003b). Polychotomous classification with pairwise classifiers: A new voting principle. *Proceedings of the 4th International Workshop on Multiple Classifier Systems* (pp. 115–124). Springer, Berlin.

FIDE (2006). Fide swiss rules. In *Fide handbook*, chapter 4. World Chess Federation — Federation Internationale des Echecs. `http://www.fide.com/official/handbook.asp?level=C04`.

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

Fürnkranz, J. (2003). Round robin ensembles. *Intelligent Data Analysis*, *7*, 385–404.

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

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

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, *13*, 415–425.

Hüllermeier, E., & Fürnkranz, J. (2004a). Comparison of ranking procedures in pairwise preference learning. *Proceedings of the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-04)*. Perugia, Italy.

Hüllermeier, E., & Fürnkranz, J. (2004b). Ranking by pairwise comparison: A note on risk minimization. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-04)*. Budapest, Hungary.

Hüllermeier, E., & Fürnkranz, J. (2005). Learning label preferences: Ranking error versus position error. *Advances in Intelligent Data Analysis: Proceedings of the 6th International Symposium (IDA-05)* (pp. 180–191). Madrid, Spain: Springer-Verlag.

Lewis, D. D., Yang, Y., Rose, T., & Li, F. (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, *5*, 361–397.

Murzin, A. G., Brenner, S. E., Hubbard, T., Chothia, C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, *247*, 536–540.

Park, S.-H. (2006). Efficient classification and ranking with pairwise comparisons. Diploma Thesis, TU-Darmstadt, In German.

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

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

Witten, I. H., & Frank, E. (2005). *Data mining — practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers. 2nd edition.

Wu, T.-F., Lin, C.-J., & Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, *5*, 975–1005.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *The Fourteenth International Conference on Machine Learning (ICML-97)*, 412–420. Morgan Kaufmann.