Efficient Pairwise Classification

Sang-Hyeun Park and Johannes Fürnkranz

TU Darmstadt, Knowledge Engineering Group,
D-64289 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.

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 [3, 12]. 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 [7] or rule learning algorithms [3]. Moreover, Fürnkranz [3] 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. 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 [10]; 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 [8]. In essence, 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.

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 [3], 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 all players play each other 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 [8]. Other algorithms for combining votes exist (cf. *pairwise coupling* [5, 12]), but are not subject of this paper.

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. Due to space restrictions we will focus here only on classification. However, the extended version of this paper [9] deals also with the problem of

efficiently predicting a full class ranking. We propose for this case the so-called *Swiss-System*, a common scheme for conducting multi-round chess tournaments. 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.

3 Efficient Pairwise Classification

3.1 The Quick Weighted Voting (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.[1] The class with the current minimal loss is one of the top candidates for the top rank class. First the pairwise classifier $C_{a,b}$ will

Algorithm 1: QWeighted

```
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} not yet 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;
```

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

[1] This loss is essentially identical to the voting-against principle introduced by [1, 2], which we will discuss later on in Section 3.2.

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 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 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 [1, 2] 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} ($k = 1 \ldots c, k \neq i$) 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 [10], 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 [4].

Table 1. Comparison of QWeighted and DDAGs with different base learners on seven multi-class datasets. 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*)		

3.3 Evaluation

We compare the QWeighted algorithm with the full pairwise classifier and with DDAGs [10] on seven arbitrarily selected multi-class datasets from the UCI database of machine learning databases [6]. 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 [11]. 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.[2] 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

[2] 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.

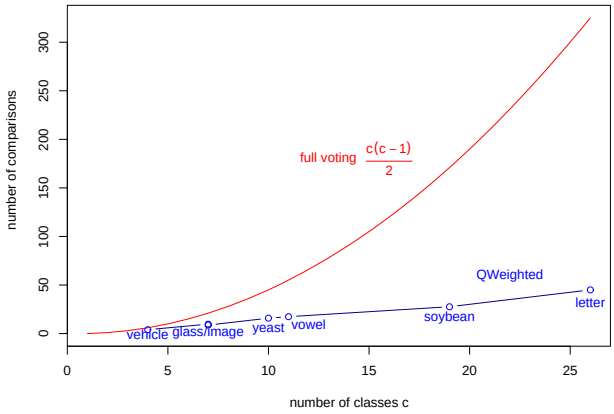

Fig. 1. Efficiency of QWeighted in comparison to a full pairwise classifier

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 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 Fig. 1, 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 $1 \ldots c$, 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 $\epsilon = 1$) 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 ($\epsilon = 0.0, 0.05, 0.1, 0.2, 0.3, 0.5$).

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	1225
	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	1684.21	2504.54	3772.45	4950
	0.019 | **1.058**	*0.089* | **1.628**	*0.165* | **1.842**	*0.327* | **1.995**	*0.496* | **2.045**	*0.757* | **2.119**	

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 QWeighted. 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 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.

In summary, our results indicate that QWeighted 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 Conclusions

In this paper, we have proposed a novel algorithm that allows to speed up the prediction phase for pairwise classifiers. QWeighted 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. A restriction of our approach is that it is 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* [5, 12]). Nevertheless, efficient alternatives for other pairwise coupling techniques are an interesting topic for further research.

References

1. 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.
2. 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.
3. Fürnkranz, J. (2002). Round robin classification. *Journal of Machine Learning Research*, *2*, 721–747.
4. Fürnkranz, J. (2003). Round robin ensembles. *Intelligent Data Analysis*, *7*, 385–404.
5. Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *Advances in Neural Information Processing Systems 10 (NIPS-97)* (pp. 507–513). MIT Press.
6. 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.
7. 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.
8. Hüllermeier, E., & Fürnkranz, J. (2004). Ranking by pairwise comparison: A note on risk minimization. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-04)*. Budapest, Hungary.
9. Park, S.-H., & Fürnkranz, J. (2007). Efficient Pairwise Classification and Ranking. *Technical Report TUD-KE-2007-3*. Knowledge Engineering Group, TU Darmstadt.
10. 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.
11. Witten, I. H., & Frank, E. (2005). *Data mining — practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers. 2nd edition.
12. 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.