Combining Pairwise Classifiers with Stacking

Petr Savicky and Johannes Fürnkranz

[1] Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodarenskou Vezi 2, 182 07 Praha 8, Czech Republic
savicky@cs.cas.cz
[2] Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
juffi@oefai.at

Abstract. Pairwise classification is the technique that deals with multi-class problems by converting them into a series of binary problems, one for each pair of classes. The predictions of the binary classifiers are typically combined into an overall prediction by voting and predicting the class that received the largest number of votes. In this paper we try to generalize the voting procedure by replacing it with a trainable classifier, i.e., we propose the use of a meta-level classifier that is trained to arbiter among the conflicting predictions of the binary classifiers. In our experiments, this yielded substantial gains on a few datasets, but no gain on others. These performance differences do not seem to depend on quantitative parameters of the datasets, like the number of classes.

1 Introduction

A recent paper [6] showed that pairwise classification (aka *round robin* learning) is superior to the commonly used one-against-all technique for handling multi-class problems in rule learning. Its basic idea is to convert a c-class problem into a series of two-class problems by learning one classifier for each pair of classes, using only training examples of 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. The most important finding of [6, 7] was that this procedure not only increases predictive accuracy, but it is also no more expensive than the more commonly used one-against-all approach.

Previous work in this area only assumed voting techniques for combining the predictions of the binary classifiers, i.e., predictions were combined by counting the number of predictions for each individual class and assigning the class with the maximum count to the example (ties were broken in favor of larger classes). In this paper, we evaluate the idea of replacing the voting procedure with a trainable classifier.

We will start with a brief recapitulation of previous results on round robin learning (Section 2). Section 3 presents our algorithm based on stacking. The experimental setup is sketched in Section 4. Section 5 shows our results for using stacking for combining the predictions of the individual classifiers.

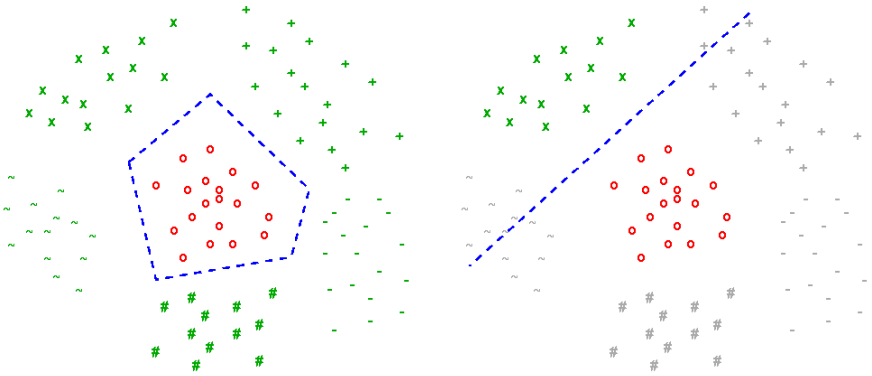

Fig. 1. *One-against-all class binarization* (left) transforms each c-class problem into c binary problems, one for each class, where each of these problems uses the examples of its class as the positive examples (here o), and all other examples as negatives. *Round robin class binarization* (right)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

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 [6, 7]. Separate-and-conquer rule learning algorithms [5] 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.

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 Fig. 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. [12] 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 and Lin [10] obtained better re-

sults using one-against-one technique than one-against-all for SVM with both linear and non-linear kernels.

The main contributions of previous work [6, 7] 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 more information we refer to [7] and [8].

3 Algorithm

The general framework of combining the predictions of multiple classifiers via a separate, trainable classifier is commonly referred to as *stacking* [16]. Its basic idea may be derived as a generalization of voting as follows. Let us consider the voting step as a separate classification problem, whose input is the vector of the responses of the base classifiers. Simple voting uses a predetermined algorithm for this, namely to count the number of predictions for each class in the input and to predict the most frequently predicted class. Stacking replaces this with a trainable classifier. This is possible, since for the training set, we have both the predictions of the base learners and the true class. The matrix containing the predictions of the base learners as predictors and the true class for each training case will be called the *meta-data set*. The classifier trained on this matrix will be called the *meta-classifier* or the classifier at the *meta-level*.

In our case, the round robin procedure learns a number of theories, one for each pair of classes. At the classification time, a new example is tested on each of the binary classifiers. Those, which are trained to distinguish the true class of the new example from some other class, will be called *relevant* predictors. The others, which distinguish a pair of classes, none of which is the correct one for this example, will be called *irrelevant*.

Simple voting relies on the assumption that the relevant classifiers frequently predict the correct class and, hence, give more votes to the true class than any other class can get from the irrelevant ones. If all relevant classifiers predict the correct class, then the overall prediction is correct as well, independently of the predictions of irrelevant classifiers. However, if some of the relevant classifiers give a wrong prediction and several of the irrelevant predictors predict the same wrong class, the overall prediction may be wrong. Occurrence of such an event depends on the behavior of both relevant and irrelevant predictors. Although the training process tries to optimize the behavior of the

Classifier			$original$
c_1/c_2	c_1/c_3	c_2/c_3	$class$
c_1	c_1	c_2	c_1
c_2	c_3	c_2	c_2
c_1	c_3	c_3	c_1
...	...	...	...
c_2	c_3	c_3	c_3

Fig. 2. Combining stacking with pairwise classification: The original problem was a three-class problem with classes (c_1,c_2,c_3). Each training example consists of the predictions of *all* pairwise classifiers c_i/c_j for this example and the original example's class label.

relevant predictors, it provides no guarantee concerning the behavior of the irrelevant ones.

The idea of our approach is as follows. Construct the pairwise classifiers as usual in round robin. When they are finished, each of them is tested on all the training data. This gives us information about the possible misleading votes caused by irrelevant predictors, which is then used to minimize their influence to the classification of new cases. A natural framework for such an approach is the stacking idea described above.

The fact that we have to test each training example on *each* of the binary classifiers, adds a quadratic complexity term (in the number of classes) to the otherwise linear complexity of pairwise classification [7]. However, testing is usually cheap in comparison to training, so we would expect that at least for moderate numbers of classes these additional costs are not noticeable.

Fig. 2 shows the meta-data training set that results from stacking the pairwise classifiers of a hypothetical 3-class problem. Note that the last example of the original training set (which has class c_3) was not part of the training set of the classifier c_1/c_2, but that the meta-data nevertheless contain the response of this classifier.

To see the potential gain of this procedure, consider the third example of Fig. 2. Straight-forward voting would predict class c_3 because the classifier that should discriminate class c_1 from class c_3 erroneously classified this example as c_3, and by chance the classifier c_2/c_3 also considered this example of class c_1 more likely to belong to class c_3. Thus, class c_3 receives 2 votes, whereas the correct class c_1 receives only one vote. However, if cases with this property occur already in the training set, there is a chance that the meta-classifier learns that in such cases, c_1 is actually the more likely prediction. In our experiments, we indeed found situations, when stacking with a nearest neighbor classifier was able to correct some of the mistakes of simple voting. For example, in the letter dataset, we found that in 65% (163 out of 250) cases, for which the true class was not among the classes with the largest number of votes, nearest neighbor classifier still provided a correct classification (see Table 3 in Section 5).

4 Experimental Setup

To evaluate the combination of stacking and pairwise classification, we used the rule learner ripper [3] as the base learning algorithm in a round robin procedure. The round robin learning procedure was implemented in Perl as a wrapper around the base learning algorithm. The wrapper reads in a dataset, provides the binary training and test sets for each pair of the classes to the base learner (by writing them to the disk), and calls the base learner on these sets. The predictions of the learned theories are then used to form the features of the meta dataset for a meta-level classifier. Because ripper is not symmetric in the classes (it produces a rule set for one of the classes and classifies all examples that are not covered by this rule set as belonging to the other class), we performed a double round robin procedure, i.e., we transformed a c-class problem into $c(c-1)$ pairwise problems, where discriminating c_i/c_j and discriminating c_j/c_i are different problems. For details on this we refer to [6].

We implemented the combination of stacking and pairwise classification as discussed in the previous section. At the meta-level, we experimented with three different classifiers:

- the decision-tree learner c4.5 [14]
- ripper in its default settings using ordered binarization to handle the multi-class meta-problem
- a simple nearest neighbor classifier that computes the distance between examples by counting the number of attributes in which they differ. For a given test example, the algorithm predicts the class of the nearest training example. Should there be more than one example at the same minimum distance, the class that appears most frequently in these cases is predicted.

The meta-level problem can become quite large. For example, for the 26-class letter dataset we had to store $26 \times 25 = 650$ predictions for each of the 16,000 training examples, and the nearest neighbor classifier had to compare each of them to each of the 4000 test examples. We handled this quite efficiently by encoding the binary attributes in the form of bit vectors, which also reduced the computation of the similarity measure to counting the number of on-bits in the result of an XOR of two example representations. Nevertheless, these problems may have an unusually high dimension compared to other stacking problems, where the number of classifiers (and hence the number of features at the meta-level) is typically independent of the number of classes in the problem.[3] Because of this, one may expect that the training examples of stacked round robin problems are less dense than in conventional stacking problems, with larger differences for a larger number of classes.

The experiments were performed on the 18 multi-class datasets shown in Table 1. These are the same datasets as used by Fürnkranz [7] except for the largest sets *vowel* and *covertype*. The datasets were chosen arbitrarily among datasets with ≥ 4 classes

[3] An exception is the proposal of Ting and Witten [15], who encode the meta-level using the classifiers' probability estimates for each class. But even there, the encoding is only linear in the number of classes (for a fixed number of classifiers), while we have a quadratic size at the meta level.

Table 1. *Data sets used.* The first column shows the total number of examples in this domain. The second column shows the evaluation methodology (an independent test set for *letter*, 10-fold cross-validation for all others). The next three columns show the number of symbolic and numeric attributes as well as the number of classes. The last column shows the default error, i.e., the error one would get by always predicting the majority class of the dataset.

name	train	test	sym	num	classes	def. error
letter	16,000	4000	0	16	26	95.9
abalone	4177	10-x-val	1	7	29	83.5
car	1728	10-x-val	6	0	4	30.0
glass	214	10-x-val	0	9	7	64.5
image	2310	10-x-val	0	19	7	85.7
lr spectrometer	531	10-x-val	1	101	48	89.6
optical	5620	10-x-val	0	64	10	89.8
page-blocks	5473	10-x-val	0	10	5	10.2
sat	6435	10-x-val	0	36	6	76.2
solar flares (c)	1389	10-x-val	10	0	8	15.7
solar flares (m)	1389	10-x-val	10	0	6	4.9
soybean	683	10-x-val	35	0	19	94.1
thyroid (hyper)	3772	10-x-val	21	6	5	2.7
thyroid (hypo)	3772	10-x-val	21	6	5	7.7
thyroid (repl.)	3772	10-x-val	21	6	4	3.3
vehicle	846	10-x-val	0	18	4	74.2
vowel	990	10-x-val	0	10	11	90.9
yeast	1484	10-x-val	0	8	10	68.8

available at the UCI repository [2]. The implementation of the algorithm was developed independently and not tuned on these datasets. On all datasets but one we performed a 10-fold paired stratified cross-validation using Johann Petrak's Machine Learning Experimentation Environment (MLEE) [4]. The results on the *letter* datasets used the same training/test split that is commonly used in the literature. In addition to the error rates, we also report improvement ratios over round robin learning with unweighted voting. These improvement rates are the ratio of the error rate of the classifier over the error rate of the base-line classifier. For averaging these ratios, we use geometric averages so that performance ratios r and $1/r$ average to 1.

5 Results

The results of the experiments are shown in Table 2. They are very mixed. To some extent, the first line (the letter dataset) illustrates the main finding: Stacking may in some cases lead to a considerable reduction in error, in particular when the nearest-neighbor algorithm is used at the meta-level. In this case, the error can be reduced to about 85% of the error of unweighted pairwise classification, whereas there is no average gain when c4.5 or ripper are used as meta-learners. In fact, their performance

[4] `http://www.ai.univie.ac.at/~johann`

Table 2. Error rates of pairwise classification (using unweighted voting) and three versions of pairwise stacking, using nearest-neighbor, c4.5 and ripper at the meta level (and ripper at the base level). For comparison, we also show previous results (taken from [8]) using weighted voting and the results of unmodified ripper. Shown are the error rates and (where applicable) the performance ratio (i.e., the error rate divided by the error rate of unweighted voting). The averages in the last line exclude the letter dataset, which was evaluated with the designated test set. All other results are from 10-fold cross-validations.

	unweighted	pairwise stacking			weighted	default
dataset	voting	nearest	c4.5	ripper	voting	ripper
letter	7.85	4.23 *0.539*	14.15 *1.803*	12.90 *1.643*	7.90 *1.006*	15.75
abalone	74.34	79.03 *1.063*	76.51 *1.029*	81.25 *1.093*	73.69 *0.991*	81.18
car	2.26	0.64 *0.282*	0.87 *0.385*	0.93 *0.410*	2.14 *0.949*	12.15
glass	25.70	24.77 *0.964*	28.50 *1.109*	28.50 *1.109*	25.70 *1.000*	34.58
image	3.42	2.34 *0.684*	3.42 *1.000*	3.16 *0.924*	3.16 *0.924*	4.29
lr spectrometer	53.11	59.89 *1.128*	52.17 *0.982*	56.69 *1.067*	54.05 *1.018*	61.39
optical	3.74	3.08 *0.824*	7.15 *1.914*	5.93 *1.586*	3.58 *0.957*	9.48
page-blocks	2.76	2.81 *1.020*	2.76 *1.000*	2.96 *1.073*	2.72 *0.987*	3.38
sat	10.35	11.08 *1.071*	11.58 *1.119*	11.84 *1.144*	10.55 *1.020*	13.04
solar flares (c)	15.77	16.05 *1.018*	16.41 *1.041*	15.77 *1.000*	15.69 *0.995*	15.91
solar flares (m)	5.04	5.47 *1.086*	5.18 *1.029*	5.33 *1.057*	4.90 *0.971*	5.47
soybean	5.86	7.03 *1.200*	6.15 *1.050*	6.44 *1.100*	5.86 *1.000*	8.79
thyroid (hyper)	1.11	1.11 *1.000*	1.22 *1.095*	1.17 *1.048*	1.19 *1.071*	1.49
thyroid (hypo)	0.53	0.34 *0.650*	0.48 *0.900*	0.34 *0.650*	0.45 *0.850*	0.56
thyroid (repl.)	1.01	1.03 *1.026*	1.06 *1.053*	1.11 *1.105*	0.98 *0.974*	0.98
vehicle	29.08	28.49 *0.980*	28.96 *0.996*	29.67 *1.020*	28.61 *0.984*	30.38
vowel	19.29	6.46 *0.335*	20.81 *1.079*	21.21 *1.099*	18.18 *0.942*	27.07
yeast	41.78	47.17 *1.129*	43.94 *1.052*	42.32 *1.013*	41.44 *0.992*	42.39
average	17.36	17.46 *0.852*	18.07 *1.012*	18.51 *0.997*	17.23 *0.977*	20.74

(only 3–4 wins on 18 datasets) is significantly worse than unweighted voting at the 5% level according to a sign test.

A more detailed analysis of the performance of stacking with nearest neighbor classifier at the meta level is presented in Table 3. The table is calculated on the letter data set with 26 classes. The test cases are splitted into several groups according to the properties of the set of classes, which received the largest number of votes. Namely, we distinguish the cases according to whether the true class belongs to this set and according to the size of this set. The last two lines of the table show that nearest neighbor remains relatively succesfull even on cases, where the true class received strictly smaller number of votes than some other classes. Simple voting, of course, cannot predict the correct class for any of these cases.

For all meta classifiers, in particular for the nearest-neighbor classifier, there are a few selected datasets, where stacking can lead to a substantial gain (*letter*, *car*, and *vowel* for the nearest-neighbor case). The obvious question at this point is in which cases we can expect a gain from the use of stacking for combining the predictions of pairwise classifiers. This does not seem to depend on simple quantitative parameters

true class among max. votes	no. classes max. votes	n.n. predictions wrong	correct
yes	1	68	3574
yes	2	21	82
yes	3	0	5
no	1	77	159
no	2	10	4

Table 3. Results of nearest neighbor meta-classifier on the test sample (4000 cases) splitted into several subsets of cases according to whether the true class received the maximum number of votes and the number of classes, which received the maximum number of votes.

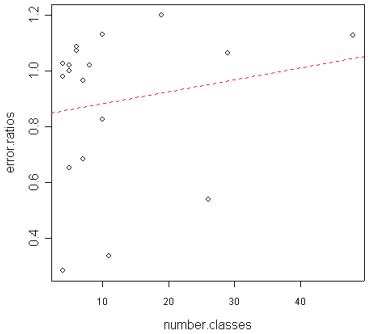

Fig. 3. Error reduction ratio vs. number of classes, with a linear least-squares fit.

of the dataset, like the number of classes in the original problem. There is no correlation ($r^2 = 0.0328$) between the error reduction ratio and the number of classes of the problem (see Fig. 3). We also looked at the correlation of the error reduction ratio with several other data characteristics, including default error, sample size, and the ratio of sample size over the number of classes, with similarly negative results. The performance gain of stacking depends on the structure of the problem and may be tested e.g. by cross-validation.

For comparison, we also show the results of regular ripper (without the use of pairwise classification) and previous results using weighted voting (taken from [8]) where the vote of each binary classifier is weighted with its confidence into its prediction, estimated by the precision of the rule that was used to make the classification. Weighted voting leads to more consistent gains, but never to such an extent as stacking in those cases where it works.

The question when stacking can outperform unweighted voting remains open. However, our experiments demonstrate that it is an alternative that is worth to be tried. For example, in the *letter* dataset, the combination of stacking and pairwise classification has produced the best result that we have seen with ripper as a base learner.

6 Related Work

The basic idea of pairwise classification is fairly well-known from the literature. It can be found in the areas of statistics, neural networks, support vector machines, and others. We refer to [7, Section 8] for a brief survey of the literature on this topic.

Unweighted voting is regularly used for combining the predictions of multiple classifiers, but there is some work on alternative procedures. Hastie and Tibshirani [9] give several examples where votes that depend on class probability estimates may result in considerably different predictions than unweighted voting. Whether this is desirable or not, it can even happen that a class that wins all pairwise comparisons against all other classes, is not predicted because the sum of the probabilities of the predictions are lower

than the weights for all other classes. An empirical evaluation of this issue in a similar setup as in this paper confirmed that weighted voting leads to small but consistent performance improvements over unweighted voting [8].

Allwein et al. [1] treat both one-against-all and pairwise classification (and other class binarization techniques) as special cases of a generalization of error-correcting output codes [4]. They also use a nearest neighbor classifier to combine the predictions of the base learners, however, the distance is measured to prototypes given by a predetermined learning scheme, which does not depend on the training data. In particular, with an appropriate matrix this is equivalent to the simple voting in the one-against-all or pairwise setting. This technique is further analyzed in [11], where experimental evidence showed that unweighted voting performed at least as well (for support-vector machines as the base learner). In [13] another approach for combining binary classifiers is suggested, where separate classifiers are trained for deciding whether a binary classifier is relevant or irrelevant for a given example.

The proposed combination of round robin learning and stacking may also be viewed as analogous to a similar proposal for integrating bagging with stacking [17]. In this sense, this work complements previous investigation of combining round robin learning with other ensemble techniques, namely bagging and boosting [8].

7 Conclusions

In this paper, we evaluated a technique for combining pairwise classification with stacking. The basic idea is to replace the voting procedure that combines the predictions of the binary classification by a trainable classifier. The training set of this meta-classifier consists of all predictions of the binary classifiers for each training example. We evaluated the use of a rule learner, a decision tree learner and a nearest-neighbor learning algorithm. Stacking with the nearest-neighbor classifier at the meta level achieved a substantial improvement in some cases. The performance gain does not seem to depend on the number of classes of the problem and similar characteristics. However, it may be estimated e.g. by crossvalidation in concrete situations. The results show that stacking should be kept in mind as a potential candidate for optimizing the accuracy of the final prediction.

Acknowledgments

We would like to thank Johann Petrak for his experimentation scripts and help using them, and the maintainers of and contributors to the UCI collection of machine learning databases. Petr Savicky was supported by the Ministry of Education of the Czech Republic under Research and Development project LN00A056, Institute of Theoretical Computer Science - ITI. Johannes Fürnkranz 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

[1] 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.

[2] 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.

[3] 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.

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

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

[6] 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.

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

[8] J. Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5), 2003. In press.

[9] 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.

[10] 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.

[11] A. Klautau, N. Jevtić, and A. Orlitsky. On nearest-neighbor ECOC with application to all-pairs multiclass SVM. *Journal of Machine Learning Research*, 4:1–15, 2003.

[12] 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.

[13] 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.

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

[15] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

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

[17] D. H. Wolpert and W. G. Macready. Combining stacking with bagging to improve a learning algorithm. Technical Report SFI-TR-96-03-123, Santa Fe Institute, Santa Fe, New Mexico, 1996.