Combining Pairwise Classifiers with Stacking

Petr Savicky\(^1\) and Johannes Fürnkranz\(^2\)

\(^1\) Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodarenskou Vězi 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@oeaif.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 \(\binom{c}{2}\) 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: