Expert Systems with Applications 38 (2011) 343–359

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Constructing ensembles of classifiers using supervised projection methods based on misclassified instances

Nicolás García-Pedrajas [a,*], César García-Osorio [b]

[a] Department of Computing and Numerical Analysis, University of Córdoba, Campus de Rabanales, 14071 Córdoba, Spain
[b] Department of Civil Engineering, University of Burgos, Escuela Politécnica Superior, Calle Villadiego, s/n 09001 Burgos, Spain

ARTICLE INFO

Keywords:
Classification
Ensembles of classifiers
Boosting
Subspace methods

ABSTRACT

In this paper, we propose an approach for ensemble construction based on the use of supervised projections, both linear and non-linear, to achieve both accuracy and diversity of individual classifiers. The proposed approach uses the philosophy of boosting, putting more effort on difficult instances, but instead of learning the classifier on a biased distribution of the training set, it uses misclassified instances to find a supervised projection that favors their correct classification. We show that supervised projection algorithms can be used for this task. We try several known supervised projections, both linear and non-linear, in order to test their ability in the present framework. Additionally, the method is further improved introducing concepts from oversampling for imbalance datasets. The introduced method counteracts the negative effect of a low number of instances for constructing the supervised projections.

The method is compared with ADABOOST showing an improved performance on a large set of 45 problems from the UCI Machine Learning Repository. Also, the method shows better robustness in presence of noise with respect to ADABOOST.

1. Introduction

An ensemble of classifiers consists of a combination of different classifiers, homogeneous or heterogeneous, to jointly perform a classification task (García-Pedrajas, García-Osorio, & Fyfe, 2007). A classification problem of L classes and n training observations consists of a set of instances whose class membership is known. Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ be a set of n training samples where each instance $\mathbf{x}_i$ belongs to a domain X. Each label is an integer from the set $Y = \{1, \ldots, L\}$. A multiclass classifier is a function f: $X \rightarrow Y$ that maps an instance $\mathbf{x} \in X \subseteq \mathbb{R}^D$ into an element of Y.

The task is to find a definition for the unknown function, $f(\mathbf{x})$, given the set of training instances. In a classifier ensemble framework, we have a set of classifiers $\mathbb{C} = \{C_1, C_2, \ldots, C_m\}$, each classifier performing a mapping of an instance vector $\mathbf{x} \in \mathbb{R}^D$ into the set of labels $Y = \{1, \ldots, L\}$.

Techniques using multiple models usually consist of two independent phases: model generation and model combination (Merz, 1999). Most techniques are focused on obtaining a group of classifiers which are as accurate as possible but which disagree as much as possible. These two objectives are somewhat conflicting, since if the classifiers are more accurate, it is obvious that they must agree more frequently. Many methods have been developed to enforce diversity on the classifiers that form the ensemble (Dietterich, 2000a). Kuncheva (2001) identifies four fundamental approaches: (i) using different combination schemes, (ii) using different classifier models, (iii) using different feature subsets, and (iv) using different training sets. Perhaps the last one is the most commonly used. The algorithms in this last approach can be divided into two groups: algorithms that adaptively change the distribution of the training set based on the performance of the previous classifiers, and algorithms that do not adapt the distribution. Boosting methods are the most representative methods of the first group. The most widely used boosting method is ADABOOST (Freund & Schapire, 1996) and its numerous variants. It is based on adaptively increasing the probability of sampling the instances that are not classified correctly by the previous classifiers.

Bagging (Breiman, 1996a) is the most representative algorithm of the second group. Bagging (after *B*ootstrap *agg*regat*ing*) just generates different bootstrap samples from the training set. Several empirical studies have shown that ADABOOST is able to reduce both bias and variance components of the error (Bauer & Kohavi, 1999; Breiman, 1996b; Schapire, Freund, Bartlett, & Lee, 1998). On the other hand, bagging seems to be more efficient in reducing bias than ADABOOST (Bauer & Kohavi, 1999).

Although these techniques are focused on obtaining as diverse classifiers as possible without deteriorating the accuracy of each classifier, Kuncheva and Whitaker (2003) failed to establish a clear relationship between diversity and ensemble performance.

* Corresponding author. Tel.: +34 957211032; fax: +34 957218630.
E-mail addresses: npedrajas@uco.es (N. García-Pedrajas), cgosorio@ubu.es (C. García-Osorio).
URLs: http://cibrg.org (N. García-Pedrajas), http://cibrg.org (C. García-Osorio).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.06.072

Boosting methods are the most popular techniques for constructing ensembles of classifiers. Their popularity is mainly due to the success of ADABOOST (Dietterich, 2000b). Boosting constructs an ensemble in an stepwise manner. At each step a new classifier is added to the ensemble. The basic idea is that the new classifier is trained on a distribution of the learning instances biased towards the most difficult ones. In this way, each instance has an associated weight that is higher if the instance has been misclassified by several of the previous classifiers. ADABOOST tends to perform very well for some problems but can also perform very poorly on other problems. One of the sources of the bad behavior of ADABOOST is that although it is always able to construct diverse ensembles, in some problems the individual classifiers tend to have large training errors. Moreover, ADABOOST usually performs poorly on noisy problems (Bauer & Kohavi, 1999).

In this work, we present a method based on using the misclassified instances to obtain a supervised projection of the dataset to favor the correct classification of these instances but without putting too much pressure on their correct classification. In the previous work (García-Pedrajas et al., 2007), we developed this model using the hidden layer of a neural network for the supervised projection. Although the method showed a good performance, the use of a neural network to obtain the projection posed a number of problems. Firstly, fine tuning the parameters of the network was not an easy task. In fact, for several problems the bad performance of the method might be due to the bad projection obtained with the network. Also, the saturation of the sigmoid functions of the hidden layer was an issue for many datasets. Secondly, the behavior of the hidden layer is not easy to understand, thus the performance of the method was difficult to explain.

In this paper, we removed this hidden layer and use several other methods for obtaining a supervised projection. In this way, several advantages over the original method can be obtained. Firstly, the parameter fine tuning step is not necessary, as these methods have few parameters and most of them can be chosen from a fairly large interval without damaging the performance of the algorithm. Secondly, these methods offer a more principled way of obtaining the supervised projection and their behavior is fully understood.

The rest of this paper is organized as follows: Section 2 describes our method, Section 3 reviews the supervised projection methods used in the paper, Section 4 shows the experimental setup, Section 5 shows the experimental results, and finally Section 6 states the conclusions of our work.

2. Construction of ensembles of classifiers using supervised projections

One of the sources of failure of boosting is putting too much stress on correctly classifying all the instances. Outliers or noisy instances become too relevant in the training set undermining the performance of the ensemble. In the previous work (García-Pedrajas et al., 2007), we constructed ensembles projecting the input variables in a way that made easier the classification of misclassified instances. This projection was performed using the hidden layer of a multilayer perceptron. In this paper, we show how we can use supervised projections to perform the same task in an easier and faster way.

This approach is able to incorporate the advantages of boosting without its main drawbacks. The construction of the projection taking into account only instances that have been misclassified by a previous classifier permits the new classifier to focus on difficult instances. Nevertheless, as this classifier receives a uniform distribution of the training instances, the sensitivity to noise and the effect of small datasets is greatly reduced. The proposed method at each step t considers only the subset of instances, $S' \subset S$, misclassified by the classifier added in step $t-1$. It uses the instances in S' to obtain either a linear or a non-linear supervised projection that is focused only on misclassified instances. Then, the original training set is projected using this transformation and the next classifier is trained on this projection using an uniform distribution of the instances. The proposed method is shown in Algorithm 1. The next section explains how the supervised projections are obtained.

Algorithm 1. Algorithm for constructing the ensemble of classifiers using linear/non-linear supervised projections.

```
Data: A training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a base learning
    algorithm, $\mathbb{L}$, and the number of iterations $T$.
Result: The final classifier: $C^*(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t:C(\mathbf{x})=y} 1$.
1   $C_0 = \mathbb{L}(S)$
    for $t = 1$ to $T-1$ do
2       $S' \subset S, S' = \{\mathbf{x}_i \in S : C_{t-1}(\mathbf{x}_i) \neq y_i\}$
3       Obtain supervised linear/non-linear projection $\mathbf{P}(\mathbf{x})$
        using $S'$
4       $C_t = \mathbb{L}(\mathbf{P}(S))$
    end
```

2.1. Populating the dataset

One of the problems of our approach appears when the classifier is largely correct and thus the subset of misclassified instances is small. Supervised projection methods suffer from small dataset size, and the performance of the algorithm is damaged. To avoid this problem, we have used a method taken from imbalance dataset classification (Barandela, Sánchez, García, & Rangel, 2003).

This method, called SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), constructs synthetic instances from actual instances. Synthetic samples are generated in the following way: take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. In this way, we can generate new samples that share the main features of actual instances for a more dense dataset.

The procedure for generating this subset of synthetic samples, based on SMOTE procedure, is given in Algorithm 2. With this procedure, we populate the subsets of misclassified instances, obtaining better results than when using only the misclassified instances. In our experiments the parameter N_s was set to 1.

Algorithm 2. Procedure for obtaining synthetic samples.

```
Data: The subset of misclassified instances
    $S_M = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, and the number of synthetic
    instances to generate for each instance $N_s$.
Result: The populated $S_M$ with the instances added.
    for $t = 1$ to $N$ do
5       Obtain nearest neighbor $\mathbf{x}_{nn}$ of $\mathbf{x}_t$
        if $y_{nn} == y_t$ then
            /* Generate $N_s$ synthetic samples */
            for $i = 1$ to $N_s$ do
                for $j = 1$ to $D$ do
6                   $r$ = random value in $(0,1)$
7                   $\mathbf{x}^i_{new} = \mathbf{x}_{t,j} + r\mathbf{x}_{nn,j}$
                end
8               Add $\mathbf{x}_{new}$ to $S_M$
            end
        end
    end
```