Neurocomputing 73 (2010) 3079–3088

Contents lists available at ScienceDirect

Neurocomputing

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

Edited AdaBoost by weighted kNN

Yunlong Gao *, Feng Gao

Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China

ARTICLE INFO

Article history:
Received 2 December 2009
Received in revised form
30 May 2010
Accepted 28 June 2010
Communicated by G.-B. Huang
Available online 25 August 2010

Keywords:
AdaBoost
Overfitting
kNN rules
Adaptability
Feature space

ABSTRACT

Any realistic model of learning from samples must address the issue of noisy data. AdaBoost is known as an effective method for improving the performance of base classifiers both theoretically and empirically. However, previous studies have shown that AdaBoost is prone to overfitting, especially in noisy domains. On the other hand, the kNN rule is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art. In this paper, an edited AdaBoost by weighted kNN (EAdaBoost) is designed where AdaBoost and kNN naturally complement each other. First, AdaBoost is run on the training data to capitalize on some statistical regularity in the data. Then, a weighted kNN algorithm is run on the feature space composed of classifiers produced by AdaBoost to achieve competitive results. AdaBoost is then used to enhance the classification accuracy and avoid overfitting by editing the data sets using the weighted kNN algorithm for improving the quality of training data. Experiments performed on ten different UCI data sets show that the new Boosting algorithm almost always achieves considerably better classification accuracy than AdaBoost. Furthermore, experiments on data with artificially controlled noise indicate that the new Boosting algorithm is robust to noise.

1. Introduction

Despite the fact that learning in the presence of noxious noise is generally quite difficult, the importance of being able to cope with noisy data has led many researchers to study PAC learning in the presence of malicious noise. AdaBoost, the most promising PAC learning algorithm, has been proved theoretically and shown empirically to be an effective method for improving the classification accuracy [1–5]. It was believed initially, that AdaBoost seldom overfits the training data. Even though the training error reaches zero, the AdaBoost algorithm still has a lower test error while training. However, recent studies have suggested that AdaBoost might suffer from the problem of overfitting [6–10], especially for noisy data sets.

The main advantage of AdaBoost over other boosting techniques is that it is adaptive, i.e., it is able to take advantage of weak hypotheses to maximize the minimum margin even if the training error of the combination of hypotheses is zero [10–12]. There are theoretical bounds on the generalization error of linear classifiers [11,13], which decrease as the smallest margin of the samples increases. Hence, the adaptive character of AdaBoost ensures that it creates hypotheses with good generalization. The adaptiveness of AdaBoost, however, is a double-edged sword, in the sense that for noisy data sets, there could be certain data samples that are difficult for the classifier to capture. The boosting algorithm then tends to concentrate its resources on these suspect samples [14], thereby distorting the optimal decision boundary. As a result, the decision boundary will only be suitable for those difficult data samples and not necessarily general enough for other data. The margin maximized by AdaBoost is actually a 'hard margin', that is, the smallest margin of the noisy data samples. Consequently, the margin of the other data points may decrease significantly when we maximize the 'hard margin', thereby forcing the generalized error bound to increase.

Clearly, the property of concentrating AdaBoost resources on a few suspect samples leads to overfitting. The most effective approach for dealing with the overfitting problem in AdaBoost is to eliminate the harmful effects of suspect samples. Several strategies have been proposed to cope with overfitting. The key ideas of these methods can be summarized into two groups: the first attempts to reduce the effects of some hard-to-learn samples, while the other eliminates the effects of hard-to-learn samples. However, to avoid overfitting, two crucial problems should be noted. Which hard-to-learn samples can be dealt with as suspect samples? And, how many hard-to-learn samples can be dealt with as suspect samples?

The rest of the paper is arranged as follows. In Section 2, we discuss related work on other boosting algorithms. In Section 3, we present a weighted kNN algorithm for distinguishing samples into two categories: suspect and non-suspect samples. Then, a full description of EAdaBoost is presented in Section 4; some analyses

* Corresponding author. Tel.: +86 13289368676.
E-mail addresses: ylgao@sei.xjtu.edu.cn, gaoyl05@yahoo.com.cn (Y. Gao).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.06.024

of our proposed algorithm are performed too in Section 4. Finally, Section 5 describes our experiments, while Section 6 concludes the paper.

2. Related work

In most works, estimating the 'hardness' of every training example is based on observations of algorithm behavior. The harmful effects of some hard-to-learn samples are reduced by a modification weighting scheme or by introducing a different cost function. As an example of this, Servedio [15] proposed a smooth boosting algorithm using a modified weighting scheme. This algorithm generates only smooth distributions that do not assign too much weight to any single sample. Ratsch et al. [16] analyzed the dynamic evolution of AdaBoost weights for estimating the 'hardness' of every training sample and regularized the exponential cost function with a penalty term, such as the weight decay method. Based on the concept of robust statistics, Kanamori et al. [17] proposed a transformation of loss functions that makes boosting algorithms robust against extreme outliers.

All the works described above can be characterized as reducing the harmful effects of hard-to-learn samples to avoid overfitting, and implicitly solve the problems of which and how many samples can be dealt with as suspect samples by weight shrinking. Although these weight shrinking methods seem to be very promising, researchers have not reported any significant difference between AdaBoost and the methods of weight shrinking in experiments on noisy data. For example, Domingo and Watanabe [18] proposed a modification of AdaBoost in which the weights of the samples are kept bounded by its initial value. Nevertheless, the authors have reported no significant difference between AdaBoost and this modification in experiments on noisy data. Vezhnevets and Barinova [19] showed that, despite regularization, MadaBoost is also prone to overfitting as the algorithm approaches termination.

A large body of research demonstrates that removing hard samples is worthwhile [20–22]. The main goal of these approaches is to enhance the classification accuracy by eliminating the harmful effects of suspect samples. As the algorithm approaches its predetermined end, it is less and less likely that samples with large negative margins will eventually be correctly labeled. Thus, it is more beneficial for the algorithms to 'give up' on these samples and concentrate their efforts on those samples with small negative margins. Examples of such algorithms are *BrownBoost* and regularized boosting algorithms, which eliminate the harmful effects of suspect samples by 'giving up' on these samples. These methods estimate the degree of 'hardness' of the training sample based on the influence of the sample on the combined hypotheses. However, they determine how many samples can be dealt with as suspect samples based on certain important parameters fixed in advance. For example, to use *BrownBoost* one needs to pre-specify an upper bound $(1/2)-\gamma$ on the error of a weak learner and a 'target' error $\varepsilon>0$ should be given a priori [7]. Regularization methods [23] need to fix a regularization constant in advance.

The problem of handling mislabeled, atypical and noisy training samples has been the focus of much attention in both pattern recognition and machine learning domains. The k-nearest neighbors (kNN) rule is one of the oldest, simplest and nonparametric methods for improving the quality of the training data [24]. However, the performance of kNN depends crucially on the distance metric used to identify nearest neighbors. In fact, as shown by many researchers [25–28], kNN classification can be significantly improved if the input features can capitalize on any statistical regularities in the data. Even a simple (global) linear transformation of input features has been shown to yield much better kNN classifiers.

Since the feature space composed of the obtained classifier in AdaBoost capitalizes on some statistical regularities in the data, in this paper, we first give a weighted kNN algorithm which takes full advantage of the feature space. Then, we propose a new boosting method called Edited AdaBoost by weighted kNN (EAdaBoost). In each iteration of the new method, the weighted kNN algorithm is used to distinguish samples into two categories: suspect and non-suspect samples. A suspicious sample is a sample, for which the optimal prediction for a given loss, and the family of classifiers, differs from its current label, or those observations regarding rare samples, which are often associated with variable asymmetry and are an important cause for leverage points [29]. Samples viewed as being suspicious should have their influence eliminated in the next iteration. In each cycle of the new algorithm, the quality of the training data is improved by 'giving up' on some suspect samples, thus ensuring that the new algorithm enhances classification accuracy and avoids overfitting. In contrast to the works described above which try to cope with overfitting in AdaBoost, the new method explicitly and strictly defines which and how many samples can be dealt with as suspect samples in each cycle. Moreover, none of the important additional parameters need to be given in advance in the new algorithm.

3. Weighted kNN

One of the most widely studied non-parametric classification approaches corresponds to the kNN rule [24,30]. The goal of the kNN algorithm is to form a generalization from a set of labeled training samples such that the classification accuracy for new samples is maximized. The maximum accuracy achievable depends on the quality of the input data and on the appropriateness of the chosen k nearest neighbors. In this section, we give a brief description of the kNN algorithm. After describing the statistical basis of the kNN method, we present a new weighted kNN algorithm, which focuses mainly on how to choose the k nearest neighbors.

3.1. KNN classification rule

For convenience, we fix some terminology. Let $S=(x_i, y_i)$, $i=1,2,\ldots,N$ be the training set, where x_i is a $d-$dimensional vector of attributes and $y_i\in\{+1,-1\}$ is the associated observed class label (for simplicity, we consider a binary classification task). To justify generalization, we make the assumption that the training data are *iid* samples of random variables (X, Y) having some unknown distribution.

Given N previously labeled samples as the training set S, the kNN algorithm constructs a local subregion $R(x)\subseteq\mathbb{R}^d$ of the input space, centered at the estimation point x. The predicting region $R(x)$ contains the k closest training points to x:

$$R(x)=\{\hat{x}|D(x,\hat{x})\le d_{(k)}\} \tag{1}$$

where $d_{(k)}$ is the kth order statistic of $\{D(x,\hat{x})\}_1^N$, and $D(x,\hat{x})$ is a distance metric. Let $k[y]$ denote the number of samples in region $R(x)$, which are labeled y. The kNN algorithm is statistically inspired in the estimation of the posterior probability $P(y|x)$ of the observation point x:

$$p(y|x)=\frac{p(y|x)p(y)}{p(x)}\cong\frac{k[y]}{k} \tag{2}$$

For a given observation x, the decision $g(x)$ is taken by evaluating the values of $k[y]$, and selecting the class that has the