Knowledge-Based Systems 61 (2014) 59–75

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Robust boosting classification models with local sets of probability distributions

Lev V. Utkin *, Yulia A. Zhuk

Department of Control, Automation and System Analysis, St. Petersburg State Forest Technical University, Russia
Department of Information Technology and Systems, St. Petersburg State Forest Technical University, Russia

ARTICLE INFO

Article history:
Received 15 December 2012
Received in revised form 9 February 2014
Accepted 13 February 2014
Available online 24 February 2014

Keywords:
Machine learning
Classification
Boosting
Imprecise model
Robust

ABSTRACT

Robust classification models based on the ensemble methodology are proposed in the paper. The main feature of the models is that the precise vector of weights assigned for examples in the training set at each iteration of boosting is replaced by a local convex set of weight vectors. The minimax strategy is used for building weak classifiers at each iteration. The local sets of weights are constructed by means of imprecise statistical models. The proposed models are called RILBoost (Robust Imprecise Local Boost). Numerical experiments with real data show that the proposed models outperform the standard AdaBoost algorithm for several well-known data sets.

1. Introduction

The classification problem can be formally written as follows. Given n training data (examples, instances, patterns) $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, in which $\mathbf{x}_i \in \mathbb{R}^m$ represents a feature vector involving m features and $y_i \in \{0, 1, \ldots, k-1\}$ indices the class of the associated examples, the task of classification is to construct an accurate classifier $c : \mathbb{R}^m \to \{0, 1, \ldots, k-1\}$ that maximizes the probability that $c(\mathbf{x}) = y_i$ for $i = 1, \ldots, n$. Generally $\mathbf{x}_i$ may belong to an arbitrary set $\mathcal{X}$, but we consider the special case for simplicity $\mathcal{X} = \mathbb{R}^m$. Moreover, we assume that $k = 2$ and $y_i \in \{-1, 1\}$. The main problem of classification is to find a real valued function $f(\mathbf{x}, \mathbf{w}, b)$ having parameters $\mathbf{w}$ and b such that $\mathbf{w} = (w_1, \ldots, w_m) \in \mathbb{R}^m$ and $b \in \mathbb{R}$, for example, $f(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. Here $\langle \mathbf{w}, \mathbf{x} \rangle$ denotes the dot product of two vectors $\mathbf{w}$ and $\mathbf{x}$. The sign of the function determines the class label prediction or $c(\mathbf{x})$.

A classification problem is usually characterized by an unknown probability distribution $p(\mathbf{x}, y)$ on $\mathbb{R}^n \times \{-1, +1\}$ defined by the training set or examples $\mathbf{x}_i$ and their corresponding class labels y_i. Many classification models accept the uniform distribution $p(\mathbf{x}, y)$ which means that every example in the training set has the probability $1/n$. In particular, the empirical risk functional [1] and the well known support vector machine (SVM) method [2–4] exploit this assumption. We will denote the probabilities of examples $h = (h_1, \ldots, h_n)$.

At the same time, many weighted classification models violate this assumption by assigning unequal weights to examples in the training set in accordance with some available additional information. These weights can be regarded as a probability distribution. Of course, the equal weights as well as the arbitrary weights can be viewed as a measure of importance of every example. Many authors adhere to this interpretation of weights. It should be noted however that both the interpretations do not formally contradict one another.

One of the very popular approaches to classification is the ensemble methodology. The basic idea of classifier ensemble learning is to construct multiple classifiers from the original data and then aggregate their predictions when classifying unknown samples. It is carried out by means of weighing several weak or base classifiers and by combining them in order to obtain a classifier that outperforms every one of them. The improvement in performance arising from ensemble combinations is usually the result of a reduction in variance of the classification error [5]. This occurs because the usual effect of ensemble averaging is to reduce the variance of a set of classifiers.

There are many books and review works devoted to the ensemble methodology and the corresponding algorithms due to the popularity of the approach. One of the first books studying how to combine several classifiers together in order to achieve improved

* Corresponding author. Tel.: +7 8126709262.
E-mail addresses: lev.utkin@gmail.com (L.V. Utkin), zhuk_yua@mail.ru (Y.A. Zhuk).

http://dx.doi.org/10.1016/j.knosys.2014.02.007
0950-7051/

recognition performance was written by Kuncheva [6]. The book explains and analyzes different approaches to ensemble methodology comparatively. A comprehensive and exhaustive recent review of the ensemble-based methods and approaches can be found in the work of Rokach [7]. This review provides a detailed analysis of the ensemble-based methodology and various combination rules for combining weak classifiers. Another very interesting and detailed review is given by Ferreira and Figueiredo [8]. The authors of the review cover a wide range of boosting algorithms recently available. This is the only review where the authors attempt to briefly consider and compare a huge number of modifications of boosting algorithms. Re and Valentini [9] proposed a qualitative comparison and analysis of the ensemble methods and their applicability to astronomy and astrophysics.

The most well known ensemble-based technique is boosting. Boosting is a method for improving the performance of weak classifiers which are combined into a single composite strong classifier in order to achieve a higher accuracy. One of the most popular boosting methods is AdaBoost (Adaptive Boosting) proposed by Freund and Schapire [10]. AdaBoost is a general purpose boosting algorithm that can be used in conjunction with many other learning algorithms to improve their performance via an iterative process.

As pointed out by Rokach [7], AdaBoost improves the performance accuracy for two main reasons:

1. It generates a final classifier whose misclassification rate can be reduced by combining many classifiers whose misclassification rate may be high.
2. It produces a combined classifier whose variance is significantly lower than the variances produced by the weak base learners.

AdaBoost has many advantages. One of them is that the algorithm is adaptive, i.e., it is able to take advantage of weak hypotheses that are more accurate than it was assumed a priori. Adaptation is carried out by changing the weights of all misclassified and correctly classified examples in accordance with some rule. In fact, the uniform probability distribution (equal wights assigned to all examples) is replaced by another probability distribution in each iteration of the algorithm in order to improve the performance accuracy in the next iteration. The weights of examples in boosting methods can be regarded as a discrete probability distribution (probability mass function) over a training set. The algorithm searches for a suitable probability distribution starting from the uniform distribution. On the one hand, the adaptation may lead to overfitting in the high-noise regime by assigning too much weights onto a few hard-to-learn examples. This is one of the bottlenecks of AdaBoost and similar ensemble-based classification algorithms.

Many modifications of AdaBoost have been proposed to overcome the problem of overfitting. A very popular technique to prevent overfitting is to stop boosting algorithms after a small number of iterations. There are a lot of papers illustrating the successful early stopping rules, for example, [11–13]. At the same time, Mease and Wyner [11] show by means of simple and visual examples that the early stopping rule may lead to incorrect classification results.

Another interesting approach to overcome overfitting is to restrict the set of possible weights of examples in training data or their probability distributions according to a certain rule. The restriction can be carried out by means of various ways, for example, by introducing an upper bound for the weights, by generating only smooth distributions, etc. (see [14–16]). There exist other approaches to the problem of overfitting, see, for example, [17–21]. They can be viewed as a very small part of many algorithms developed in order to avoid overfitting for noisy data.

A quite different algorithm is proposed in this paper in order to overcome some difficulties of the AdaBoost including the problem of avoiding overfitting. The main idea of the algorithm is to assign to examples not a probability distribution, but a set of probability distributions in each iteration. If we imagine the simplex of all possible weights (probability distributions), then a point within the simplex which is used in the standard AdaBoost in each iteration as a weight vector is transformed to a set of distributions, i.e., we get some polyhedron around the point. So, every iteration has a separate polyhedron constructed in accordance with some rule, namely, by using the so-called imprecise statistical models [22], in particular, an extension of the well-known ε-contaminated (robust) model [23].

Let us consider by means of a simple example how the weights are changed after every iteration of the AdaBoost algorithm. Suppose we have a training set consisting of three examples. This implies that we have probability distribution $h = (h_1, h_2, h_3)$ in each iteration of the boosting such that h belongs to the unit simplex. Starting from the point $(1/3, 1/3, 1/3)$, the probability distribution moves within the unit simplex. This is schematically shown in Fig. 1. Since the probabilities tend to concentrate on "hard" examples [10], we could suppose that one of the possible points to which the probability distribution moves is an extreme (corner) point of the simplex depicted in Fig. 1. Indeed, we assign the weight 1 to a "hard" example and weight 0 to other examples. The idea of the proposed model is to replace distribution points by small simplexes which are schematically depicted in Fig. 2. We will return to the pictures when we consider the formal statement of the classification problem.

It is supposed that every distribution within the polyhedron or the simplex can be used as a weight vector for classification. However, we select a single distribution according to the minimax strategy. In order to solve the classification problem under the set of probability distributions we select the "worst" distribution providing the largest value of the expected risk. It corresponds to the minimax (pessimistic) strategy in decision making and can be interpreted as an insurance against the worst case [24]. The minimax strategy makes the classification problem to be robust.

It should be noted that the proposed algorithm may be efficient when there are available only small training samples. It was pointed out by Hertz et al. [25] that classification on the basis of small training samples is an important problem. The authors of [25] indicated that the successful generalization from a very small number of training data often requires the use of additional available information. As a result, they proposed a boosting algorithm which can learn from very small samples. The authors used 10%

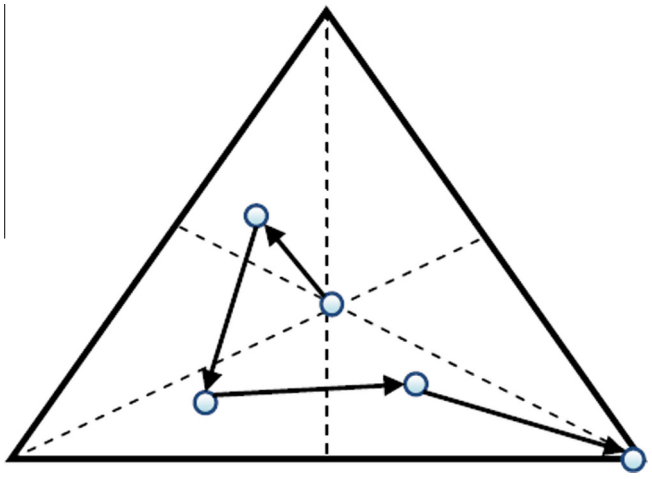

Fig. 1. The unit simplex of probabilities of examples in each iterations for the AdaBoost.