Pattern Recognition 47 (2014) 2080–2090

Contents lists available at ScienceDirect

Pattern Recognition

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

Multi-class boosting with asymmetric binary weak-learners

Antonio Fernández-Baldera, Luis Baumela *

Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus Montegancedo s/n, 28660 Boadilla del Monte, Spain

A R T I C L E I N F O

Article history:
Received 9 May 2012
Received in revised form
11 November 2013
Accepted 23 November 2013
Available online 2 December 2013

Keywords:
AdaBoost
Multi-class classification
Asymmetric binary weak-learners
Class imbalance

A B S T R A C T

We introduce a multi-class generalization of AdaBoost with binary weak-learners. We use a vectorial codification to represent class labels and a multi-class exponential loss function to evaluate classifier responses. This representation produces a set of margin values that provide a range of punishments for failures and rewards for successes. Moreover, the stage-wise optimization of this model introduces an asymmetric boosting procedure whose costs depend on the number of classes separated by each weak-learner. In this way the boosting algorithm takes into account class imbalances when building the ensemble. The experiments performed compare this new approach favorably to AdaBoost.MH, GentleBoost and the SAMME algorithms.

1. Introduction

Boosting algorithms are learning schemes that produce an accurate or *strong classifier* by combining a set of simple base prediction rules or *weak-learners*. Their popularity is based not only on the fact that it is often much easier to devise a simple but inaccurate prediction rule than building a highly accurate classifier, but also because of the successful practical results and good theoretical properties of the algorithm. They have been extensively used for detecting [1–4] and recognizing [5,6] faces, people, objects and actions [7] in images. The boosting approach works in an iterative way. First a weight distribution is defined over the training set. Then, at each iteration, the best weak-learner according to the weight distribution is selected and combined with the previously selected weak-learners to form the strong classifier. Weights are updated to decrease the importance of correctly classified samples, so the algorithm tends to concentrate on the "difficult" examples.

The most well-known boosting algorithm, AdaBoost, was introduced in the context of two-class (binary) classification, but it was soon extended to the multi-class case [8]. Broadly speaking, there are two approaches for extending binary Boosting algorithms to the multi-class case, depending on whether multi-class or binary weak-learners are used. The most straightforward extension substitutes AdaBoost's binary weak-learners by multi-class ones, this is the case of AdaBoost.M1, AdaBoost.M2 [8], J-classes LogitBoost [9], multi-class GentleBoost [10] and SAMME [11]. The second approach transforms the original multi-class problem into a set of binary problems solved using binary weak-learners, each of which separates the set of classes in two groups. Schapire and Singer's AdaBoost.MH algorithm [12] is perhaps the most popular approach of this kind. It creates a set of binary problems for each sample and each possible label, providing a predictor for each class. An alternative approach is to reduce the multi-class problem to multiple binary ones using a codeword to represent each class label [13–15]. When training the weak-learners this binarization process may produce imbalanced data distributions, that are known to affect negatively in the classifier performance [16,17]. None of the binary multi-class boosting algorithms reported in the literature address this issue.

Another aspect of interest in multi-class algorithms is the codification of class labels. Appropriate vectorial encodings usually reduce the complexity of the problem. The encoding introduced in [18] for building a multi-class Support Vector Machine (SVM), was also used in the SAMME [11] and GAMBLE [19] algorithms and is related to other margin-based methods [10]. Schapire uses Error Correcting Output Codes for solving a multi-class problem using multiple binary classifiers [12,13]. Our proposal uses vectorial encodings for representing class labels and classifier responses.

In this paper we introduce a multi-class generalization of AdaBoost that uses ideas present in previous works. We use binary weak-learners to separate groups of classes, like [12,13,15], and a margin-based exponential loss function with a vectorial encoding like [11,18,19]. However, the final result is new. To model the uncertainty in the classification provided by each weak-learner we use different vectorial encodings for representing class labels and classifier responses. This codification yields an asymmetry in the evaluation of classifier performance that produces different

* Corresponding author. Tel.: +34 913367440; fax: +34 913524819.
E-mail address: lbaumela@fi.upm.es (L. Baumela).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2013.11.024

margin values depending on the number of classes separated by each weak-learner. Thus, at each boosting iteration, the sample weight distribution is updated as usually according to the performance of the weak-learner, but also, depending on the number of classes in each group. In this way our boosting approach takes into account both the uncertainty in the classification of a sample in a group of classes and the imbalances in the number of classes separated by the weak-learner [16,17]. The resulting algorithm is called *PIBoost*, which stands for Partially Informative Boosting, reflecting the idea that the boosting process collects partial information about classification provided by each weak-learner.

In the experiments conducted we compare two versions of PIBoost with GentleBoost [9], AdaBoost.MH [12] and SAMME [11] using 15 databases from the UCI repository. These experiments prove that one of PIBoost versions provides a statistically significant improvement in performance when compared with the other algorithms.

The rest of the paper is organized as follows. Section 2 presents the concepts from binary and multi-class boosting that are most related to our proposal. In Section 3 we introduce our multi-class margin expansion, based on which in Section 4 we present the PIBoost algorithm. Experiments with benchmark data are discussed in Section 5. In Section 6 we relate our proposal with others in the literature and in Section 7 we draw conclusions. Finally, we give the proofs of some results in two Appendices.

2. Boosting

In this section we briefly review some background concepts that are directly related to our proposal. Suppose we have a set of N labeled instances $\{(\mathbf{x}_i, l_i)\}, i=1,\dots,N$; where $\mathbf{x}_i$ belongs to a domain X and l_i belongs to $L=\{1,2,\dots,K\}$, the finite label set of the problem (when $K=2$ we simply use $L=\{+1,-1\}$). Henceforth the words *label* and *class* will have the same meaning. $\mathcal{P}(L)$ will denote the power-set of labels, i.e. the set of all possible subsets of L. We will use capital letters, e.g. $T(\mathbf{x})$ or $H(\mathbf{x})$, for denoting weak or strong classifiers that take values on a finite set of values, like L. Small bold letters, e.g. $\mathbf{g}(\mathbf{x})$ or $\mathbf{f}(\mathbf{x})$, will denote classifiers that take value on a set of vectors.

2.1. Binary boosting

The first successful boosting procedure was introduced by Freund and Schapire with their AdaBoost algorithm [8] for the problem of binary classification. It provides a way of combining the performance of many weak classifiers, $G(\mathbf{x}): X \to L$, here $L=\{+1,-1\}$, to produce a powerful "committee" or strong classifier

$$H(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x}),$$

whose prediction is $\text{sign}(H(\mathbf{x}))$.

AdaBoost can also be seen as a stage-wise algorithm fitting an additive model [9,20]. This interpretation provides, at each round m, a *direction* for classification, $G_m(\mathbf{x}) = \pm 1$, and a *step size*, α_m, the former understood as a sign on a line and the latter as a measure of confidence in the predictions of G_m.

Weak-learners G_m and constants α_m are estimated in such a way that they minimize a *loss function* [9,12]

$$\mathcal{L}(l, H(\mathbf{x})) = \exp(-lH(\mathbf{x}))$$

defined on the value of $z=lH(\mathbf{x})$ known as *margin* [10,15].

To achieve this a weight distribution is defined over the whole training set, assigning each training sample $\mathbf{x}_i$ a weight w_i. At each iteration, m, the selected weak-learner is the best classifier according to the weight distribution. This classifier is added to the ensemble multiplied by the goodness parameter α_m. Training data $\mathbf{x}_i$ are re-weighted with $\mathcal{L}(l, \alpha_m G_m(\mathbf{x}))$. So, the weights of samples miss-classified by G_m are multiplied by e^{α_m}, and are thus increased. The weights of correctly classified samples are multiplied by $e^{-\alpha_m}$ and so decreased (see Algorithm 1). In this way, new weak-learners will concentrate on samples located on the frontier between the classes. Other loss functions such as the Logit [9], Squared Hinge [10] or Tangent loss [21] have also been used for deriving alternative boosting algorithms.

Note here that there are only two possible margin values ± 1 and, hence, two possible weight updates $e^{\pm\alpha_m}$ in each iteration. In the next sections, and for multi-class classification problems, we will introduce a vectorial encoding that provides a margin interpretation that has several possible values, and thus, various weight updates.

Algorithm 1. AdaBoost.

1: Initialize the weight Vector $\mathbf{W}$ with uniform distribution $\omega_i = 1/N, i=1,\dots,N$.
2: **for** $m=1$ **to** M **do**
3: Fit a classifier $G_m(\mathbf{x})$ to the training data using weights $\mathbf{W}$.
4: Compute weighted error: $Err_m = \sum_{i=1}^{N} \omega_i I(G_m(\mathbf{x}_i) \neq l_i)$.
5: Compute $\alpha_m = (1/2)\log((1-Err_m)/Err_m)$.
6: Update weights $\omega_i \leftarrow \omega_i \cdot \exp(-\alpha_m l_i G_m(\mathbf{x}_i))$, $i=1,\dots,N$.
7: Re-normalize $\mathbf{W}$.
8: **end for**
9: Output Final Classifier: $\text{sign}(\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x}))$

2.2. Multi-class boosting with vectorial encoding

A successful way to generalize the symmetry of class-label representation in the binary case to the multi-class case is using a set of vector-valued class codes that represent the correspondence between the label set $L=\{1,\dots,K\}$ and a collection of vectors $Y=\{\mathbf{y}_1,\dots,\mathbf{y}_K\}$, where vector $\mathbf{y}_l$ has a value 1 in the l-th co-ordinate and $-1/(K-1)$ elsewhere. So, if $l_i=1$, the code vector representing class 1 is $\mathbf{y}_1=(1,-1/(K-1),\dots,-1/(K-1))^\top$. It is immediate to see the equivalence between classifiers $H(\mathbf{x})$ defined over L and classifiers $\mathbf{f}(\mathbf{x})$ defined over Y:

$$H(\mathbf{x}) = l \in L \Leftrightarrow \mathbf{f}(\mathbf{x}) = \mathbf{y}_l \in Y. \tag{1}$$

This codification was first introduced by Lee et al. [18] for extending the binary SVM to the multi-class case. More recently Zou et al. [10] generalize the concept of binary margin to the multi-class case using a related vectorial codification in which a K-vector $\mathbf{y}$ is said to be a *margin vector* if it satisfies the *sum-to-zero* condition, $\mathbf{y}^\top\mathbf{1}=0$, where $\mathbf{1}$ denotes a vector of ones. This sum-to-zero condition reflects the implicit nature of the response in classification problems in which each y_i takes one and only one value from a set of labels.

The SAMME algorithm generalizes the binary AdaBoost to the multi-class case [11]. It also uses Lee et al.'s vector codification and a multi-class exponential loss that is minimized using a stage-wise additive gradient descent approach. The exponential loss is the same as the original binary exponential loss function and the binary margin, $z=lG(\mathbf{x})$, is replaced by the multi-class vectorial margin, defined with a scalar product $z=\mathbf{y}^\top\mathbf{f}(\mathbf{x})$; i.e.

$$\mathcal{L}(\mathbf{y}, \mathbf{f}(\mathbf{x})) = \exp\left(-\frac{\mathbf{y}^\top\mathbf{f}(\mathbf{x})}{K}\right). \tag{2}$$

Further, it can be proved that the population minimizer of this exponential loss, $\arg\min_{\mathbf{f}(\mathbf{x})} E_{\mathbf{y}|X=\mathbf{x}}[\mathcal{L}(\mathbf{y}, \mathbf{f}(\mathbf{x}))]$, corresponds to the multi-class Bayes optimal classification rule [11]

$$\arg\max_k f_k(\mathbf{x}) = \arg\max_k Prob(Y=y_k|\mathbf{x}).$$