Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 41 (2008) 331–341
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Quadratic boosting

Thang V. Pham[a,*], Arnold W.M. Smeulders[b]

[a] *OncoProteomics Laboratory, Cancer Center Amsterdam, VU University Medical Center, De Boelelaan 1117, 1081 HV Amsterdam, The Netherlands*
[b] *ISLA, Informatics Institute, University of Amsterdam Kruislaan 403, 1098 SJ Amsterdam, The Netherlands*

Received 31 May 2006; received in revised form 1 May 2007; accepted 6 May 2007

Abstract

This paper presents a strategy to improve the AdaBoost algorithm with a quadratic combination of base classifiers. We observe that learning this combination is necessary to get better performance and is possible by constructing an intermediate learner operating on the combined linear and quadratic terms. This is not trivial, as the parameters of the base classifiers are not under direct control, obstructing the application of direct optimization. We propose a new method realizing iterative optimization indirectly. First we train a classifier by randomizing the labels of training examples. Subsequently, the input learner is called repeatedly with a systematic update of the labels of the training examples in each round. We show that the quadratic boosting algorithm converges under the condition that the given base learner minimizes the empirical error. We also give an upper bound on the VC-dimension of the new classifier. Our experimental results on 23 standard problems show that quadratic boosting compares favorably with AdaBoost on large data sets at the cost of training speed. The classification time of the two algorithms, however, is equivalent.

Keywords: AdaBoost; Boosting algorithm; Coordinate descent; Generalization error; Object detection; Quadratic boosting; Randomized relabeling; VC-dimension

1. Introduction

Combining classifiers by boosting algorithms, especially the AdaBoost [1] method, is currently in frequent use in machine learning and pattern recognition [2,3]. These methods convert a weakly performing base learner into one with better performance. One of the main characteristics of boosting algorithms is that they treat the base learner as a black box. They do not require access to the parameters of the base classifiers. This model of learning is flexible since knowledge about the problem can be incorporated in the base learner while it performs with moderate accuracy. Viola and Jones [4] use a space of features designed to ensure fast calculation, while the use of AdaBoost results in a face classifier that is both fast and accurate. Other typical application fields of AdaBoost are image retrieval [5], part-based object classifiers [6], and dynamic Bayesian network classifiers for event detection in video [7].

The AdaBoost algorithm uses a linear combination f of the base classifiers $h(\cdot)$,

$$f(x) = \sum_t \alpha_t h_t(x), \tag{1}$$

where α_t is the weight of the base classifier h_t. The index t denotes the iteration of the algorithm. The algorithm learns classifiers by exploiting the fact that the given learner can be used with different versions of the training set. In each round the AdaBoost algorithm employs a complex scheme of reweighting of training examples.

The limitation of the linear combination in Eq. (1) has not received much attention partly because the choice of the base learner is flexible. Virtually any learning algorithm can be used.

* Corresponding author. Tel.: +31 20 444 6998.
E-mail address: t.pham@vumc.nl (T.V. Pham).

0031-3203/$30.00
doi:10.1016/j.patcog.2007.05.008

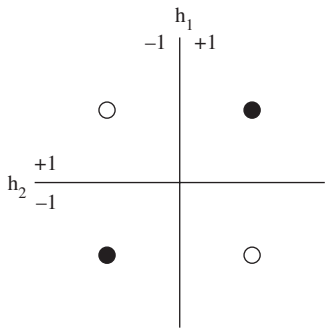

Fig. 1. The exclusive OR problem. A classifier space $H = \{h_1, h_2\}$, where each classifier is represented by a straight line boundary in 2D. The input space X consists of four points in 2D. The filled circles are labeled $+1$ while the empty circles -1. The two classes cannot be separated by any (threshold) linear combination of the two classifiers [8].

This can lead to very complex decision boundaries. However, there are cases where the choice of the base learner is restricted. For example, apart from accuracy the base classifiers may be subject to other design constraints such as classification time or integration with other modules of a larger system.

Under these circumstances the limitation of the linear combination is well studied. A standard example is the exclusive OR problem [8]. Given a space consisting of two classifiers $H = \{h_1, h_2\}$ as in Fig. 1, there is no (threshold) linear combination of h_1 and h_2 that can separate the two classes denoted by the empty and filled circles. Several practical limitations of the linear combination are also given in Ref. [8]. For instance in visual recognition, when each base classifier draws its input from a local region in the image, the combined classifier belongs to the class of diameter-limited perceptrons which cannot recognize global concepts such as connectedness.

In this paper we consider a quadratic combination of two-class classifiers

$$f_q(x) = \sum_i \alpha_i h_i(x) + \sum_{ij} \alpha_{ij} h_i(x) h_j(x). \tag{2}$$

While it seems that this is a simple generalization of the core of AdaBoost in Eq. (1), learning this form of combination is nontrivial. Replacing Eq. (1) of AdaBoost by Eq. (2) in a functional minimization strategy as introduced by Breiman [9] (to be discussed later) would lead to the task of selecting a pair of classifiers together minimizing the empirical error, which cannot be solved by simply invoking the base learner as in linear boosting. An equivalent view of this derivation is to treat the quadratic combination as linear in an extended classifier space. The learner to operate on the combined linear and quadratic terms inherits a fundamental difficulty of a boosting method, namely the base classifiers are not available directly. The realization of standard optimization techniques becomes impossible since we do not have direct access to the parameters of the base classifiers.

We will show that learning in the extended classifier space is possible by at first randomizing the labels of the training examples and subsequently steering the performance of the new learner in an indirect way with a systematically updated set of labels. Thus learning the quadratic combination is done effectively by both reweighting and relabeling of training data.

We provide an analysis of the generalization power of the new algorithm. We also discuss the computational aspects of the new combination in learning and classification. We perform extensive experiments on synthetic data, on standard machine learning data sets, as well as on data sets for the object recognition problem in computer vision. The experimental results show that the new combination outperforms the existing linear boosting in most cases, specifically on large data sets. Significantly, while the new algorithm requires more training time to learn the decision boundary better, it does not require more classification time.

The paper is organized as follows. In the next section we formulate the problem. In Section 3 we provide detail of the AdaBoost algorithm and related theoretical background. Section 4 presents the new algorithm and our analysis of its performance. The experiments are given in Section 5. Finally, Section 6 concludes the paper.

2. Problem statement

Let $\{(x_n, y_n)\}$ be the training set of N labeled examples drawn randomly from an unknown distribution $P(x, y)$, where $x \in X$ denotes a pattern and $y \in Y$ is the class label. We only consider the two-class case, that is $Y = \{-1, +1\}$. A classifier $h \in H$ predicts the class label y of an input pattern x, $y = h(x)$. A learning algorithm L_H is a functional taking the training data as input and returning a classifier, $h = L_H(\{(x_n, y_n)\})$. The aim of the learning algorithm L_H is to find the optimal classifier $h^* \in H$, or an approximation to it, that makes the least number of mistakes in predicting class labels of future patterns. The error rate of a classifier h is denoted by $\varepsilon(h)$.

Given the set of classifiers H and the learner L_H, one can train a combined classifier $\mathscr{F}(x; \{h_i\}, \Gamma)$, where $\{h_i\} \subset H$ is a finite subset of base classifiers, Γ is a set of additional parameters and $\mathscr{F}(x; \cdot)$ is some combination function that may perform better than each of the classifiers learned by L_H.

We consider a combination rule $\mathscr{F}(x; \cdot)$ which is the sign of a quadratic combination of classifiers f_q as in Eq. (2):

$$\mathscr{F}(x; \cdot) = \begin{cases} +1 \text{ if} & f_q(x) > 0, \\ -1 & \text{otherwise.} \end{cases} \tag{3}$$

For Eq. (2), $f_q(x)$ is linear in the space $G = H \cup H^2$, where $H^2 = \{h_i h_j \mid h_i, h_j \in H\}$. As a result, it can be learned by existing methods for linear combination with a base learner in the space G. We choose the AdaBoost algorithm as the learner

for this linear combination. Based on the argument of Breiman [9] presented in the next section, an optimization criterion for the base learner is to minimize the error on the training data. Thus, the first problem we address is to construct from the input learner L_H an intermediate learner L_G aiming at minimizing the empirical error.

The second issue we address is that of the generalization error. By going from the classifier space H to G we effectively increase the degree of freedom of the combined classifier. In case H is finite with C classifiers, the number of free parameters increases from C to $C + C^2$. Thus care must be taken with regard to overtraining the combined classifier. Our analysis is based on the concept of the capacity of learning algorithms and the theory of error bound [1,10,11] as reviewed in the next section.

3. Background

We present the AdaBoost algorithm and related theoretical studies that characterize its performance. The first study treats the AdaBoost algorithm as a minimization procedure, which motivates our choice of optimal criterion for L_G. The second analyzes the generalization capability of the algorithm, which provides a basis for the analysis of the new algorithm.

3.1. The AdaBoost algorithm

Input: Base learner L_H
Training data $\{(x_n, y_n)\}$, $n = 1, \ldots, N$.
Initialize: $w_n^{(1)} = 1/N$
Do for $t = 1, \ldots, T$
(1) Learn a classifier h_t and its weight α_t with the aid of the learner L_H

$$h_t = L_H(\{(x_n, y_n, w_n^{(t)})\}), \tag{4}$$

$$\hat{\varepsilon}_t = \sum_{n, y_n \neq h_t(x_n)} w_n^{(t)}, \tag{5}$$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \hat{\varepsilon}_t}{\hat{\varepsilon}_t}\right), \tag{6}$$

(2) update weights

$$w_n^{(t+1)} = \frac{w_n^{(t)} \, \mathrm{e}^{-\alpha_t y_n h_t(x_n)}}{Z_t}, \tag{7}$$

where Z_t is a normalization factor.
Output:

$$f(x) = \sum_t \alpha_t h_t(x).$$

Box 1. The AdaBoost algorithm.

The AdaBoost algorithm is described in Box 1. (Note that in implementation one has to take care of the case $\hat{\varepsilon}_t$ is zero.) The central aspect of the algorithm is to maintain and to update a weight $w_n^{(t)}$ associated with each example x_n in each iteration t. In each round a classifier h_t is learned from the weighted training set and α_t is computed based on the performance of h_t. In this algorithm T is a pre-defined number of iterations. The algorithm stops when this number is reached, or the empirical error $\hat{\varepsilon}_t$ is 0.5.

3.2. Optimization criterion for a base learner

To construct a base learner from the input learner, we need a design criterion. In Ref. [9], the author shows that under the condition that the base learner returns a classifier with minimal empirical error, the AdaBoost algorithm coincides with the so-called Gauss Southwell method [12] to solve the following minimization problem:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}), \tag{8}$$

where

$$F(\boldsymbol{\alpha}) = \sum_n \mathrm{e}^{-y_n f(x_n)}. \tag{9}$$

$\boldsymbol{\alpha}$ is the vector containing all α_t and f is the linear combination of classifiers in Eq. (1). Thus the learned classifiers are h_t with $\alpha_t \neq 0$.

Optimizing this criterion is reasonable because on average we want $yf(x)$ as large as possible. It is shown in Ref. [13] that $F(\boldsymbol{\alpha})/N$ is an upper bound on the empirical error. This is because for an incorrect prediction we have $y_n f(x_n) \leqslant 0$; hence, $\mathrm{e}^{-y_n f(x_n)} \geqslant 1$. This implies that $F(\boldsymbol{\alpha})$ is at least equal to the total number of incorrect predictions on the training set. Thus minimizing $F(\boldsymbol{\alpha})$ drives down the training error. Friedman et al. [14] provide another argument for minimizing $F(\boldsymbol{\alpha})$. They show that minimizing this criterion is equivalent to a stagewise fitting of the so-called additive logistic regression model.

Specifically, Breiman [9] shows that the AdaBoost algorithm minimizes $F(\boldsymbol{\alpha})$ by first starting with all α_t equal zero. Let f_t be the combined classifier after t rounds. Let $w_n^{(t)}$ and Z_t be defined in each round t of the AdaBoost algorithm as in Box 1 with $Z_0 = 1$. In the following rather than using t as we have so far for indexing classifiers and weights by iteration, now we use i for indexing all classifiers and weights. Let $\alpha_i^{(t)}$ denote the values of α_i in round t. In round t, consider the partial derivative with respect to the coordinate α_i:

$$\left.\frac{\partial F}{\partial \alpha_i}\right|_{\alpha_i = \alpha_i^{(t-1)}} = \sum_n \mathrm{e}^{-y_n f_{t-1}(x_n)}(-y_n)h_i(x_n). \tag{10}$$

By expanding the weight updating rule (Eq. (7)) of the AdaBoost algorithm up to and including round $(t - 1)$,

we have

$$\left.\frac{\partial F}{\partial \alpha_i}\right|_{\alpha_i=\alpha_i^{(t-1)}} = \sum_n \left\{\prod_{j=0}^{t-1} Z_j\right\} w_n^{(t)}(-y_n)h_i(x_n) \tag{11}$$

$$= \left\{\prod_{j=0}^{t-1} Z_j\right\}(2\hat{\varepsilon}_i - 1), \tag{12}$$

where $\hat{\varepsilon}_i$ is the empirical error of h_i. The absolute value of the gradient $|\partial F/\partial\alpha_i|$ at $\alpha_i = \alpha_i^{(t-1)}$ is proportional to $|\hat{\varepsilon}_i - 0.5|$. Thus, in round t the learner L_H helps to select a classifier h_{i*} as far from a random predictor as possible

$$i^* = \arg\max_i |\hat{\varepsilon}_i - 0.5|. \tag{13}$$

The AdaBoost algorithm selects the classifier h_t to be h_{i*}, and α_t to be the optimal increase in $\alpha_{i*}^{(t-1)}$ (again, the subscript t of h_t and α_t indicates boosting iteration):

$$\alpha_t = \alpha_{i*}^{(t)} - \alpha_{i*}^{(t-1)} \tag{14}$$

$$= \arg\min_{\Delta\alpha_{i*}^{(t-1)}} \sum_n \mathrm{e}^{-y_n(f_{t-1}(x_n)+\Delta\alpha_{i*}^{(t-1)}h_{i*}(x_n))}. \tag{15}$$

This optimization problem has an analytic solution by setting the derivative to zero. Canceling out the nonzero constant we have

$$\sum_n \mathrm{e}^{-y_n\Delta\alpha_{i*}^{(t-1)}h_{i*}(x_n)} w_n^{(t)}(-y_n)h_{i*}(x_n) = 0. \tag{16}$$

Note that $y_n h_{i*}(x_n) = 1$ indicates a correct prediction and $y_n h_{i*}(x_n) = -1$ an incorrect prediction. This leads to

$$\mathrm{e}^{\Delta\alpha_{i*}^{(t-1)}}\hat{\varepsilon}_{i*} - \mathrm{e}^{-\Delta\alpha_{i*}^{(t-1)}}(1-\hat{\varepsilon}_{i*}) = 0, \tag{17}$$

which, in turn, leads to Eq. (6) of the AdaBoost algorithm.

In short, when the base learner returns the classifier with minimal empirical error the AdaBoost algorithm can be seen as selecting the coordinate with maximal gradient magnitude to descent in each round. In this view we design the new learner L_G to aim at minimizing the empirical error.

3.3. VC-dimension and bounds on generalization error

Let $\hat{\varepsilon}(h, S)$ denote the empirical error of a classifier $h \in H$ on a training set S. We have seen that the AdaBoost algorithm drives down the empirical error. The goal of a learning algorithm, however, is to minimize the error made on future data, $\varepsilon(h)$. In most cases $\varepsilon(h)$ cannot be computed.

To control the generalization error, Vapnik and Chervonenkis [15] introduced a measure of capacity of H. We will use the notations as in Ref. [16]. Let S be a set of m points in X. A dichotomy of S induced by $h \in H$ is a partition of S into two disjoint subsets S^+ and S^- such that $h(x) = +1$ for $x \in S^+$ and $h(x) = -1$ for $x \in S^-$. Let $\Delta_H(S)$ denote the number of distinct dichotomies of S induced by classifiers of H. Let $\Delta_H(m)$ denote the maximum of $\Delta_H(S)$ over all $S \in X$ of size m. The Vapnik and Chervonenkis (VC) dimension of H, denoted by $VCdim(H)$, is the largest m such that $\Delta_H(m) = 2^m$. Vapnik shows that for any training set S of size N and for any $h \in H$, with probability $1-\eta$ one can assert

$$\varepsilon(h) < \hat{\varepsilon}(h, S) + 2\sqrt{\frac{VCdim(H)(\ln(2N/VCdim(H)) + 1) - \ln(\eta/9)}{N}}. \tag{18}$$

This bound indicates that as we minimize the empirical error the true error will also go down if the second term is sufficiently small. In particular, the smaller the VC-dimension, the smaller is the second term of the bound.

To analyze the generalization error of classifiers learned by the AdaBoost algorithm, Freund and Schapire [1] derive an upper bound on the VC-dimension of this class of classifiers. Let $\Theta_T(H)$ be the class of all classifiers defined by thresholding of a linear combination of T classifiers in H. Clearly, the classifier learned by the AdaBoost algorithm after T iterations belongs to $\Theta_T(H)$. Freund and Schapire show that if $d = VCdim(H) \geqslant 2$, then

$$VCdim(\Theta_T(H)) \leqslant 2(d+1)(T+1)\log_2(\mathrm{e}(T+1)), \tag{19}$$

where e is the base of the natural logarithm.

Another bound, which does not depend on the number of iterations T, is derived by Schapire et al. [11]

$$\varepsilon(h) < P_S[yf(x) \leqslant \theta] + \mathcal{O}\left(\frac{1}{\sqrt{N}}\left(\frac{d\log^2(N/d)}{\theta^2} + \log(1/\eta)\right)^{1/2}\right) \tag{20}$$

for all $\theta > 0$, where P_S denotes the probability over the finite set S and the notation $\mathcal{O}(\cdot)$ means that constant factors are ignored. Again, this bound depends on the VC-dimension of the class of the base classifiers H. If all other terms are equal, the smaller the VC-dimension, the closer the true error is to the empirical *margin error* $P_S[yf(x) \leqslant \theta]$.

It is noted by the authors in Refs. [1,11] that, in general, both bounds (19) and (20) are quite loose. However, it remains of theoretical interest to derive bounds on the generalization error. Furthermore, for a fixed value of the VC-dimension of the hypothesis space ($< \infty$), a large value of N leads to a small value of the second term in both bounds. Hence, the true error rate will follow the empirical error rate closely for large data set.

In conclusion, the VC-dimension of the base classifiers plays a central role in analyzing the generalization error of the classifier learned by the AdaBoost algorithm. We will derive an upper bound on the VC-dimension of the class of classifiers G.

Since $f_q(x)$ is a linear combination of classifiers in G, we can control the generalization error using the above results as they are for a linear combination as well.

4. Quadratic boosting

We will first present a base learner that aims at minimizing the empirical error. We show that under the condition that the given learner L_H minimizes the empirical error, the new algorithm converges to a local minimum. The new algorithm is summarized in Section 4.2. Finally we derive an upper bound for the VC-dimension of the classifier space G.

4.1. Derivation of the intermediate base learner

Our purpose is to design a base learner $g^* = L_G(\{(x_n, y_n, w_n)\})$ returning the best classifier on the training data

$$g^* = \arg\min_{g \in G} \hat{\varepsilon}(g, \{(x_n, y_n, w_n)\}), \tag{21}$$

where $\{(x_n, y_n, w_n)\}$ is a labeled weighted training set and $\hat{\varepsilon}(g, \{(x_n, y_n, w_n)\})$ denotes the empirical error

$$\hat{\varepsilon}(g, \{(x_n, y_n, w_n)\}) = \sum_{n, y_n \neq g(x_n)} w_n.$$

Recall that $G = H \cup H^2$. The case $g \in H$ is dealt with using the original learner L_H. For the case $g \in H^2$ we follow an iterative coordinate descent approach. The method consists of selecting an initial value and repeatedly descending along the two coordinates in turn. When the parameter set of the classifier space is known, one can easily initialize a random value and possibly implement a gradient descent algorithm, rather than coordinate descent. However, in boosting the parameter set is unknown. We will show that both selecting an initial value and performing coordinate descent are possible by calling L_H with different training sets obtained from the original training set by relabeling of examples. Consider the following procedure:

Procedure 1. (1) First, set $k = 0$ and let h_0 be a classifier resulting from randomly relabeling y_n:

$$h_0 = L_H(\{(x_n, y_n r_n, w_n)\}),$$

where r_n is a random value in $\{-1, +1\}$.

(2) Subsequently, learn classifiers by systematically relabeling y_n while incrementing k

$$h_k = L_H(\{(x_n, y_n h_{k-1}(x_n), w_n)\}),$$

$$\hat{\varepsilon}^{(k)} = \hat{\varepsilon}(h_k, \{(x_n, y_n h_{k-1}(x_n), w_n)\}).$$

We will prove the following convergence result of $\hat{\varepsilon}^{(k)}$.

Lemma 1. *Under the condition that L_H always achieves minimal empirical error, $\hat{\varepsilon}^{(k)}$ converges.*

Proof. We have that h_k is learned by L_H. Therefore $h_k \in H$. Since L_H always finds the best $h \in H$ in terms of empirical error and $h_{k-1} \in H$ for $k > 0$, $h_{k+1} = L_H(\{(x_n, y_n h_k(x_n), w_n)\})$ will perform better than or equal to h_{k-1} on the data set it is trained on

$$\hat{\varepsilon}(h_{k+1}, \{(x_n, y_n h_k(x_n), w_n)\})$$

$$\leqslant \hat{\varepsilon}(h_{k-1}, \{(x_n, y_n h_k(x_n), w_n)\}) \tag{22}$$

$$= \sum_{n, y_n h_k(x_n) \neq h_{k-1}(x_n)} w_n \tag{23}$$

$$= \sum_{n, y_n h_{k-1}(x_n) \neq h_k(x_n)} w_n \tag{24}$$

$$= \hat{\varepsilon}(h_k, \{(x_n, y_n h_{k-1}(x_n), w_n)\}). \tag{25}$$

In short, $\hat{\varepsilon}^{(k+1)} \leqslant \hat{\varepsilon}^{(k)}$. In addition, $\hat{\varepsilon}^{(k)} \geqslant 0$. Thus $\hat{\varepsilon}^{(k)}$ converges. □

Assign $g = h_k h_{k+1}$, we have

$$\hat{\varepsilon}(g, \{(x_n, y_n, w_n)\}) = \sum_{n, y_n \neq h_k(x_n) h_{k+1}(x_n)} w_n \tag{26}$$

$$= \sum_{n, y_n h_k(x_n) \neq h_{k+1}(x_n)} w_n \tag{27}$$

$$= \hat{\varepsilon}^{(k+1)}. \tag{28}$$

Thus, Procedure 1 effectively finds a local minimum for Eq. (21) by iterative coordinate descent. To avoid local minima, the practical implementation of the intermediate base learner L_G invokes this procedure several times (with different set of random values $\{r_n\}$) and compares the relative improvement in performance.

Note that the nature of the combination rules as well as the optimization criteria employed by AdaBoost and L_G are different. As a result, while both algorithms use a special type of coordinate descent, the optimization procedure leads to relabeling in case of L_G and to reweighting in case of AdaBoost.

Procedure 1 can be extended to accommodate a polynomial combination of degree d with $d > 2$. Instead of relabeling y_n by multiplying it with the classification result of the previously learned classifier as in Eq. (22), we now multiply it with the product of $d - 1$ previously learned classifiers. A similar convergence result as Lemma 1 can be obtained.

4.2. The algorithm

Input: Base learner L_H, training data $\{(x_n, y_n)\}$, $n = 1, \ldots, N$.
Number of restarts R, number of maximum inner iterations K.

Initialize: $w_n^{(1)} = 1/N$

Do for $t = 1, \ldots, T$

(1) Learn a classifier g_t and its weight α_t
  (a) Learn a single classifier with the aid of L_H, $g^{(0)} = L_H(\{(x_n, y_n, w_n^{(t)})\})$.
  (b) Learn several compound classifiers $g^{(i)} = h_k^{(i)} h_{k-1}^{(i)}, i = 1, \ldots, R$, to avoid local minima
    (i) First, set $k = 0$ and let $h_0^{(i)}$ be a classifier resulting from randomly relabeling y_n.

$$h_0^{(i)} = L_H(\{(x_n, y_n r_n^{(i)}, w_n^{(t)})\}),$$

where $r_n^{(i)}$ are random values in $\{-1, +1\}$.
    (ii) Subsequently, learn classifiers by systematically relabeling y_n while incrementing k until $\hat{\varepsilon}(h_k^{(i)} h_{k-1}^{(i)}, \{(x_n, y_n, w_n^{(t)})\})$ converges [or $k = K$]

$$h_k^{(i)} = L_H(\{(x_n, y_n h_{k-1}^{(i)}(x_n), w_n^{(t)})\}).$$

  (c) Assign the best classifier to g_t and compute α_t

$$g_t = \arg \min_{g^{(i)}, i \geqslant 0} \hat{\varepsilon}(g^{(i)}, \{(x_n, y_n, w_n^{(t)})\}),$$

$$\hat{\varepsilon}_t = \hat{\varepsilon}(g_t, \{(x_n, y_n, w_n^{(t)})\}),$$

$$\alpha_t = \tfrac{1}{2} \ln \left(\frac{1 - \hat{\varepsilon}_t}{\hat{\varepsilon}_t} \right).$$

(2) Update example weights

$$w_n^{(t+1)} = \frac{w_n^{(t)} \mathrm{e}^{-\alpha_t y_n g_t(x_n)}}{Z_t}$$

where Z_t is a normalization factor.

Output: $f_q(x) = \sum_t \alpha_t g_t(x)$

Box 2. The quadratic boosting algorithm.

Box 2 summarizes the quadratic boosting algorithm. In this algorithm R is the number of initial starting points to have a better chance to end up in the global minimum of (21). The maximum number of iterations of the inner most loop is K. It is needed because in practice certain base learners do not achieve minimal empirical error. In this situation the convergence result as derived in Lemma 1 is not guaranteed. The algorithm invokes the base learner $R \times K + 1$ times at most. This provides a worse case estimate for the training time of the algorithm. By increasing the value of R and K one can look for a better solution at the cost of more training time.

The learner L_G fits well with the AdaBoost algorithm. Effectively the AdaBoost algorithm employs the base learner L_H by calling it repeatedly with a new set of weights, while L_G exploits the input base learner L_H by changing the labels of the training set. As a result, the new quadratic boosting algorithm makes more intensive use of the base learner than the original AdaBoost algorithm.

4.3. Generalization error

For the generalization error of the quadratic boosting we prove the following lemma.

Lemma 2. *Let d be the VC-dimension of H. The VC-dimension of G is less than $10d$.*

Proof. This proof follows the approach in Ref. [16] for bounding the VC-dimension of a linear threshold network. Let S be a set of m points in data space X. Recall from the definitions in Section 3.3 that $\Delta_H(m)$ is the maximum number of distinct dichotomies that can be induced by the classifiers in H over all subsets of X of size m. Hence there are at most $\Delta_H(m)$ different dichotomies that can be induced on S by the classifiers in H. Consider the space of classifiers H^2. Because $h(x)h(x)$ is constant for all $h \in H$ and $h_0(x)h_1(x) = h_1(x)h_0(x)$ for all $h_0, h_1 \in H$, the number of distinct dichotomies of S by classifiers in H^2 is at most $\Delta_H(m)(\Delta_H(m) - 1)/2 + 1$. Thus

$$\Delta_G(m) \leqslant \frac{\Delta_H(m)(\Delta_H(m) - 1)}{2} + 1 + \Delta_H(m) \tag{29}$$

$$\leqslant \Delta_H(m)^2. \tag{30}$$

Furthermore, by the result of Sauer [17,18] we have, whenever $VCdim(F) = k < \infty$, $\Delta_F(m) \leqslant (\mathrm{e}m/k)^k$ for all $m \geqslant k$ where e is the base of the natural logarithm. Hence, $\Delta_G(m) \leqslant (\mathrm{e}m/d)^{2d}$ for all $m \geqslant d$.

It is simple to verify that $c = 10$ is the smallest integer satisfying the inequality $(\mathrm{e}c)^2 < 2^c$, for $c \geqslant 1$. For $m = 10d$, we have $\Delta_G(m) \leqslant (\mathrm{e}10)^{2d} < 2^{10d} = 2^m$. Thus for any set S of $m = 10d$ points, $\Delta_G(m) < 2^m$. Therefore, the VC-dimension of G is less than $10d$. □

Theoretically, this result can be used together with the bounds in Section 3.3 to control the generalization error.

5. Experiments

We perform three sets of experiments to compare quadratic boosting with the AdaBoost algorithm. The first experiment is done on synthetic data. Our purpose is to highlight the difference between quadratic boosting and AdaBoost. The next set

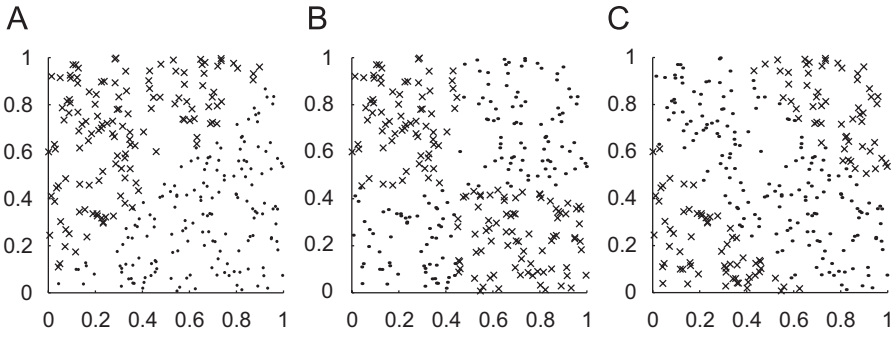

Fig. 2. Three synthetic data sets. (A) A straight boundary between classes. (B) XOR pattern in the boundary. (C) Band classes.

of experiments is carried out on 18 standard machine learning data sets previously used to evaluate the AdaBoost algorithm. Finally, we perform the comparison on two very large data sets for the object classification problem.

5.1. On synthetic data

The following three data sets were generated to visualize the difference between the two methods. The patterns x are uniformly random points in the square $(0, 1) \times (0, 1)$. Each data set has 300 examples. The labels of y are depicted in Fig. 2. The first data set has a straight boundary between the two classes. The second data set has an XOR pattern in the class boundary. Finally, the third data set has a band class separation.

The classifier space H consists of classifiers parallel with one of the two coordinates. Let $x = (u, v)$ be a data point. The classifiers are $\text{sign}(i/10 - u)$, $-\text{sign}(i/10 - u)$, $\text{sign}(i/10 - v)$, and $-\text{sign}(i/10 - v)$ for $i = 0, 1, \ldots, 10$. Overall, H consists of 44 classifiers. The base learner L_H returns the best classifier in terms of empirical error for any set of weights.

Note that the AdaBoost algorithm using this learner has a staircase decision boundary on the first data set [19]. This standard example shows the power of the AdaBoost algorithm to approximate more complex decision boundaries using simpler rules [19].

Fig. 3 shows the result on the first data set. The AdaBoost algorithm produces the staircase approximation as expected. The quadratic boosting algorithm performs only slightly worse than AdaBoost, but still produces a good approximation in spite of the fact that it has more degrees of freedom.

The improvement offered by the new algorithm is clearly seen in Figs. 4 and 5. On the last two data sets AdaBoost performs poorly. In contrast, quadratic boosting approximates well the true class boundary because of a richer classifier space.

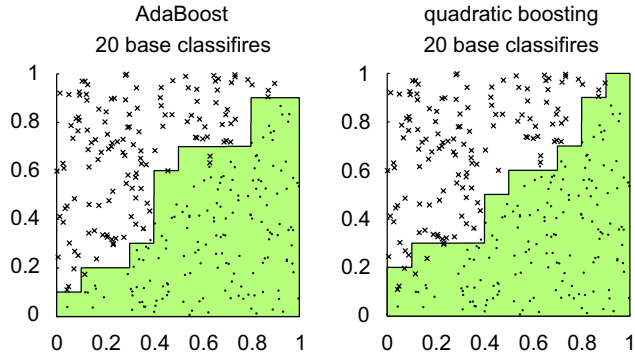

Fig. 3. Results on data set A (300 data points) with single-coordinate base classifiers. A straight edge between two classes is the most favorable case for linear boosting and an extreme case for quadratic boosting. Nevertheless, the performance is more or less similar.

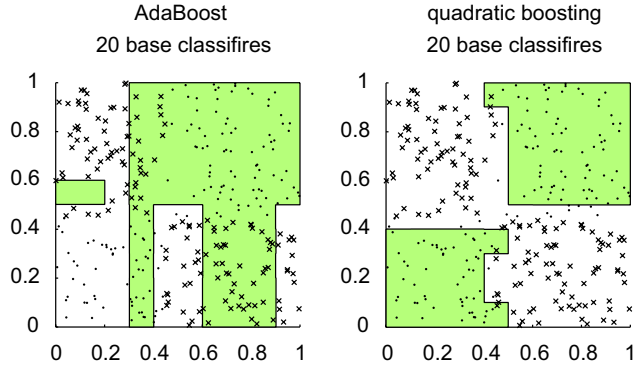

Fig. 4. Results on data set B (300 data points) with single-coordinate base classifiers. As expected, quadratic boosting performs better than AdaBoost in this case.

5.2. On standard machine learning data sets

The next set of experiments is performed on standard machine learning data sets. All data sets are obtained from the UCI machine learning repository [20]. There are 18 data sets in total; all of which are two-class data sets. Table 1 lists the detail of these data sets, ordered by the number of examples. Most of these data sets have been used in previous studies of the AdaBoost algorithm [9,21–23].

We use decision stump [24,25] as the base learner. This learner produces a decision tree with only one level. In particular, we use the implementation available in the Weka machine learning package [25]. We leave the handling of discrete/continuous attributes and missing values to the base learning algorithm.

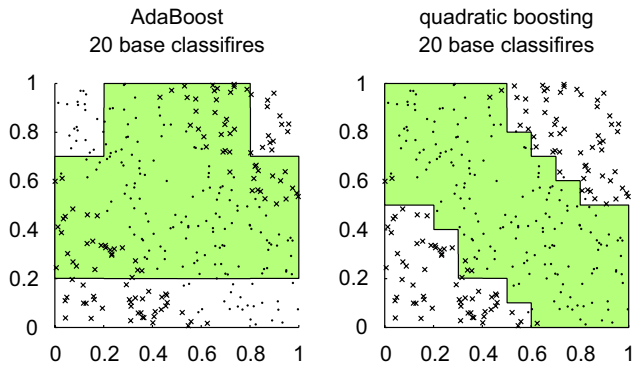

Fig. 5. Results on data set C (300 data points) with single-coordinate base classifiers. Again, quadratic boosting outperforms AdaBoost on this type of class separation.

Table 1
The 18 two-class data sets used in our experiment

#	Data set	# Examples	#Attributes		Missing values
			Disc.	Cont.	
1	labor	57	8	8	x
2	promoters	106	57	–	–
3	hepatitis	155	13	6	x
4	sonar	208	–	60	–
5	cleve	303	7	6	x
6	ionosphere	351	–	34	–
7	house-votes-84	435	16	–	x
8	vote1	435	15	–	x
9	crx	690	9	6	x
10	breast-cancer-w	699	–	9	x
11	pima-indians-di	768	–	8	–
12	german	1000	13	7	–
13	hypothyroid	3163	18	7	x
14	sick-euthyroid	3163	18	7	x
15	kr-vs-kp	3196	36	–	–
16	clean2	6598	2	166	–
17	agaricus-lepiota	8124	22	–	–
18	adult	48 842	12	2	x

For evaluation we use tenfold cross validation [25,26]. Each data set is split into 10 disjoint partitions of (approximately) equal sizes. Both training and evaluation are repeated 10 times for each data set. Each time a different partition is used for testing and the rest (nine partitions) are used for training. The error rates are averaged over 10 runs to obtain the final estimate.

Table 2 gives the test error of AdaBoost and quadratic boosting on the 18 data sets after 250 base classifiers were learned. The quadratic boosting algorithm outperforms AdaBoost on four large data sets (14–17). On data set number 17 "agaricus-lepiota" although both algorithms reach an error rate of 0% quadratic boosting does so with many fewer base classifiers than AdaBoost, 16 in comparison with 74.

Quadratic boosting performs consistently better than the AdaBoost algorithm on training data (data not shown). However, on data sets with less than 1000 samples, overfitting occurs

Table 2
The generalization error (%) using tenfold cross validation after 250 classifiers have been learned

#	Data set	N	Test error (%)		
			Decision stump	AdaBoost	Quadratic boosting
1	labor	57	19.67 ± 14.64	9.00 ± 9.07	7.33 ± 9.04
2	promoters	106	30.18 ± 8.43	11.36 ± 7.20	9.45 ± 7.53
3	hepatitis	155	21.88 ± 3.80	18.04 ± 7.26	17.96 ± 6.60
4	sonar	208	28.40 ± 8.51	14.95 ± 9.02	16.38 ± 5.03
5	cleve	303	26.80 ± 7.50	18.54 ± 5.73	24.49 ± 7.99
6	ionosphere	351	17.39 ± 5.36	7.13 ± 4.10	7.40 ± 3.39
7	house-votes-84	435	4.38 ± 2.42	3.69 ± 1.86	5.06 ± 3.07
8	vote1	435	15.85 ± 3.97	9.65 ± 3.18	11.03 ± 2.66
9	crx	690	14.49 ± 3.67	13.62 ± 3.73	12.03 ± 2.43
10	breast-w	699	8.30 ± 3.78	4.29 ± 2.86	4.43 ± 3.16
11	pima-indians-di	768	27.99 ± 4.22	24.21 ± 4.73	26.17 ± 4.37
12	german	1000	30.00 ± 0.00	25.60 ± 4.34	27.10 ± 2.91
13	hypothyroid	3163	2.62 ± 0.97	0.92 ± 0.52	0.95 ± 0.43
14	sick-euthyroid	3163	5.56 ± 0.65	2.75 ± 0.63	2.34 ± 0.60
15	kr-vs-kp	3196	33.95 ± 1.29	4.26 ± 1.19	1.50 ± 0.46
16	clean2	6598	12.96 ± 1.36	0.75 ± 0.27	0.24 ± 0.15
17	agaricus-lepiota	8124	11.32 ± 0.61	0.00 ± 0.00	0.00 ± 0.00
18	adult	48 842	23.93 ± 0.00	14.85 ± 0.62	14.48 ± 0.62

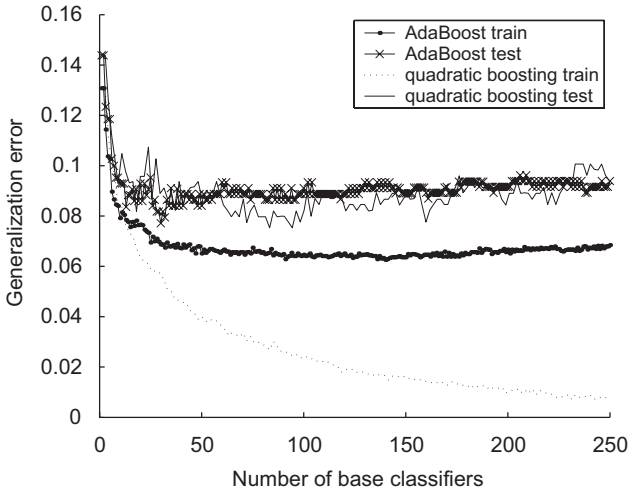

Fig. 6. Performance curve with respect to the number of base classifiers for data set vote1 (435 examples).

more often with the new algorithm than it does with the AdaBoost algorithm (see Fig. 6 for data set "vote1"). This can be explained by the fact that the larger VC-dimension of the new base learner in this case does not guarantee the decrease of the test error as the training error goes down. When the number of samples is large as of data sets from 15 to 18, the test error follows the training error closely (see Fig. 7 for data set "kr-vs-kp"). This can be explained by the fact that the second terms of both error bound Eqs. (18) and (20) are small because a large value of N overwhelms the VC-dimension of the base learner.

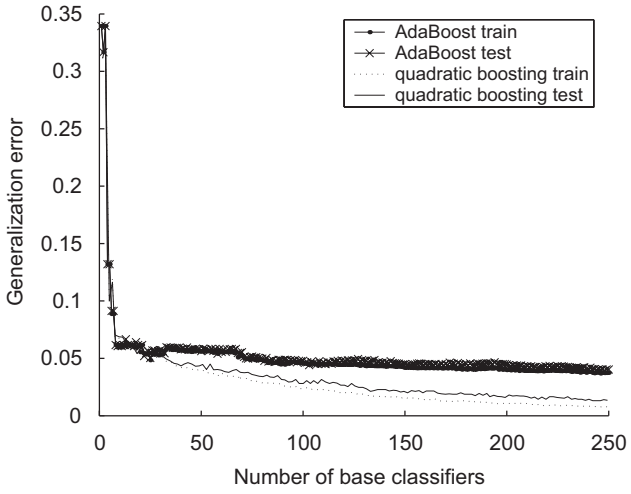

Fig. 7. Performance curve with respect to the number of base classifiers for data set kr-vs-kp (3196 examples).

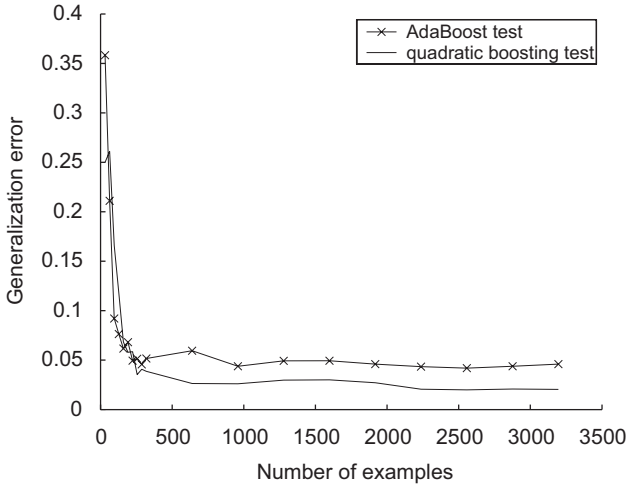

Fig. 8. Performance curve with respect to size of the data set for kr-vs-kp.

Fig. 9. Examples of the MIT pedestrian data set, courtesy of Ref. [27].

The next run of the experiment examines the relative performance of quadratic boosting and AdaBoost with respect to the amount of training data. Fig. 8 shows the performance curve on the data set "kr-vs-kp". For this data set, when the size is less than about 400, both AdaBoost and quadratic boosting do not perform well. The two methods are equivalent. For a larger size the result is stable, and quadratic boosting achieves better result than the AdaBoost algorithm.

Fig. 10. Examples of the MIT car data set, courtesy of Ref. [27].

In this experiment, the training time of quadratic boosting varies from approximately 5–20 times longer than that of the AdaBoost algorithm. Note that we compare the two methods with the same number of base classifiers. Assume the evaluation of each base classifier is equal, the classification time of quadratic boosting is not more than that of the AdaBoost algorithm. In the linear combination (Eq. (1)) one addition and one multiplication are required for each base classifier. Whereas for each compound term (with two classifiers) in the quadratic combination (Eq. (2)) one addition and two multiplications are required.

5.3. On data sets for learning in vision

The next set of experiments is performed on the pedestrian and car data sets for the object detection problem [27]. Figs. 9 and 10 show four examples of each data set, respectively. The task is to distinguish target objects from the background. For each data set in addition to the object examples, we generate 400 000 background patterns by sampling uniformly over a set of images containing no target object. Each data set is split into two equal partitions, one for training and one for testing.

In Ref. [4], the authors use the AdaBoost algorithm with set of features designed to be computed very fast. A two-rectangle feature is the difference between values of two vertically or horizontally adjacent regions, where the value of a region is the sum of all pixel values within that region. A three-rectangle feature is the difference between the sum of the values of the two outside rectangles and the value of the middle one. Finally a four-rectangle feature is the difference between two diagonal pairs. The base learner used is also the single-coordinate learner as described in previous experiments.

Figs. 11 and 12 show the result on the two data sets after 150 classifiers have been learned. The quadratic boosting algorithm performs better than the AdaBoost algorithm on both data sets. With 150 classifiers, the AdaBoost algorithm has an error rate of 0.33% on the pedestrian data set, while the quadratic boosting algorithm achieves 0.17%. On the car data set, the error is 0.31% for the AdaBoost algorithm and 0.15% for the quadratic boosting algorithm.

In both experiments the training time of the quadratic boosting algorithm is about 10 times that of the AdaBoost algorithm. Again, as we have discussed, the runtime of the two methods are equivalent. Note that in both Figs. 11 and 12, 100 base classifiers combined by quadratic boosting can achieve the same

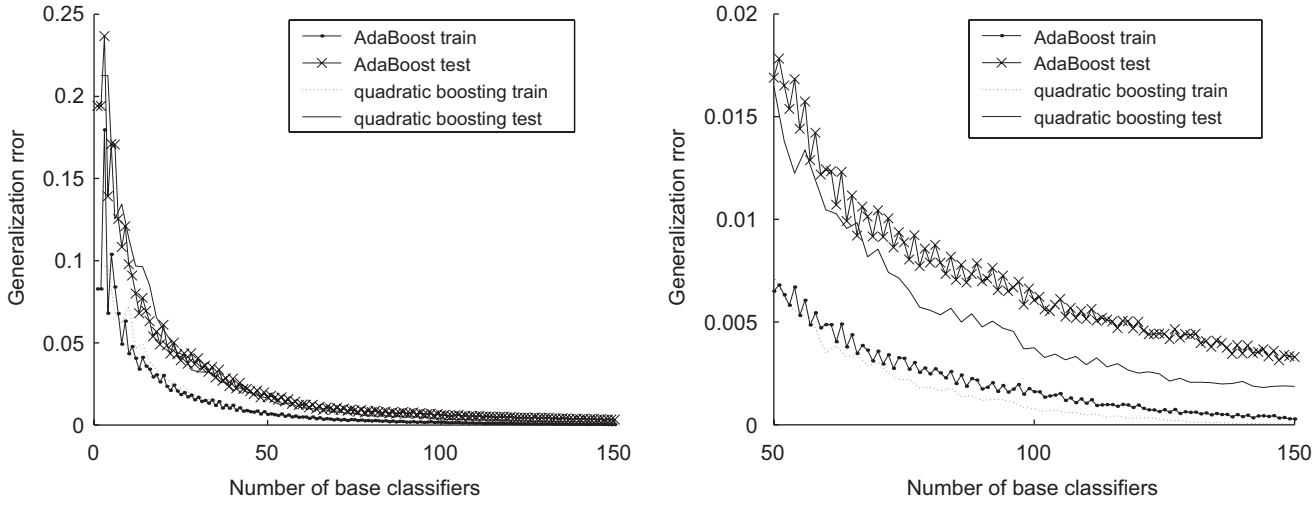

Fig. 11. Result on the MIT pedestrian data set. For better visualization we provide the figure on the right with the number of classifiers ranges from 50 to 150.

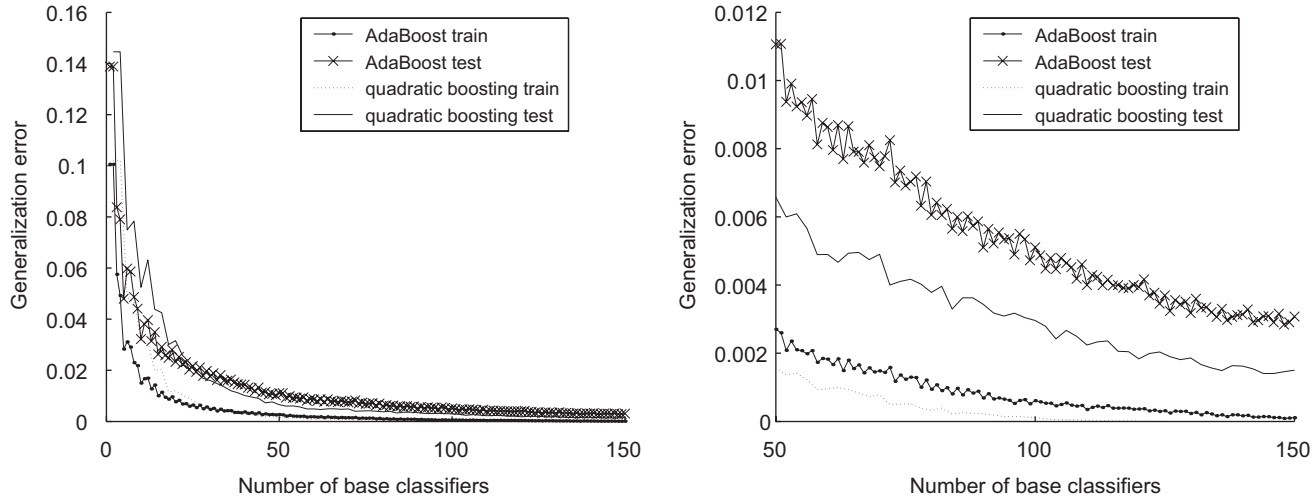

Fig. 12. Result on the MIT car data set. Again, the figure on the right shows the curve with the number of classifiers ranges from 50 to 150.

accuracy as combining 150 base classifiers by linear boosting. Thus at this level of accuracy, substantial gain in classification time is obtained by quadratic boosting.

6. Discussion and conclusion

We addressed the problem of improving the accuracy of learning algorithms with a quadratic combination of base classifiers. This type of combination can be achieved with the construction of an intermediate base learner minimizing the empirical error. This step is difficult because as with other boosting methods, we face the problem that the base classifiers are not available directly. Thus it is not possible to use standard techniques for the minimization problem.

We derive a new algorithm solving the above problem. The method consists of calling the input learner after randomizing the labels of the data set in the first round and subsequently calling it with a systematic update of the labels. This method is in contrast to the AdaBoost algorithm that uses reweighting of training examples. Together they form a powerful combination that makes intensive use the given base learner by both reweighting and relabeling the original training set.

We analyzed the algorithm, showing that under the condition that the input base learner minimizes the empirical error for any set of weights, the new base learner converges to a local minimum. We also study the generalization error of the new algorithm by upper bounding the VC-dimension of the new classifier space.

We performed extensive experiment with the quadratic boosting algorithm. It takes more time to train with the quadratic boosting algorithm than with the AdaBoost algorithm, typically from 5–20 times. This factor is dependent on the internal correlation of the data, but independent of the data size. The difference in classification time of the two algorithms, however, is marginal.

The experiment on synthetic data sets shows the clear improvement of the quadratic combination. In the experiment on standard machine learning data sets, we observed that the training error is almost always smaller for the quadratic boosting algorithm. For small data sets, however, quadratic boosting tends to overfitting more often than the AdaBoost algorithm. Nevertheless, the result on large data sets shows that quadratic boosting outperforms AdaBoost. The experiment on object recognition data sets shows the error of quadratic boosting is approximately half that of the AdaBoost algorithm, which is a significant improvement.

We also performed an experiment with a third order polynomial combination on 18 machine learning data sets (data not shown). As expected, the experimental results show a worse performance than quadratic boosting on small size data sets. On large data sets, we observe a minor improvement in comparison with the result of the quadratic combination.

In conclusion, we have developed a new boosting algorithm with a quadratic combination of base classifiers. The algorithm is particularly suited for large data sets. The algorithm requires more time than the AdaBoost algorithm in training, but not in classification, which provides an attractive choice for practical application.

Acknowledgments

This research is supported by MultimediaN. It was performed while the first author was at the Intelligent Systems Lab Amsterdam (ISLA), University of Amsterdam. The revisions were carried out while the first author was at the OncoProteomics Laboratory, Cancer Center Amsterdam, VU University Medical Center.

References

[1] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[2] X. Huang, S.Li, Y. Wang, Jensen-shannon boosting learning for object recognition, in: Proceedings of CVPR, 2005.
[3] C. Huang, H. Ai, Y. Li, S. Lao, Vector boosting for rotation invariant multi-view face detection, in: Proceedings of ICCV, 2005.
[4] P. Viola, M. Jones, Robust real-time object detection, Int. J. Comput. Vision 57 (2) (2004) 137–154.
[5] K. Tieu, P. Viola, Boosting image retrieval, in: Proceedings of CVPR, vol. I, 2000, pp. 228–235.
[6] H. Schneiderman, T. Kanade, A statistical method for 3d object detection applied to faces and cars, in: Proceedings of CVPR, 2000, pp. 746–751.
[7] V. Pavlovic, A. Garg, J. Rehg, T. Huang, Multimodal speaker detection using error feedback dynamic Bayesian networks, in: Proceedings of CVPR, vol. II, 2000, pp. 34–41.
[8] M. Minsky, S. Papert, Perceptrons, MIT Press, Cambridge, MA, 1969.
[9] L. Breiman, Arcing classifiers (with discussion), Ann. Stat. 26 (3) (1998) 801–849.
[10] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[11] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Stat. 26 (5) (1998) 1651–1686.
[12] D.G. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.
[13] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (3) (1999) 297–336.
[14] J.H. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Stat. 28 (2) (2000) 337–374.
[15] V.N. Vapnik, Estimations of Dependences Based on Statistical Data, Springer, Berlin, 1982.
[16] E.B. Baum, D. Haussler, What size net gives valid generalization?, Neural Comput. 1 (1) (1989) 151–165.
[17] N. Sauer, On the density of families of sets, J. Combin. Theory (Ser. A) 13 (1972) 145–147.
[18] M. Anthony, N. Biggs, Computational Learning Theory, Cambridge University Press, Cambridge, MA, 1992.
[19] T.G. Dietterich, Machine learning research: four current directions, AI Mag. 18 (4) (1997) 97–136.
[20] C. Blake, C. Merz, UCI repository of machine learning databases, Department of Information and Computer Sciences, University of California, Irvine, 1998.
[21] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the 13th International Conference on Machine Learning, 1996, pp. 148–156.
[22] J.R. Quinlan, Bagging, boosting, and c4.5, in: Proceedings 13th American Association for Artificial Intelligence National Conference on Artificial Intelligence, 1996, pp. 725–730.
[23] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting and variants, Mach. Learn. 36 (1/2) (1999) 105–139.
[24] R.C. Holte, Very simple classification rules perform well on most commonly used data sets, Mach. Learn. 11 (1) (1993) 63–90.
[25] I.H. Witten, E. Frank, Data mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, Los Altos, CA, 1999.
[26] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (1) (2000) 4–37.
[27] C. Papageorgiou, T. Poggio, A trainable system for object detection, Int. J. Comput. Vision 38 (1) (2000) 15–33.

About the Author—THANG V. PHAM received the B.Sc. degree in computer science from the RMIT University in 1998 and the Ph.D. degree from the University of Amsterdam in 2005. He was a postdoctoral fellow at the Intelligent Systems Lam Amsterdam until he joined the OncoProteomics Laboratory, Cancer Center Amsterdam, VU University Medical Center in October 2006. He is currently working on pattern recognition techniques for cancer biomarker discovery.

About the Author—ARNOLD W.M. SMEULDERS graduated from Technical University of Delft in physics in 1977 (M.Sc.) and in 1982 from Leyden University in medicine (Ph.D.) on the topic of visual pattern analysis. In 1994, he became full professor in multimedia information analysis at the University of Amsterdam. In the past, he received a Fulbright grant at Yale University visiting professorship at Hong Kong and Tsukuba. Since 2000, he is fellow of International Association of Pattern Recognition and the associated editor of the International Journal for Computer Vision and the IEEE transactions Multimedia. He is on the steering committee of IEEE's International Conference on Multimedia and Expo and was the general chair of ICME2005 in Amsterdam.
Currently, he is scientific director of the Intelligent Systems Lab ISLA at the University of Amsterdam, concentrating on the theory, practice and implementation of multimedia information analysis including picture search engines, machine learning and Internet mood following. The lab has an extensive record in co-operations with Dutch institutions and industry. He is the scientific director of the MultimediaN public–private partnership of 30 institutions and companies. And, he is the scientific director of the ASCI national research school. He leads the Vidi-Video EU-project and is a member of CHORUS and the steering board of the ETP on new and electronic media.