Kim *et al. EURASIP Journal on Image and Video Processing* (2017) 2017:40
DOI 10.1186/s13640-017-0189-y

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Object detection using ensemble of linear classifiers with fuzzy adaptive boosting

Kisang Kim, Hyung-Il Choi and Kyoungsu Oh*

Abstract

The Adaboost (Freund and Schapire, Eur. Conf. Comput. Learn. Theory 23–37, 1995) chooses a good set of weak classifiers in rounds. On each round, it chooses the optimal classifier (optimal feature and its threshold value) by minimizing the weighted error of classification. It also reweights training data so that the next round would focus on data that are difficult to classify. When determining the optimal feature and its threshold value, a process of classification is employed. The involved process of classification usually performs a hard decision (Viola and Jones, Rapid object detection using a boosted cascade of simple features, 2001; Joo et al., Sci. World J 2014: 1–17, 2014; Friedman et al., Ann. Stat 28:337–407, 2000). In this paper, we extend the process of classification to a soft fuzzy decision. We believe this extension could allow some flexibility to the Adaboost algorithm as well as a good performance especially when the size of a training data set is not large enough. The Adaboost algorithm, in general, assigns a same weight to each training datum on the first round of a boosting process (Freund and Schapire, Eur. Conf. Comput. Learn. Theory 23–37, 1995). We propose to assign different initial weights based on some statistical properties of involved features. In experimental results, we show that the proposed method yields higher performances compared to other ones.

Keywords: Adaboost, Fuzzy, Distribution, Classification

1 Introduction

Classifiers are an invaluable tool for many tasks today, such as gesture recognition, medical or genomics predictions, and face recognition [1]. For example, through recognition of users in a real-time interactive system, various personalized services can be provided to each user [2]. Thus, increasing the accuracy and the performance speed of classifying a desired object are very important factors. There appear many classification methods in the literature. Some of them, especially useful for object detection, are the SVM (Support Vector Machine) [3] and the Neural Networks [4]. Recently, boosting methods, originally proposed by Freund and Shapire [5], have gained much attention. They are statistical additive techniques that utilize a combination of various low-quality classifiers to obtain a compound classifier that outperforms any of its components.

The Adaboost usually combines linear classifiers as component classifiers. However, other decision algorithms can be combined to obtain a compound classifier that performs well for some specific applications. For example, the boosted decision trees [6] combines each decision tree successively in an adaptive boosting manner, and Boosted Fuzzy-Rule-Based Classifiers [7]combines each fuzzy rule successively in an adaptive boosting way. There also have been many researches that studied the learning strategy itself of the boosting algorithm [8–10]. The Adaboost may induce overfitting problem when training data include heavy noisy terms, since it explicitly emphasizes data that are difficult to classify [8, 10]. Gunnar [8] suggested a method that achieves a soft margin by introducing slack variables, and Joo [9] suggested the approach that discriminates component classifiers in such a way that the early stage classifier is for high recall rate while the late stage classifier is for high precision rate. Friedman and Tibshirani [10] interpreted boosting as an additive logistic regression problem. They considered boosting as an approximation to additive modeling

*Correspondence: oks@ssu.ac.kr
School of Media, Soongsil University, Seoul, South Korea

on the logistic scale using maximum Bernoulli likelihood as a criterion. They developed LogitBoost algorithm which directly approximates the maximum likelihood and applied it to an ensemble of decision trees.

In this paper, we address the problem of how to determine a linear classifier (a pair of feature and its threshold value) at each round of boosting. We confine our discussion to the ensemble of linear classifiers, and we represent a linear classifier with a feature and its threshold value without loss of generality. The Adaboost algorithm sequentially chooses some set of features that are effective for classifying training data. Initially, each training datum is treated equally important. However, when a next feature is to be determined, they assign a different weight to each training datum, so that some misclassified data in the previous stage would be correctly classified by a next feature. The Adaboost algorithm also determines a threshold value for each chosen feature, so that an input is to be classified belonging to a target class when its feature value is greater than the chosen threshold. The best threshold value is the one that minimizes a misclassification rate. This paper proposes the new method that can improve the accuracy and performance of the Adaboost algorithm by providing the way of assigning initial weights differently for positive training data as well as the way of determining the threshold of each involved feature.

Figure 1 shows the structure of our training method. The initial weights of positive training data are set based on two statistical properties, the average value of each feature and dispersion of each datum from the average. The feature selection process needs the information on the error rate of each feature. The error rate is to be computed with a soft decision. Section 2 describes the basic idea of the ensemble of linear classifiers with adaptive boosting. In Section 3, we explain the process of assigning different initial weights for positive data. Our initial weight is a so called a guide line for selecting features at beginning round of training. It leads to favor features that represent well the object to be classified. Section 4 details how to make a fuzzy decision when selecting features. In Section 5, we present experimental results which show the performance of our approach, comparing against that of other ensembles of linear classifiers with adaptive boosting.

2 Ensemble of linear classifiers

Figure 2 shows the overall structure of ensemble of linear classifiers, in which each weak linear classifier employs some feature to test an input against its threshold value. Each weak classifier returns +1 or -1. When the evaluated feature value becomes greater than a threshold, +1 is returned. Otherwise, -1 is returned. The output of each weak classifier is properly weighed and combined altogether to become a final output of ensemble of linear classifiers. The two main points arise when one builds an ensemble of linear classifiers. One is how to determine an appropriate feature and its threshold value for each weak classifier, and another is how to determine an appropriate weight to be used when combining the output of each weak classifier. The Adaboost resolves these problems with some training algorithm which uses a pool of training data and a pool of candidate features.

The Adaboosing proceeds in rounds. In each round, it determines a weak classifier (feature and its threshold value) as well as the weight for the classifier. The main idea of the Adaboosing is to alter the weights of training data at each round, so that the next weak classifier could cover the training data which are misclassified by previous features. Table 1 summarizes the typical Adaboosing algorithm. Initially, it assigns a same weight to each training data. On each round of Adaboosing, it chooses a feature F_t^* and its threshold T_t^* for which Eq. (1) is minimized. In Eq. (1), $F(i,j)$ is the i-th feature value evaluated on the j-th training datum I_j. The first term of Eq. (1) is the sum of weights of training data that are false negatives and the second term is the sum of weights of training data that are

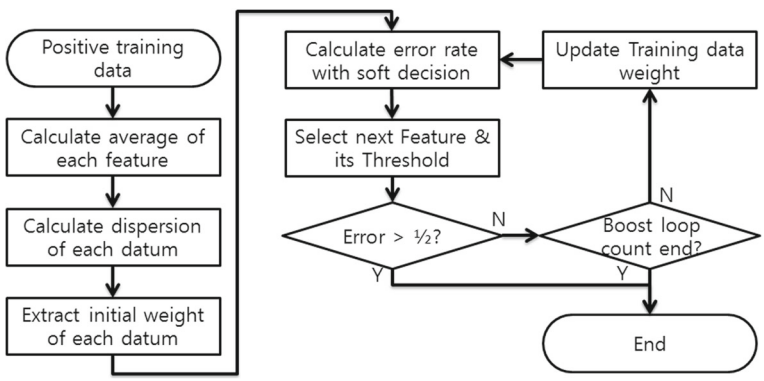

Fig. 1 Overall process of proposed method

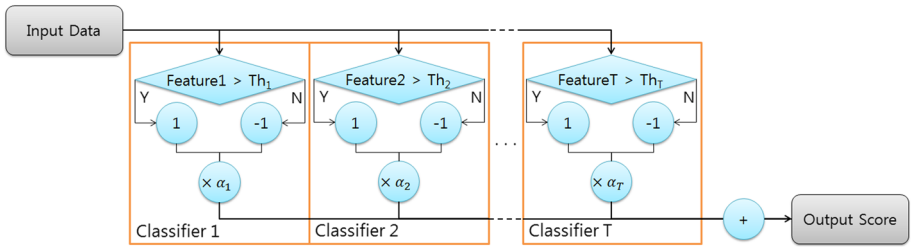

Fig. 2 Ensemble of linear classifiers

Table 1 Adaboosting algorithm

Input : Training Data : $I_1, I_2, \ldots, I_n$

Training Algorithm :

for(j = 1; j ≤ *n*; j++)

$W_1(j) = 1/n$

for(t = 1; t ≤ T; t++) {

Find F_t^* and T_t^* such that ϵ_t is minimized.

$$\epsilon_i = \sum_{\{j|FN(j)\}} W_t(j) + \sum_{\{j|FP(j)\}} W_t(j) \quad (1)$$

where $FN(j) \triangleq I_j$ *is positive data* $\wedge\ T_i > F(i,j)$
and $FP(j) \triangleq I_j$ *is negative data* $\wedge\ T_i \leq F(i,j)$

If $\epsilon_t \geq 1/2$ then stop

Set $\alpha_t = \frac{1}{2} log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

Update

$$W_{t+1}(j) = \begin{cases} \frac{W_t(j)exp(-\alpha_t h_t(I_j))}{Z_t}, & \text{if } I_j \text{ is positive data} \\ \frac{W_t(j)exp(\alpha_t h_t(I_j))}{Z_t}, & \text{if } I_j \text{ is negative data} \end{cases} \quad (2)$$

where Z_t is normalization factor and $h_t(I_j) = \begin{cases} 1, & F_t^*(I_j) > T_t^* \\ -1, & \text{Otherwise} \end{cases}$

}

Classifier :

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \quad (3)$$

false positives. Therefore, it tries to minimize the weighted misclassification rate. In Algorithm 1, $h_t(I_j)$ is the tentative hypothesis (output of weak classifier) made at the t-th round for the j-th training datum I_j. It has a value of 1 when the feature value is greater than the threshold and a value of -1 otherwise.

Once the optimal feature and its threshold are found by minimizing the weighted error, the next step is to update the weights. The weight $W_t(j)$ for each training datum I_j is updated as in Eq. (2). In Eq. (2), the weight adjustment is made with the exponential function where the weighted misclassification rate α_t is used as an exponent and the tentative hypothesis $h_t(I_j)$ determines the sign of the exponent. The misclassified datum gets to have an increased weight, while the correctly classified datum gets to have a decreased weight. The final classification is made by integrating tentative hypotheses $H(x)$ as in Eq. (3).

The Adaboost can be used in many areas of applications. One typical area is an object recognition and detection. When a building the classifier in the area, one can use various types of features without changing the principal strategy of the Adaboost. Some of them are a Haar-like feature [11], LBP (Local Binary Pattern) [12], MCT (Modified Census Transform) [13], Garbor Wavelet [14], and Depth feature [9]. Among them, a Haar-like feature is the mostly used one, since it is easy to compute and gives quite good results especially when detecting some desired object. In this paper, we consider a class of Haar-like features as a potential pool of candidate features. However, our approach can be applied to other types of features mentioned above without a major modification.

3 Methods

3.1 Adjusted initial weights

There have been many reports about setting initial weights in a training process [15]. Many researchers reported that initial weights may play a crucial role [16]. Different initial weights may induce different results when a learning process may not guarantee the convergence to a global optimal solution [17]. Therefore, one often tries to learn necessary parameters of a system with different sets of initial weights and combines results to get a final set

of parameters [18]. When building a system of classifying a target object with cluttered background, one relies on training data with positive and negative samples. In general, negative samples could be quite different from each other and the variance of them can be very large, while positive samples may show somewhat common properties. Therefore, to increase the generality of the system, one should not count on a specific negative sample too much since the negative sample may reappear with little chance in real situation. On the other hand, one may need to treat the positive sample with more attention when the sample shows some common property of positive training data, since a target object may show such a property with a larger chance in real situation.

The typical Adaboost algorithm assigns a same weight to each training datum at the beginning of training [5, 11]. We are to handle this problem under the following observations. First, when a feature is meaningful, positive training data generally respond to it well and show high scores. Second, positive training data have a well concentrated distribution of feature values while negative training data have a scattered distribution of feature values. Figure 3 shows the example of distribution of positive (e.g., face) and negative (e.g., non-face) training data for the case of classifying face images against non-face images. Each training datum is accumulated at the position of its feature values. As can be seen, positive training data are well concentrated compared to negative training data.

Based on the above observations, we assign initial weights differently to positive and negative training data. For negative training data, we assign an equivalent initial weight. For positive training data, we assign a higher initial weight to the datum whose feature values are closer to the peak of the overall distribution as in Fig. 3. We calculate the average value of each feature and also the dispersion of each feature value from the average. Figure 4 shows this process illustratively. In Eq. (4), $F(i,j)$ denotes the i-th feature value of the j-th training datum and $Avg(i)$ denotes the average of the i-th feature value over the total of N positive training data.

$$Avg(i) = \frac{1}{N}\sum_{j=1}^{N} F(i,j) \tag{4}$$

$$Disp(j) = \frac{1}{M}\sum_{i=1}^{M} \big(F(i,j) - Avg(i)\big)^2 \tag{5}$$

In Eq. (5), the average values obtained in Eq. (4) can be used to calculate $Disp$, the degree of dispersion of the feature value at each datum. The total of M feature values can be used to find the dispersion. We set the weights so that a high weight would be assigned to the datum that has a low dispersion and a low weight to the datum with a high dispersion. We also scale the weight properly as in Eq. (6), avoiding the case where the denominator becomes zero. In Eq. (6), $Disp_{min}$ is the smallest value among all the $Disp$ values and $Disp_{max}$ is the largest value.

$$D(j) = \frac{Disp_{max} - Disp_{min}}{1 + \big(Disp(j) - Disp_{min}\big)} \tag{6}$$

Lastly, the initial weight is calculated by normalizing the weight values as in Eq. (7). We assign different initial weights only to positive data while keeping equivalent weights to negative data. To make the total sum of initial weights of positive data 1/2, the Eq. (7) includes 1/2.

$$W(j) = \frac{1}{2} \cdot \frac{D(j)}{\sum_{j=1}^{N} D(j)} \tag{7}$$

3.2 Generating a linear classifier with a fuzzy decision

The Adaboost chooses a good set of weak classifiers in rounds. On each round, it chooses the pair of the optimal feature and its threshold value by minimizing the weighted error of classification [5]. The involved process of classification performs a hard decision in most cases. In

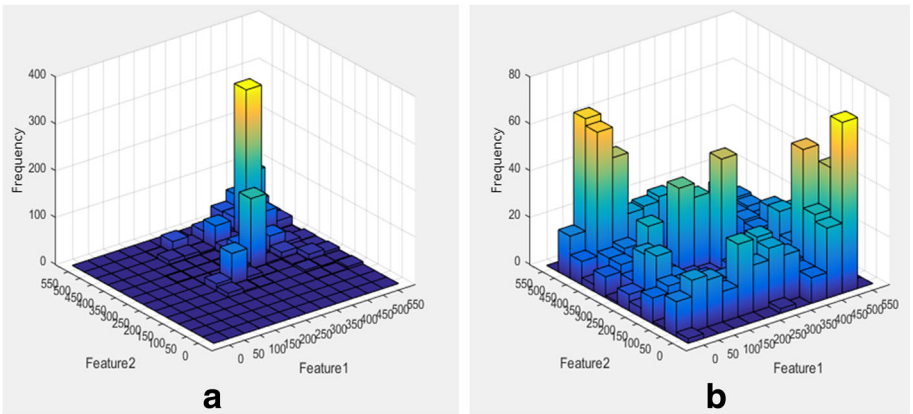

Fig. 3 Data distribution. **a** Positive data (face). **b** Negative data (non-face)

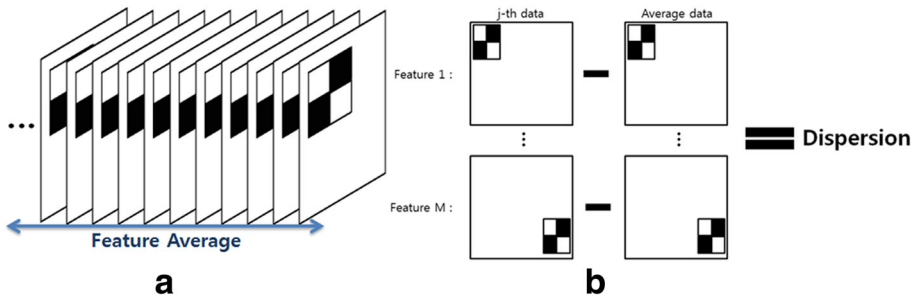

Fig. 4 Feature average (**a**) and data dispersion (**b**)

other words, it decides each training datum to belong to a positive group or a negative group. The weighted misclassification rate then becomes the sum of weights of the data that are from positive training set but decided to belong to a negative group. In this paper, we extend the process of classification to a soft fuzzy decision. We compute a soft membership, with which each training datum belongs to a positive group or a negative group. The misclassification of the potential weak classifier is calculated based on the soft membership values. We believe this extension could allow some flexibility to the Adaboost algorithm as well as an increased generalization ability of the Adaboost algorithm, since the Adaboost algorithm may cause the overfitting problem when the size of training data is not large enough or the training data contain heavy noise terms [8].

As mentioned earlier, the Adaboost employs a hard decision when it constructs a weak classifier at each round. To choose a pair of an optimal feature and its threshold, it evaluates each training datum with a tentative feature and a tentative threshold, and decides the membership of each datum either 0 or 1. When a datum has value of a tentative feature less than a tentative threshold, it declares that the datum belongs to a negative group (non-object group) with a membership of 1. Otherwise, it declares that the datum belongs to a positive group (object group) with a membership of 1. It does not consider an in-between membership. We want to convert this decision process to a soft decision by introducing another variable (eventually two threshold values, Th_1 and Th_2). When a datum has the feature value less than Th_1, it is declared to belong to a negative group with a membership of 1. When a datum has the feature value greater than Th_2, it is declared to belong to a positive group with a membership of 1. But, when a datum has a feature value between Th_1 and Th_2, we assign to the datum an in-between membership. Thus, the procedure of building a weak classifier has to choose the triplet of an optimal feature and its two thresholds Th_1 and Th_2.

The general idea of fuzzy decision is illustrated in Fig. 5. In Fig. 5, a horizontal axis denotes the feature value and a vertical axis denotes membership belonging to a positive group. When a training datum has a feature value greater than Th_2, it is declared to belong to a positive group with possibility 1. When a training datum has a feature value between Th_1 and Th_2, it is declared to belong to a positive group with possibility less than 1 as in Eq. (8). The negative group membership value of each datum can be obtained by subtracting the positive group membership value from 1.

$$pm_i(j) = \begin{cases} 0, & \text{if } F(i,j) < Th_1(i) \\ \frac{F(i,j)-Th_1(i)}{Th_2(i)-Th_1(i)}, & \text{if } Th_1(i) \le F(i,j) \le Th_2(i) \\ 1, & Th_2(i) < F(i,j) \end{cases} \tag{8}$$

On each round of the Fuzzy Adaboost, we choose the triplet of optimum feature F_i^* and its two threshold values Th_1^* and Th_2^* for which Eq. (9) is minimized. The overall misclassification rate of ϵ_i is determined through multiplying the weight of a datum by its opposite group membership and adding them up over all the training data. The first term of Eq. (9) is the sum of products of the weight and negative membership of positive training data, and the second term is the sum of products of the weight and positive membership of negative training data.

$$\epsilon_i = \sum_{j:pos\ data} W_t(j)\cdot(1-pm_i(j)) + \sum_{j:neg\ data} W_t(j)\cdot pm_i(j) \tag{9}$$

$$Th_2(i) = Th_1(i) + \frac{1}{N}\sum_{j=1}^{N}\sqrt{\left(Avg(i) - F(i,j)\right)^2} \tag{10}$$

In each round of our Fuzzy Adaboost algorithm, it has to search for two threshold values as well as a feature for which the Eq. (9) is minimized. This task requires heavy computational efforts. To reduce the computational complexity, we put some constraint between Th_1 and Th_2 as in Eq. (10). The Eq. (10) tells that Th_2 is set to be

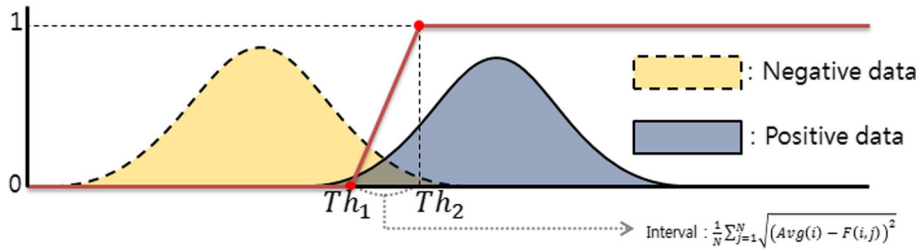

Fig. 5 Illustration of fuzzy membership

apart from Th_1 by the amount of dispersion of the feature value. This constraint seems to be reasonable since a large dispersion implies somewhat mixed class memberships of data in general. With the constraint of Eq. (10), our Fuzzy Adaboost algorithm needs to search for the optimum feature and Th_1 in each round. It could reduce the computation time very much.

$$h_t(I_j) = \begin{cases} 1, & \text{if } F_t^*(I_j) > Th_2^* \\ -1 \cdot (1 - pm^*(j)) + 1 \cdot pm^*(j), & \text{if } Th_1^* \le F_t^*(I_j) \le Th_2^* \\ -1, & \text{if } F_t^*(I_j) < Th_1^* \end{cases} \tag{11}$$

Once the optimal feature and its thresholds are found by minimizing Eq. (9), the weight of each training datum is updated for the next round. For this step, our Fuzzy Adaboost determines the output of each weak linear classifier $h_t(I_j)$ as in Eq. (11). The process of updating weights of training data is carried up using the Eq. (2), and the overall classification result of the ensemble of linear classifiers is obtained as in Eq. (3).

4 Results and discussion

For experimental evaluations, we have used a computer with Intel(R) Core^{TM} i7-4790 3.60 GHz CPU and 8 Gb memory. To implement the suggested algorithms, we have used Windows 8 OS, Visual Studio 2010 and OpenCV 2.4.9 Library. We evaluated the algorithms with the task of classifying a target object against background. As for target objects, we considered two different cases; one is a face object and the other is a pedestrian object. This is to test our algorithm with several databases. We used the face dataset of the New York University [19], The Chinese university of Hong Kong [20], and part of the FEI face database [21] as well as the pedestrian dataset MIT [22]. We collected arbitrary outdoor and indoor images from an internet for negative training data. As for training data, we used 400 positive facial images, 200 positive pedestrian images, and 800 negative images. As for test data, we used different data from training set. We tested with 1600 positive facial images, 723 positive pedestrian images, and 3200 negative images.

As for the type of features, we have used the Haar-like feature set [23]. The size of images in databases is normalized to be 25 × 25 for face images and 20 × 40 for pedestrian images. Background images are set to have the same size as object images. They are set to be 25 × 25 for the task of classifying face images and 20 × 40 for the task of classifying pedestrian images. Figure 6 shows 20 Haar-like features that were selected from a set of 100,000 features during the training with our fuzzy Adaboost algorithm. They include most facial components like eyes, ears, and mouth.

Figure 7 shows the histogram of the sum of weighted hypotheses, $\sum_{t=1}^{T} \alpha_t h_t(x)$, in Eq. (3) for face and non-face images. The values for face images are depicted on the right side with blue color and the values of background images are depicted on the left side with orange color. The histogram has properly quantized bins of the weighted sum. 20 features are used for this evaluation. The horizontal axis denotes bins and the vertical axis denotes the number of test data whose weighted sum falls on each bin. increases. As can be seen in the figure, our algorithm discriminates two different classes more clearly, having less amount of mixed data around the value of zero. Our algorithm has the average value of 9.4 for face images and -10 for background images, while Viola-Jones [11] has the average value of 7.2 for face images and -8.3 for background images.

Figure 8 depicts the distribution of feature values for positive (face image) and negative (non-face image) training data, where F_1^* feature is evaluated. F_1^* feature is the one chosen by our Fuzzy Adaboost algorithm at the first round of learning. The dashed red line depicts the distribution of 400 positive data and dotted red line depicts the distribution of 800 negative data. Our fuzzy decision determined two thresholds Th_1^* and Th_2^* at 826 and 1956, respectively. For this experiment, we did not normalize feature values, so they may have large values in magnitude. We can notice that substantial amount of positive and negative data fall in the range between Th_1^* and Th_2^*. We assign to those data soft memberships and use the membership values when computing misclassification rate.

Fig. 6 Selected features

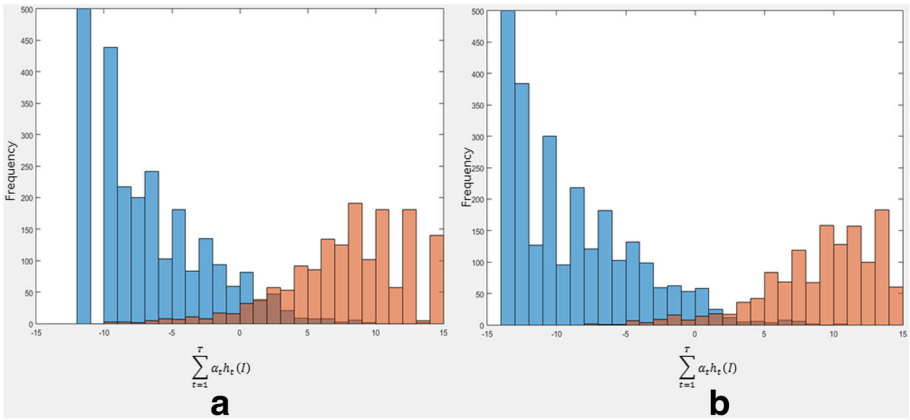

Fig. 7 Histogram of weighted sum of hypotheses in Eq. (3). **a** Viola-Jones [11]. **b** Suggested method

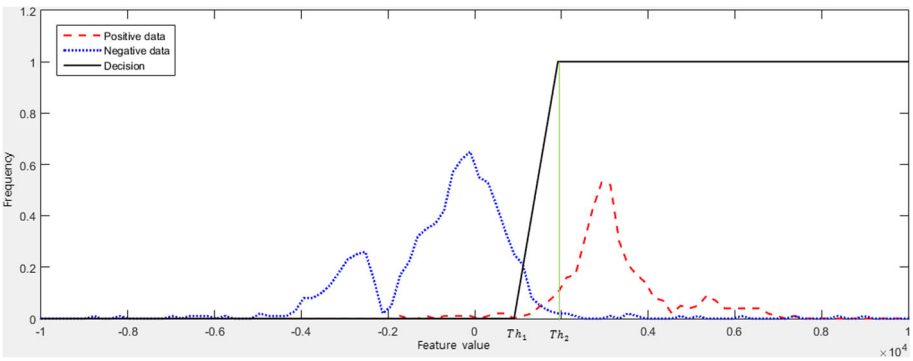

Fig. 8 Data distribution and thresholds along feature value F_1^*

Tables 2 and 3 compare the performance of our method against that of [9, 11] and [10] in terms of FPR (false positive rate), FNR (false negative rate) and TER (total error rate) along the number of included weak classifiers (used features). We implemented adaptive boosting methods of [9, 11], and [10], and constructed an ensemble of linear classifiers for each of them to compare them against ours under the same platform. All of [9, 11], and [10] carry out a hard decision at included weak classifiers, while ours does a soft fuzzy decision at included weak classifiers.

For the case of [11], FPR is well balanced with FNR. It is because [11] determines the threshold of involved linear classifiers in a way that weighs both types of misclassifications equally important. Results of [9] show a lower FNR than FPR. It is because [9] sets the threshold in favor of positive data at the early rounds of boosting. Results of [10] show that the performance seems to get better very slowly but solidly as the number of included weak classifiers increases. It is because, at each round, [10] updates weights of training data reflecting outputs of all the previous rounds. We can notice that our method outperforms the other types of Adaboost algorithm [9, 11] and [10], though the effect gets smaller as the number of included weak classifiers increases. The effect of adjusting initial weights is noticeable especially when the number of included weak classifiers is small. The effect of fuzzy decision is also strong for the small number of features weak classifiers, and the effect lasts over the large range of the number of features weak classifiers.

Table 2 Comparision of face classification rate

Number of weak classifiers	5	10	15	20	25	30
Viola-Jones [11]						
TER	0.25	0.20	0.15	0.12	0.10	0.07
FPR	0.14	0.12	0.09	0.07	0.06	0.04
FNR	0.11	0.08	0.06	0.05	0.04	0.03
Joo [9]						
TER	0.23	0.18	0.14	0.12	0.10	0.06
FPR	0.16	0.13	0.10	0.08	0.07	0.04
FNR	0.07	0.05	0.04	0.04	0.03	0.02
LogitBoost [10]						
TER	0.28	0.22	0.15	0.13	0.10	0.06
FPR	0.18	0.15	0.09	0.08	0.06	0.04
FNR	0.10	0.07	0.07	0.05	0.04	0.02
Adjusted initial weight only						
TER	0.21	0.16	0.13	0.12	0.10	0.07
FPR	0.12	0.09	0.07	0.07	0.06	0.04
FNR	0.09	0.07	0.06	0.05	0.04	0.03
Fuzzy decision only						
TER	0.23	0.18	0.14	0.11	0.08	0.06
FPR	0.12	0.10	0.08	0.06	0.04	0.03
FNR	0.11	0.08	0.06	0.05	0.04	0.03
Adjusted initial weight and fuzzy decision						
TER	0.17	0.12	0.10	0.07	0.06	0.05
FPR	0.10	0.07	0.06	0.04	0.03	0.03
FNR	0.07	0.05	0.04	0.03	0.03	0.02

Table 3 Comparision of pedestrian classification rate

Number of weak classifiers	5	10	15	20	25	30
Viola-Jones [11]						
TER	0.31	0.15	0.11	0.09	0.08	0.07
FPR	0.16	0.08	0.06	0.05	0.05	0.04
FNR	0.15	0.07	0.05	0.04	0.03	0.03
Joo [9]						
TER	0.26	0.14	0.11	0.09	0.07	0.06
FPR	0.20	0.09	0.07	0.06	0.05	0.04
FNR	0.06	0.05	0.04	0.03	0.02	0.02
LogitBoost [10]						
TER	0.36	0.18	0.10	0.09	0.07	0.06
FPR	0.21	0.10	0.06	0.05	0.04	0.03
FNR	0.15	0.08	0.04	0.04	0.03	0.03
Adjusted initial weight only						
TER	0.17	0.11	0.09	0.07	0.06	0.05
FPR	0.10	0.06	0.05	0.04	0.04	0.03
FNR	0.07	0.05	0.04	0.03	0.02	0.02
Fuzzy decision only						
TER	0.25	0.13	0.11	0.08	0.06	0.05
FPR	0.13	0.07	0.06	0.04	0.03	0.03
FNR	0.12	0.06	0.05	0.04	0.03	0.02
Adjusted initial weight and fuzzy decision						
TER	0.14	0.09	0.07	0.05	0.04	0.03
FPR	0.07	0.05	0.04	0.03	0.02	0.02
FNR	0.07	0.04	0.03	0.02	0.02	0.01

To examine the effects of adjusting initial weights in detail, we performed the experiments as in Table 4. We built two ensembles of linear classifiers (10 linear classifiers and 30 linear classifiers) and tested them with different settings of initial weights. The table shows that different initial weights make different results. As mentioned earlier, we can see the strong effect of adjusting initial weights especially when the number of included weak classifiers is small. Computing the initial weights takes

Table 4 Comparison of correct classification rates with differential initial weights

Types of initial weights	With 10 weak classifiers	With 30 weak classifiers
Same weights	80.8%	92.6%
Random weights 1	72.5%	91.2%
Random weights 2	78.1%	91.7%
Random weights 3	81.1%	92.7%
Proposed weights	82.4%	93.3%

about 4.2 s under the current experimental environment. It is relatively small amount of time compared to overall learning time of 8 min under the current experimental setting.

Figure 9 shows the correct classification rates of various methods with 30 weak classifiers when varying the size of a training data set. We randomly selected the specified number of training data from our data sets. We used a different test data set which has 1600 positive data and 3200 negative data as before. The table shows again the importance of initial weights when the size of a training data set is small. When the size of a training data set is too small (100 positives and 200 negatives), our fuzzy soft decision seems to show a contrary effect. It seems to be that the small number of training data does not yield a good membership function (Th_1 and Th_2). As the size of training data increases, the difference among various methods seems to get reduced.

Figure 10 shows ROC (receiver operating characteristics) curves of various classifiers. The classifiers used the total number of 30 features weak classifiers. The curves depict the correlation between a false positive rate and a correct classification rate. The "Fuzzy Decision" curve and "Initial Weight" curve cross over at the false positive rate

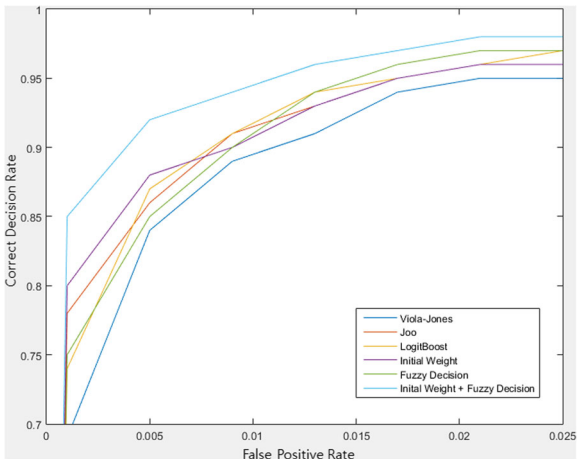

Fig. 10 ROC curves

of 0.013, while the "Fuzzy Decision + Initial Weight" curve show the better correct classification rate over the entire range of false positive rates.

Figure 11 shows some testing data which were misclassified by the Adaboost classifier [9, 11] and [10] though correctly classified by our classifier. All classifiers used 10 features. We can notice that Fig. 11 includes images that do not have exact front view or do have heavily cluttered background like glasses or other persons. Our classifier seems to tolerate such variations relatively better.

5 Conclusions

We have suggested a new method that can improve the accuracy and performance of the Adaboost algorithm by providing the way of assigning initial weights for training data as well as the way of determining the threshold of each involved feature. Positive training data tend to

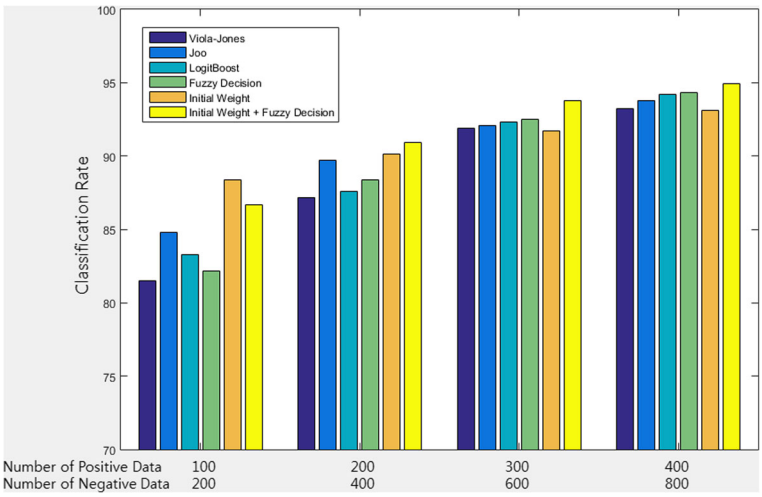

Fig. 9 Correct classification rate along the size of training data set

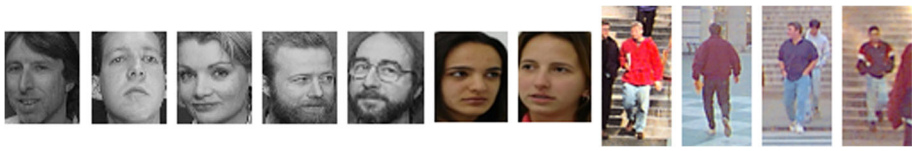

Fig. 11 Examples of false negative

have a well concentrated distribution of feature values while negative training data have a scattered distribution of feature values. Based on such property, we assign initial weights differently to positive and negative training data. For negative training data, we assign an equivalent initial weight. For positive training data, we assign a higher initial weight to the datum which gives the feature value closer to the peak of the overall distribution of total positive training data.

At each round of adaptive boosting, we define a soft membership with which each training datum belongs to a positive group or a negative group. The misclassification of the potential weak classifier is calculated based on the soft membership values. The soft membership is determined with two threshold values which can be learned from training data. To improve the computational efficiency, we define the constraint that relates the second threshold to the first threshold based on the dispersion of feature values of training data. Experimental results seem to confirm that the proposed fuzzy decision could increase the generalizing ability of the classifier. We expect that the suggested soft decision could extend the frame of adaptive boosting into a more flexible one.

Abbreviations

FNR: False negative rate; FPR: False positive rate; LBP: Local binary pattern; MCT: Modified census transform; ROC: Receiver operating characteristics; SVM: Support vector machine; TER: Total error rate

Acknowledgements

This work (Grants No. C0342479) was supported by Business for Academic-industrial Cooperative establishments funded Korea Small and Medium Business Administration in 2015.

Funding

We do not have any funding for this work.

Authors' contributions

KM contributed to the conception or design of the work. HC contributed to the data collection and critical revision of the article. KO contributed to the data analysis and interpretation. KK contributed to the drafting the article. KO gave final approval of the version to be published. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 24 November 2016 Accepted: 4 June 2017
Published online: 20 June 2017

References

1. RK Rout, A survey on object detection and tracking algorithms. PhD thesis (2013). NIT Rourkela
2. DK Prasad, Survey of the problem of object detection in real images. Int. J. Image Process. (IJIP). **6**, 441–466 (2012)
3. S Tong, D Koller, Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. **2**, 45–66 (2001)
4. HB Demuth, MH Beale, O De Jess, MT Hagan, *Neural network design*. (Martin Hagan, 2014)
5. Y Freund, RE Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting. Eur. Conf. Comput. Learn. Theory. **55**, 23–37 (1995)
6. J Ye, J-H Chow, J Chen, Z Zheng, in *Proceedings of the 18th ACM conference on Information and knowledge management*. Stochastic gradient boosted distributed decision trees, (ACM, 2009), pp. 2061–2064
7. MJ Del Jesus, F Hoffmann, LJ Navascués, L Sánchez, Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. IEEE Trans. Fuzzy Syst. **12**, 296–308 (2004)
8. G Rätsch, T Onoda, K-R Müller, Soft margins for adaboost. Mach. Learn. **42**, 287–320 (2001)
9. S-I Joo, S-H Weon, H-I Choi, Real-time depth-based hand detection and tracking. Sci. World J. **2014**, 1–17 (2014)
10. J Friedman, T Hastie, R Tibshirani, et al., Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Stat. **28**, 337–407 (2000)
11. P Viola, M Jones, in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference On*. Rapid object detection using a boosted cascade of simple features, vol. 1 (IEEE, 2001), pp. I:511–518
12. Y Zilu, F Xieyan, in *Signal Processing, 2008. ICSP 2008 9th International Conference on*. Combining lbp and adaboost for facial expression recognition (IEEE, 2008), pp. 1461–1464
13. B Froba, A Ernst, in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*. Face detection with the modified census transform (IEEE, 2004), pp. 91–96
14. M Zhou, H Wei, S Maybank, in *Applications of Computer Vision 2006 workshop in Conjunction with ECCV 2006*. Gabor wavelets and AdaBoost in feature selection for face verification, (2006), pp. 101–109
15. S Lomax, S Vadera, A survey of cost-sensitive decision tree induction algorithms. ACM Comput. Surv. (CSUR). **45**, 1–16 (2013)
16. GE Hinton, S Osindero, Y-W Teh, A fast learning algorithm for deep belief nets. Neural Comput. **18**, 1527–1554 (2006)
17. H Li, C Shen, in *Computing: Techniques and Applications, 2008. DICTA'08, Digital Image*. Boosting the minimum margin: LPBoost vs. AdaBoost (IEEE, 2008), pp. 533–539
18. A Graves, A-R Mohamed, G Hinton, in *Acoustics, speech and signal processing (icassp), 2013 IEEE International Conference on*. Speech recognition with deep recurrent neural networks (IEEE, 2013), pp. 6645–6649
19. The Newyork University, Facial Image Database. http://www.cs.nyu.edu/~roweis/data.html. Accessed 15 June 2017
20. The Chinese University of Hong Kong, Face Database. http://mmlab.ie.cuhk.edu.hk/archive/facesketch.html. Accessed 15 June 2017
21. The FEI, Face Database. http://fei.edu.br/~cet/facedatabase.html. Accessed 15 June 2017
22. The MIT, Pedestrian Database. http://cbcl.mit.edu/software-datasets/PedestrianData.html. Accessed 15 June 2017
23. FC Crow, Summed-area tables for texture mapping. ACM SIGGRAPH Comput. Graph. **18**, 207–212 (1984)