# The AdaBoost Algorithm with Linear Modification of the Weights

Robert Burduk(✉)

Department of Systems and Computer Networks, Wroclaw University of Science and Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
robert.burduk@pwr.edu.pl

**Abstract.** This paper presents a new extension of the AdaBoost algorithm. This extension concerns the weights used in this algorithm. In our approach the original weights are modified, we propose a linear modification of the weights. In our study we use the boosting by the reweighting method where each weak classifier is based on the linear classifier. The described algorithm was tested on Pima data set. The obtained results are compared with the original the AdaBoost algorithm.

**Keywords:** AdaBoost algorithm · Ensemble of classifiers · Linear classifier

## 1 Introduction

Boosting is a machine learning effective method of producing a very accurate classification rule by combining a weak classifier [7]. The weak classifier is defined to be a classifier which is only slightly correlated with the true classification i.e. it can classify the object better than a random classifier. In boosting, the weak classifier is learned from various training examples sampled from the original learning set. The sampling procedure is based on the weight of each example. In each iteration, the weights of examples are changing. The final decision of the boosting algorithm is determined on the ensemble of classifiers derived from each iteration of the algorithm. One of the fundamental problems of the development of different boosting algorithms is choosing the weights and to define rules for an ensemble of classifiers. In recent years, many authors presented various concepts based on the boosting idea [6,9]. In this article we present a new extension of AdaBoost [5] algorithm in which a linear modification of the weights was applied.

This paper is organized as follows: Sect. 2 introduces the necessary terms of the AdaBoost algorithm. In the next section there is our modification of the presented algorithm. Section 4 presents the experiment results comparing AdaBoost with our modification. Finally, some conclusions are presented.

## 2 AdaBoost Algorithm

In the work [5] weak and strong learning algorithms were discussed. The weak algorithms can classify the object better than random, on the other hand strong


M. Choraś and R.S. Choraś (eds.), *Image Processing and Communications Challenges 9*, Advances in Intelligent Systems and Computing 681, DOI 10.1007/978-3-319-68720-9_11

algorithms can classify the object accurately. Schapire formulated the first algorithm to "boost" a weak classifier. The main idea of boosting is to improve the prediction of weak learning algorithms by creating a set of weak classifiers which is a single strong classifier. The well-known and widely applied is the AdaBoost algorithm. Its main steps are as follows [2] (Tables 1 and 2):

**Table 1.** AdaBoost algorithm

| | |
|---|---|
| 1. | Initialize the weight vector $w_{1,1} = ... = w_{1,n} = 1/n$. |
| 2. | Assign weights to the learning sample $LS_n$. |
| 3. | For $t = 1, 2, ..., T$: |
| a. | At the base of $LS_n$ learn the classifier $\Psi_t$, |
| b. | Calculate the classification error $e_t = \sum_{i=1}^{n} w_{t,i} * I(\Psi_t(x_i) \neq j_i)$, |
| c. | Calculate $c_t = \frac{\ln(1-e_t)}{e_t}$ |
| d. | Update weights: $w_i(t+1) = \frac{w_{t,i} \exp(c_t * I(\Psi_t(x_i) \neq j_i))}{\sum_{j=1}^{n}(w_{t,j} \exp(c_t * I(\Psi_t(x_i) \neq j_i)))}, \quad i = 1, ..n,$ |
| e. | Assign updated weights to the learning sample $LS_n$. |
| 4. | Classify observation $x$ according to the rule: $\Psi_{AdaBoost}(x) = sign\left(\sum_{t=1}^{T} c_t \Psi_b(x)\right)$. |

**Table 2.** Notation of the AdaBoost algorithm

| | |
|---|---|
| $i$ | Observation number, $i = 1, ..., n$. |
| $t$ | Stage number, $t = 1, ..., T$. |
| $x_i$ | A $p$-dimensional vector containing the quantitative variables of the $i$th observation. |
| $\Psi_t$ | The weak classifier at the $t$th stage. |
| $w_{t,i}$ | The weight of the $i$th observation at the $t$th stage, $\sum_i w_{t,i} = 1$. |
| $I$ | The indicator function, $I(\Psi_t(x_i) \neq j_i)$. |
| $e_t$ | The classification error at the $t$th stage, $\sum_i w_{t,i} I_{t,i}$. |
| $c_t$ | The weight of $e_t$. |
| sign($x$) | $= 1$ if $x \geq 0$ and $= -1$ otherwise. |

One of the main steps in the algorithm is to maintain a distribution of the training set using the weights. Initially, all weights of the training set observations are set equally. If an observation is incorrectly classified (at the current stage) the weight of this observation is increased. Similarly, the correctly classified observation receives less weight in the next step. For this reason the weak learner is forced to focus on the and examples from the training set in each next step of the algorithm. In each step of the AdaBoost algorithm the best weak classifier according to the current distribution of the observation weights is found.

The goodness of a weak classifier is measured by its error. Based on the value of this error the ratio is calculated. The final prediction of the AdaBoost algorithm is a weighted majority vote of all weak classifiers.

## 3 AdaBoost Algorithm with Linear Modification of the Weights

One of the main factors that have an effect on the action of the AdaBoost algorithm is the selection of weights assigned to individual elements of the learning set. Let's propose then a modification of the AdaBoost algorithm in which a linear modification of the weights is introduced. The value of factor $c_t$ is modified in point 4d. The value of the modification depends on the iteration of the algorithm $t$. In experimental studies we have assumed that the value of the coefficient after modification (point 4d) is $1.25, 1.5, 1.75, 2, 2.25$ or $2.5$ times higher in the first iteration compared to the original algorithm (point 4c). The proposed algorithm steps are presented in Table 3.

**Table 3.** Lmw-AdaBoost algorithm

1. Initialize $a$ and $b$.
2. Initialize the weight vector $w_{1,1} = ... = w_{1,n} = 1/n$.
3. Assign weights to the learning sample $LS_n$.
4. For $t = 1, 2, ..., T$:
   a. At the base of $LS_n$ learn the classifier $\Psi_t$,
   b. Calculate the classification error $e_t = \sum_{i=1}^{n} w_{t,i} * I(\Psi_t(x_i) \neq j_i)$,
   c. Calculate $c_t = \frac{\ln(1-e_t)}{e_t}$
   d. Calculate $c_t = c_t * (a * t + b)$
   e. Update weights:
      $w_i(t+1) = \frac{w_{t,i} \exp(c_t * I(\Psi_t(x_i) \neq j_i))}{\sum_{j=1}^{n}(w_{t,j} \exp(c_t * I(\Psi_t(x_i) \neq j_i)))}, \quad i = 1, ..n,$
   f. Assign updated weights to the learning sample $LS_n$.
5. Classify observation $x$ according to the rule:
   $\Psi_{Lmw-AdaBoost}(x) = sign\left(\sum_{t=1}^{T} c_t \Psi_t(x)\right)$.

In the earlier work [1] we proposed the changes in weights based on interval-valued fuzzy sets and in the work [11] the linear combination of the upper and lower value of the weights to brain-computer interface was applied.

## 4 Experiments

To test Lmw-AdaBoost algorithm, we performed experiments on Pima data set. The feature selection process [10] was performed to indicate four most informative features for this data set. The final results are obtained via the 10-fold-cross-validation method.

The results for the twenty-five iterations of AdaBoost and proposed Lmw-AdaBoost algorithms are presented in Table 4.

**Table 4.** The results of experiments

| Iter. | AdaBoost | Lmw-AdaBoost | | | | | |
|---|---|---|---|---|---|---|---|
| | | $1.25 * c_t$ | $1.5 * c_t$ | $1.75 * c_t$ | $2 * c_t$ | $2.25 * c_t$ | $2.5 * c_t$ |
| 1 | 0.280 | 0.280 | 0.280 | 0.280 | 0.280 | 0.280 | 0.280 |
| 2 | 0.280 | 0.280 | 0.280 | 0.280 | 0.593 | 0.720 | 0.720 |
| 3 | 0.277 | **0.277** | 0.279 | 0.358 | 0.283 | 0.279 | 0.279 |
| 4 | 0.263 | **0.262** | 0.266 | 0.271 | 0.311 | 0.720 | 0.721 |
| 5 | 0.264 | 0.267 | 0.299 | 0.318 | **0.263** | 0.279 | 0.277 |
| 6 | 0.262 | **0.249** | 0.263 | 0.271 | 0.302 | 0.721 | 0.721 |
| 7 | 0.259 | 0.270 | **0.251** | 0.28 | 0.289 | 0.279 | 0.277 |
| 8 | 0.266 | 0.262 | **0.254** | 0.263 | 0.271 | 0.720 | 0.723 |
| 9 | 0.262 | 0.257 | 0.248 | **0.246** | 0.332 | 0.279 | 0.277 |
| 10 | 0.262 | 0.267 | 0.249 | **0.246** | 0.275 | 0.720 | 0.723 |
| 11 | 0.267 | 0.251 | **0.250** | 0.271 | 0.274 | 0.280 | 0.277 |
| 12 | 0.264 | **0.254** | 0.255 | 0.259 | 0.262 | 0.279 | 0.721 |
| 13 | 0.263 | 0.258 | **0.251** | 0.263 | 0.270 | 0.361 | 0.277 |
| 14 | 0.259 | 0.259 | **0.250** | 0.263 | 0.266 | 0.312 | 0.721 |
| 15 | 0.260 | 0.255 | **0.254** | 0.266 | 0.279 | 0.324 | 0.279 |
| 16 | 0.260 | 0.257 | **0.249** | 0.263 | 0.263 | 0.307 | 0.721 |
| 17 | 0.262 | 0.257 | **0.251** | 0.258 | 0.267 | 0.264 | 0.270 |
| 18 | 0.260 | 0.257 | **0.251** | 0.253 | 0.258 | 0.257 | 0.284 |
| 19 | 0.262 | 0.257 | **0.251** | 0.258 | 0.255 | 0.257 | 0.270 |
| 20 | 0.262 | 0.257 | 0.251 | 0.257 | **0.244** | 0.258 | 0.259 |
| 21 | 0.262 | 0.257 | 0.251 | 0.257 | **0.249** | 0.254 | 0.329 |
| 22 | 0.262 | 0.257 | 0.251 | 0.255 | **0.250** | 0.259 | 0.286 |
| 23 | 0.262 | 0.257 | 0.251 | 0.257 | **0.249** | 0.259 | 0.297 |
| 24 | 0.262 | 0.257 | 0.251 | 0.255 | **0.246** | 0.257 | 0.279 |
| 25 | 0.262 | 0.257 | 0.251 | 0.257 | **0.245** | 0.262 | 0.264 |

The best results (since the third iteration) are in bold. In general the results for the AdaBoost algorithm are worse than the proposed modifications Lmw-AdaBoost. For the first twelve iterations no clear results were obtained. In the iterations 13–19 the best is the algorithm in which the primary coefficient $c_t$ is increased 1.5 times in the first iteration. In the last iteration this coefficient is unchanged therefore the parameters $a$ and $b$ are equal $-0.020833333$ 1.541666667 respectively. In recent iterations the best is the algorithm in which parameters $a$

and $b$ are equal $-0.041666667$ $2.041666667$ respectively. With such parameters in the first iteration coefficient $c_t$ is increased 2 times. The obtained results show an improvement in the quality of the proposed modification the AdaBoost algorithm with respect to the ordinal one.

## 5 Conclusions

In this paper we presented the new Lmw-AdaBoost algorithm. It is a modification of the AdaBoost algorithm in which it was changed coefficient $c_t$. Consequently, the change affects the weights assigned to the individual learning objects. Changes compared to the original algorithm are linear. The value of the change is greater in the initial iterations.

The experiments have been carried out on Pima data sets. The aim of the experiments was to compare the proposed algorithm and the original AdaBoost algorithm. The results obtained show an improvement in the classification quality of the proposed method with respect to the original one.

Future work might include the proposed modification in other boosting algorithms such as RealAdaBoost or GentleAdaBoost as well as application of the proposed methods for various practical task [3,4,8] or testing other data sets.

**Acknowledgments.** This work was supported by the statutory funds of the Department of Systems and Computer Networks, Wroclaw University of Science and Technology.

## References

1. Burduk, R.: The AdaBoost algorithm with the imprecision determine the weights of the observations. In: Asian Conference on Intelligent Information and Database Systems, pp. 110–116. Springer, Cham (2014)
2. Dmitrienko, A., Chuang-Stein, C., D'Agostino, R.B.: Pharmaceutical statistics using SAS: a practical guide. SAS Institute (2007)
3. Forczmański, P., Łabędź, P.: Recognition of occluded faces based on multi-subspace classification. In: Computer Information Systems and Industrial Management, pp. 148–157. Springer, Heidelberg (2013)
4. Frejlichowski, D.: An algorithm for the automatic analysis of characters located on car license plates. In: International Conference Image Analysis and Recognition, pp. 774–781. Springer, Heidelberg (2013)
5. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory, pp. 23–37. Springer, Heidelberg (1995)
6. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. ICML **1996**, 148–156 (1996)
7. Kearns, M., Valiant, L.: Cryptographic limitations on learning boolean formulae and finite automata. J. ACM (JACM) **41**(1), 67–95 (1994)
8. Kozik, R., Choraś, M.: The HTTP content segmentation method combined with adaboost classifier for web-layer anomaly detection system. In: International Conference on EUropean Transnational Education, pp. 555–563. Springer, Heidelberg (2016)

9. Oza, N.C.: Boosting with averaged weight vectors. Multiple Classifier Systems **2709**, 15–24 (2003)
10. Rejer, I.: Genetic algorithms for feature selection for brain-computer interface. Int. J. Pattern Recogn. Artif. Intell. **29**(5), 1559008 (2015)
11. Rejer, I., Burduk, R.: Classifier selection for motor imagery brain computer interface. In: IFIP International Conference on Computer Information Systems and Industrial Management, pp. 122–130. Springer, Heidelberg (2017)