The AdaBoost Algorithm with the Imprecision Determine the Weights of the Observations

Robert Burduk

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

Abstract. This paper presents the AdaBoost algorithm that provides for the imprecision in the calculation of weights. In our approach the obtained values of weights are changed within a certain range of values. This range represents the uncertainty of the calculation of the weight of each element of the learning set. In our study we use the boosting by the reweighting method where each weak classifier is based on the recursive partitioning method. A number of experiments have been carried out on eight data sets available in the UCI repository and on two randomly generated data sets. The obtained results are compared with the original AdaBoost algorithm using appropriate statistical tests.

Keywords: AdaBoost algorithm, weight of the observation, machine learning.

1 Introduction

Boosting is a machine learning effective method of producing a very accurate classification rule by combining a weak classifiers [1]. 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 learns on 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 defining rules for an ensemble of classifiers. In recent years, many authors presented various concepts based on the boosting idea [2], [3], [4], [5]. There are also many studies showing the application of this method in the medical diagnosis problem [6] or in the multi-label classification problem [7].

In this article we present a new extension of the AdaBoost [8] algorithm. This extension is for the weights used in samples of the training sets. The original weights are the real number from the interval $[0, 1]$. We propose two approaches to this problem. In one of them in the early iterations weights are larger than in

N.T. Nguyen et al. (Eds.): ACIIDS 2014, Part II, LNAI 8398, pp. 110–116, 2014.

the original algorithm. In the second, in the early iterations weights are smaller than in the original algorithm.

This paper is organized as follows: In section 2 the AdaBoost algorithm is presented. In section 3 the our modification of the AdaBoost algorithm are presented. Section 4 presents the experiment results comparing AdaBoost with our modification. Finally, some conclusions are presented.

2 AdaBoost Algorithm

The first algorithm utilising the idea of boosting was proposed by Schapire in 1990 [9]. It concerned the binary classification problem, for which a set of three classifiers was proposed, and the final response of that set of classifiers was determined at the basis of simple majority of votes. The first of the component classifiers was a weak classifier, for the second one a half of the learning sample was constituted by misclassified observations by the first classifier. The third one used as the learning set those observations from the sample, which were placed in various groups by the two earlier classifiers.

Later, modifications of the original boosting algorithm were proposed. The first one concerned simultaneous combining of many weak classifiers [10]. In 1997, the AdaBoost algorithm was presented [8], which solved several practical difficulties noticed earlier. Its name is an acronym derived from Adaptive Boosting concept. In this case, adaptation concerns readjustment to errors of its component classifiers which result from their activity. Now, we are going to discuss the AdaBoost algorithm action for a case of two classes with Ψ_b classifier assuming values from the set $\{-1, 1\}$. For those assumptions steps of the algorithm look as follows [13] (See Tab. 1):

Table 1. The AdaBoost algorithm

1.	Let $w_{1,1} = ... = w_{1,n} = 1/n$
2.	For $t = 1, 2, ...T$ do:
a.	Fit f_t using weights $w_{t,1}, ..., w_{t,n}$, and compute the error e_t
b.	Compute $c_t = \ln((1 - e_t)/e_t)$.
c.	Update the observations weights: $w_{t+1,i} = w_{t,i} \exp(c_t, I_{t,i}) / \sum_{j=1}^{n}(w_{t,i} \exp(c_t, I_{t,i})), \quad i = 1, ..n.$
3.	Output the final classifier: $\hat{y}_i = F(x_i) = sign(\sum_{t=1}^{T} c_t f_t(x_i))$.

Action of the AdaBoost algorithm begins with assigning all objects from a learning set the corresponding weights reflecting the difficulty degree in correct classifying of a given case. At the beginning, weights are equal and amount to $1/n$, where n is the number of elements in a learning set. In the main loop of the algorithm - point 2 - so many component classifiers is created how many boosting iterations were foreseen. In the 2c step the level of error for Ψ_b qualifier is estimated, which takes into account weights of individual elements from a

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 ith observation.
y_i	A scalar quantity representing the class membership of the ith observation, $y_i = -1, 1$.
f_t	The weak classifier at the tth stage.
$f_t(x_i)$	The class estimate of the ith observation at the tth stage.
$w_{t,i}$	The weight of the ith observation at the tth stage, $\sum_i w_{t,i} = 1$.
$I_{t,i}$	The indicator function, $I(f_t(x_i) \neq y_i)$.
e_t	The classification error at the tth stage, $\sum_i w_{t,i} I_{t,i}$.
c_t	The weight of f_t.
$\text{sign}(x)$	$= 1$ if $x \geq 0$ and $= -1$ otherwise.

learning set. Thus, it is a weighted sum and not a fraction of misqualified observations. Further, the c_b factor is being determined, used for weights updating. New values of weights are normalised to a unit. The c_b coefficient is selected so that the observation weights misclassified by Ψ_b are increased, and instead, the correctly classified are decreased. Due to this, in subsequent algorithm iterations increases the probability with which an object misclassified in b iteration will be drawn to bootstrap sample LS_n^{b+1}. In subsequent iterations a component classifier is focused on more difficult samples. It result form the fact, that subsequent bootstrap sample is drawn from a distribution depending on weights of individual samples - point 2a.

The final decision of combined classifier is also dependant on the c_b coefficient. It can be said that each classifier receives its weight which is equal to that coefficient, and a classifier with higher prediction correctness has greater share in final decision of the combined classifier.

It should be also noted, that AdaBoost algorithm, in opposite to the bagging algorithm, can not be implemented at many machines simultaneously. This is caused by the fact that each subsequent component classifier is depending on results of its predecessor.

The AdaBoost algorithm presented above may be used for classifiers returning their response in a form of class label. In the work [11] a general form of the algorithm was proposed for the binary classification problem called the *real* version of AdaBoost. In this case a response of the component classifiers are estimators of a posteriori probability $\hat{p}(1|x), \hat{p}(-1|x) = 1 - \hat{p}(1|x)$.

The boosting algorithms presented above are based at resampling procedure [12]. In each of the B iterations n observations is being drawn with replacement with probability proportional to their current weights (step 2a). As earlier mentioned, weights are updated so, as to increase a share of misclassified samples in the learning set.

In case the component classifiers are able to benefit directly from weights of individual observations, than we talk of a boosting variation by reweighting [12].

In this approach, each of the component classifiers receives weighted information on each element of a learning set. Thus, there are no various learning sets, in understanding of the appearance of individual observations, for subsequent iterations of the algorithm. Each learning set LS_n^b contains the same observations, and instead, their weights are changing. An algorithm utilising the reweighted version is fully deterministic, as sampling is not present in this case.

3 AdaBoost Algorithm with the Imprecision Determine the Weights of the Observations

As we have previously described one of the main problems of the development of different boosting algorithms is the choice of weights. They concern the weights of the observation $w_{t,i}$ and are needed to determine the weighted error e_t of each learned classifier. Now we present two cases of changes in the obtained weights (step 2a in algorithm 1). In one of them in the early iterations weights are larger than in the original algorithm, but in the final iterations smaller than in the original algorithm. In order to change the received weights in the original AdaBoost algorithm the λ parameter is introduced. It defines uncertainty as it received the original weights. The algorithm in this case labeled as lsw-AdaBoost and it is as follows:

Table 3. The lsw-AdaBoost algorithm

1.	Determine the value of λ
2.	Let $w_{1,1} = ... = w_{1,n} = 1/n$
3.	For $t = 1, 2, ...T$ do:
a.	Fit f_t using weights $w_{t,1}, ..., w_{t,n}$, and compute the error e_t
b.	Fit $e_t = e_t * ((1 + (T/2 - t)) * \lambda)$
c.	Compute $c_t = \ln((1 - e_t)/e_t)$.
d.	Update the observations weights: $w_{t+1,i} = w_{t,i} \exp(c_t, I_{t,i}) / \sum_{j=1}^{n}(w_{t,i} \exp(c_t, I_{t,i})), \quad i = 1, ..n.$
3.	Output the final classifier: $\hat{y}_i = F(x_i) = sign(\sum_{t=1}^{T} c_t f_t(x_i))$.

In the second case in the finaly iterations weights are larger than in the original algorithm, but in the early iterations smaller than in the original algorithm. In this case, we change the step 3b of the algorithm 3. It is labeled now as slw-AdaBoost and it is as follows: In both of these cases, only in the middle iteration weights are the same as in the original AdaBoost algorithm.

4 Experiments

In the experiential research ten data sets were tested. Eight data sets come from the UCI repository [14] and two are generated randomly. One of them is called the banana distribution and has objects generated according to the procedure [15], the second one, instead, has random objects drawn in accordance with

Table 4. The slw-AdaBoost algorithm

1.	Determine the value of λ
2.	Let $w_{1,1} = ... = w_{1,n} = 1/n$
3.	For $t = 1, 2, ...T$ do:
a.	Fit f_t using weights $w_{t,1}, ..., w_{t,n}$, and compute the error e_t
b.	Fit $e_t = e_t * ((1 + (t - T/2)) * \lambda)$
c.	Compute $c_t = \ln((1 - e_t)/e_t)$.
d.	Update the observations weights: $w_{t+1,i} = w_{t,i} \exp(c_t, I_{t,i}) / \sum_{j=1}^{n}(w_{t,i} \exp(c_t, I_{t,i})), \quad i = 1, ..n.$
3.	Output the final classifier: $\hat{y}_i = F(x_i) = sign(\sum_{t=1}^{T} c_t f_t(x_i))$.

the procedure [16] – Higleyman distribution. In both cases the a priori probability for two classes equals 0.5, and for each class 200 elements were generated. The numbers of attributes, classes and available examples of all the data sets are presented in Tab. 5. The results are obtained via 10-fold-cross-validation method. In the experiments, the value of the parameter λ was set at 0.004. The value of this parameter determines how to change the weight of the observation. In this case, they are increased by the value of 0.004. The same value of weight, as in the original AdaBoost algorithm , is in the half of the assumed iterations, it is on $T/2$. That is, in the initial iterations of the weight are smaller than in the original AdaBoost algorithm, and after half iteration larger than in the original.

Table 5. Description of data sets selected for the experiments

Data set	example	attribute	class
Banana	400	2	2
Breast Cancer Wis.(Original)	699	10(8)	2
Haberman's Survival	306	3	2
Highleyman	400	2	2
ILPD (Indian Liver Patient)	583	10	2
Mammographic Mass	961	6	2
Parkinsons	197	22(23)	2
Pima Indians Diabetes	768	8	2
Sonar (Mines vs. Rocks)	208	60	2
Statlog (Heart)	270	13	2

Tab. 6 shows the results of classification for the AdaBoost algorithm and its modifications proposed in the work. The results are for 30, 40 and 50 iterations of the algorithms. Tab. 6 shows additionally the average ranks which were obtained in accordance with the Friedman test [17], [18]. The resulting average

Table 6. Classification error for different number iterations of AdaBoost algorithms and average rank produced by Friedman test

	Algorithm								
	AB	lsw-AB	slw-AB	AB	lsw-AB	slw-AB	AB	lsw-AB	slw-AB
	After 50 iter.			After 40 iter.			After 30 iter.		
Banan	0.044	0.045	0.042	0.045	0.047	0.043	0.047	0.050	0.044
Cancer	0.061	0.060	0.057	0.063	0.058	0.057	0.063	0.061	0.056
Haber.	0.288	0.270	0.286	0.288	0.274	0.287	0.288	0.279	0.287
Hig.	0.082	0.071	0.083	0.079	0.073	0.083	0.077	0.068	0.082
Liver	0.325	0.301	0.309	0.321	0.304	0.308	0.326	0.304	0.311
Mam.	0.186	0.191	0.199	0.186	0.190	0.199	0.186	0.208	0.199
Park.	0.145	0.140	0.124	0.145	0.136	0.121	0.145	0.150	0.117
Pima	0.240	0.239	0.250	0.237	0.240	0.243	0.239	0.250	0.237
Sonar	0.197	0.189	0.211	0.203	0.197	0.206	0.195	0.194	0.208
Statlog	0.356	0.356	0.337	0.356	0.388	0.338	0.356	0.388	0.338
Av. rank	2.35	1.65	2.00	2.20	1.80	2.00	2.20	2.10	1.70

rank shows an improvement in classification obtained by the proposed in the paper modification of the AdaBoost algorithm.

In order to determine whether the proposed modification method differs from the orginal AdaBoost algorithm the post-hoc Bonferroni-Dunn test was performed. The critical difference for the described experiments equals 0.87 at $p = 0.1$. So, we can conclude that no statistically significant differences in classification error were observed between the proposed modifications of the AdaBoost algorithms and the standard AdaBoost algorithm. However, the received mean ranks of 50 iterations are close to the critical difference.

5 Conclusions

In this paper we presented modification of the AdaBoost algorithm. We consider the situation where the weights are changed within a certain range of values. The paper proposes two approaches. In one of them in the early iterations weights are larger than in the original algorithm. In the second, in the early iteration weights are smaller than in the original algorithm. In our study we use boosting by the reweighting method where each weak classifier is based on the recursive partitioning method. The received results for ten data sets show improvement in classification obtained by the proposed in the paper modification of the AdaBoost algorithm.

Acknowledgments. The work was supported in part by the statutory funds of the Department of Systems and Computer Networks, Wroclaw University of Technology and by the by The Polish National Science Centre under the grant N N519 650440 which is being realized in years 2011–2014.

References

1. Kearns, M., Valiant, L.: Cryptographic limitations on learning boolean formulae and finite automata. J. Assoc. Comput. Mach. 41(1), 67–95 (1994)
2. Chunhua, S., Hanxi, L.: On the Dual Formulation of Boosting Algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(12), 2216–2231 (2010)
3. Oza, N.C.: Boosting with Averaged Weight Vectors. In: Windeatt, T., Roli, F. (eds.) MCS 2003. LNCS, vol. 2709, pp. 15–24. Springer, Heidelberg (2003)
4. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy, pp. 148–156 (1996)
5. Wozniak, M.: Proposition of Boosting Algorithm for Probabilistic Decision Support System. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3036, pp. 675–678. Springer, Heidelberg (2004)
6. Wozniak, M.: Boosted decision trees for diagnosis type of hypertension. In: Oliveira, J.L., Maojo, V., Martín-Sánchez, F., Pereira, A.S. (eds.) ISBMDA 2005. LNCS (LNBI), vol. 3745, pp. 223–230. Springer, Heidelberg (2005)
7. Kajdanowicz, T., Kazienko, P.: Boosting-based Multi-label Classification. Journal of Universal Computer Science 19(4), 502–520 (2013)
8. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boostin. Journal of Computer and System Scienses 55(1), 119–139 (1997)
9. Schapire, R.E.: The Strenght of Weak Learnability. Machine Learning 5, 197–227 (1990)
10. Freund, Y.: Boosting a Weak Learning Algorithm by Majority. Information and Computation 121, 256–285 (1995)
11. Friedman, J., Hastie, T., Tibshirani, R.: Additive Logistic Regression: A Statistical View of Boosting. The Annals of Statistics 38, 337–374 (2000)
12. Seiffert, C., Khoshgoftaar, T.M., Hulse, J.V., Napolitano, A.: Resampling or Reweighting: A Comparison of Boosting Implementations. In: 2008 20th IEEE International Conference on Tools with Artificial Intelligence, pp. 445–451 (2008)
13. Dmitrienko, A., Chuang-Stein, C.: Pharmaceutical Statistics Using SAS: A Practical Guide. SAS Press (2007)
14. Murphy, P.M., Aha, D.W.: UCI repository for machine learning databases. Technical Report, Department of Information and Computer Science, University of California, Irvine (1994), `http://www.ics.uci.edu/~mlearn/databases/`
15. Duin, R.P.W., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D., Tax, D., Verzakov, S.: PR-Tools4.1, A Matlab Toolbox for Pattern Recognition. Delft University of Technology (2007)
16. Highleyman, W.H.: The design and analysis of pattern recognition experiments. Bell System Technical Journal 41, 723–744 (1962)
17. Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1(1), 3–18 (2011)
18. Trawinski, B., Smetek, M., Telec, Z., Lasota, T.: Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. International Journal of Applied Mathematics and Computer Science 22(4), 867–881 (2012)