Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference

Classifying with AdaBoost.M1: The Training Error Threshold Myth

Antônio Leães,[1] Paulo Fernandes,[2] Lucelene Lopes,[2] Joaquim Assunção[3]
[1] Computer Science Department - PUCRS University - Porto Alegre - Brazil
Email: antonio.nascimento@acad.pucrs.br
[2] UNDL Foundation - Geneva - Switzerland
Email: {p.fernandes, l.lopes}@undlfoundation.org
[3] Federal Institute Farroupilha (IF Farroupilha) - São Borja - Brazil
joaquim.assuncao@iffarroupilha.edu.br

Abstract

This paper is an empirical study trying to shed some light over an obscure, and yet important, input parameter of the popular AdaBoost.M1 algorithm, which is the more frequently employed implementation of the well known Boosting method for classification. The algorithm creators and many other researchers on the topic assumed that the training error threshold should correlate with the number of classes in the target data set, and logically, most data sets should use a threshold value of 0.5. In this paper we present empirical evidence that this is not a fact, but probably a myth originated by the mistaken application of the ensemble effect theoretical assumption. Next, we focus our study in a better suggestion for defining this threshold in a general case.

Introduction

It has been some time since classification methods have been used for machine learning and several efforts in the area of knowledge discovery in databases have been done to increase the effectiveness of these methods (Tan, Steinbach, and Kumar 2006). Classification is a task of supervised learning for data analysis and to extract models that describe classes of important data, with a number of applications, including fraud detection, destination marketing, performance prediction, manufacturing, and medical diagnosis (Han, Pei, and Kamber 2011). Within the area of classification, there are ensemble methods, such as Boosting (Schapire 1990), which combine the results of many classifiers aiming to improve the classification accuracy.

The Boosting method was initially proposed with two distinct algorithms, AdaBoost.M1 and AdaBoost.M2 (Freund and Schapire 1996). Even though both algorithms were discussed in Freund and Schapire's work, only AdaBoost.M1 experienced a large usage. Nevertheless, even the authors mention as a disadvantage of AdaBoost.M1 the application for data sets with more than two classes. According to the authors, such disadvantage is a consequence of a specific test within the algorithm to drop classifiers with an accuracy below 50%, frequently called the ϵ training error threshold.

This anticipated limitation of AdaBoost.M1 is ignored for the majority of users that keep using this algorithm indistinctly for all data sets. As for the research community, a large number of studies suggest adaptations to AdaBoost.M1 to improve classification results, *e.g.*, (Milidiú et al. 2009; Freund, Schapire, and Abe 1999; Friedman 2002). Curiously, few research works observe the training error threshold (ϵ) which remains a relatively obscure point, even thou some researchers believe it is logical application of the ensemble effect (Hansen and Salamon 1990). AdaBoost.M1 uses ϵ to interrupt the classification process if the classifier shows a loss in its matching capacity. Following the remarks of Freund and Schapire, Zhu et al.'s tried to use a training error threshold dependent of the number of classes NC, *i.e.*, $\epsilon = 1/NC$, which correspond to assume that the classifier must be better than a random choice (Zhu et al. 2009).

Our goal in this paper is to empirically analyze other possible choices of numerical values for ϵ threshold with respect to its impact in the accuracy and execution time over some randomly chosen data sets. To do so, we took 32 data sets previously made available and employed in a comparison study (Fernandes, Lopes, and Ruiz 2010) using AdaBoost.M1.

- First we want to empirically verify if threshold $\epsilon = 0.5$ is effective to data sets with only two classes;
- Secondly, we want to verify if the approach of using $\epsilon = 1/NC$ brings any help;
- At last, we try to propose a more effective way to choose the numeric value of ϵ.

The next section briefly describes the Boosting method, and the AdaBoost details as employed in this paper experiments. The third section describes the experiments' methodology, including the employed data sets. The fourth section presents the numerical results for the experiments and a comparative discussion. Finally, this paper contribution is summarized and future works are suggested.

Boosting method

Boosting was proposed by (Schapire 1990) and it is frequently known as an alternative ensemble classifier method to the also popular Bagging method (Breiman 1996). However, unlike Bagging in which the generation of classifiers is independent and affected only by random decisions, Boosting implementations generate new classifiers taking into account the performance of previous generated classifiers.

Specifically, such procedure is referred as an iterative procedure to change in an adaptive way the distribution of training examples in order to have the classifiers focusing in examples that are hard to classify (Tan, Steinbach, and Kumar 2006). The first implementation of the method is called AdaBoost.M1 (Adaptive Boosting) (Freund and Schapire 1996), which builds a set of base classifiers through a weighted voting (Freund and Schapire 1995). According to the authors, AdaBoost.M1 performs very well for data sets with only two classes, but it is too restrictive to data sets with more than two classes. Therefore, another implementation of the Adaptive Boosting, called AdaBoost.M2, is aimed to cope with the problems brought by the arbitrary use of the training error threshold $\epsilon = 0.5$.

In such way, the major difference between the two algorithms is the fact that, according to (Freund and Schapire 1996), AdaBoost.M2 is better than its two-class effective counterpart AdaBoost.M1. While AdaBoost.M1 stops generating new classifiers when the accuracy is below the threshold $\epsilon = 0.5$, AdaBoost.M2 does not quit generating new classifiers, since it computes not an accuracy of new classifiers, but it computes a pseudo-loss to try to guess the input to generate new classifiers.

As will be seen in the results of our experiments, we did not find empirical evidences justifying the problem raised as motivation to propose AdaBoost.M2. This and the fact that the large majority of practitioners using Boosting concentrate their experiments with AdaBoost.M1, led us to focus our interest solely in this algorithm, more precisely, the version implemented in the popular software Weka (Witten and Frank 2005), using as base classifier J48, an implementation inspired on c4.5 (Quinlan 1986).

Methodology

The proposed study in this paper is empirical, therefore our great concern with the employed methodology. Three key aspects were taken into account for our methodology:

- the choice of target data sets;
- the choice of the software and hardware test bed;
- the care of statistical relevance of the results.

The data sets

All our experiments were conducted over a several of public data sets chosen without bias. Specifically, we took a collection of 32 data sets gathered with another purpose. These data sets were described in detail in the original work and they are not focused on theme, size, number of classes. Table 1 summarizes the data sets and its information was taken from (Fernandes, Lopes, and Ruiz 2010).

The first column of the Table 1 shows the identification of the data set file (ID). The second column shows the data set name (BD) and the original repository, being △ for University of California Irvine (Asuncion and Newman 2007) and ∇ for University of West Virginia (Boetticher, Menzies, and Ostrand 2007). The other columns show the information about the data, such as number of attributes (NA), number of instances (NI), number of classes of target attribute (NC) and the rate of imbalance (IR).

Table 1: Data sets

ID	BD	NA	NI	NC	IR
B01	Abalone△	9	4177	29	0.071
B02	Arrythmia△	280	452	13	0.520
B03	Audiology△	70	226	24	0.103
B04	Balance△	5	625	3	0.146
B05	Breast cancer△	10	286	2	0.165
B06	Car Evaluation△	7	1728	4	0.390
B07	CM1 software defect∇	22	498	2	0.645
B08	Datatrieve∇	9	130	2	0.690
B09	Desharnais∇	12	81	3	0.150
B10	Ecoli△	9	336	8	0.168
B11	Echo cardiongram△	12	132	3	0.054
B12	Glass△	11	214	6	0.116
B13	Heart (Cleveland)△	14	303	2	0.008
B14	Heart statlog△	14	270	2	0.012
B15	Hepatitis△	20	155	2	0.345
B16	JM1 software defect∇	22	10885	2	0.376
B17	Kr-vs-kp△	37	3196	2	0.002
B18	MW1 software defect∇	38	403	2	0.716
B19	Pima-diabetes△	9	768	2	0.091
B20	Post-operative△	9	90	3	0.366
B21	Primary-tumor△	18	339	21	0.066
B22	Reuse∇	28	24	2	0,063
B23	Solar Flare△	13	1389	8	0.682
B24	Tic-Tac-Toe Endgame△	10	958	3	0.094
B25	Thyroid (Allhyper)△	30	2800	4	0.928
B26	Thyroid (Hypothyroid)△	30	3772	4	0.807
B27	Thyroid (Sick euthyroid)△	26	3163	2	0.664
B28	Wbdc△	31	569	2	0.065
B29	Wisconsin breast cancer△	10	699	2	0.096
B30	Wine recognition△	14	178	3	0.065
B31	Yeast△	10	1484	10	0.137
B32	Zoo△	18	101	7	0.114

The rate of imbalance (IR) is a numerical index is calculated by:

$$IR = \frac{\left(\frac{STD}{(NI/NC)}\right)}{\sqrt{NC}}$$

It is the ratio between the standard deviation of the number of cases in each class (STD), by a completely balanced distribution of the cases between the classes (NI/NC), divided by the square root of the number of classes (NC). Hence, the rate of imbalance is normalized between 0 and 1. For example, when observing Table 1, we can see that the B17 has a low rate of imbalance, equal to 0.002, because its instances are divided by the two classes in 1669 and 1527. On the contrary, B25 rate of imbalance is high, because its instances are divided by its four classes in 62, 8, 7 and 2723 instances each. According to Fernandes et al. (Fernandes, Lopes, and Ruiz 2010), the rate of imbalance can be seen as the reverse entropy of the data set.

The hardware and software test bed

All our experiments were conducted MacBook Pro 2.9 GHz Intel core i5 with 8 Gb 1867 MHz DDR3 running MacOS v.10.12.1 using the Weka 3.6.12 software. The AdaBoost.M1 implementation was applied to all data sets generating up to 50 classifiers (50 iterations), using resample, using as base classifier J48 with confidence factor 0.25 and minimum number of objects equal to 2.

Each data set was repeatedly experimented with different values of the training error thresholds (ϵ) from 0.1 to 0.9. Additionally, some data sets had another value of ϵ tested according to the number of classes (NC), *e.g.*, B10 data set was tested with $\epsilon = 0.125$, since it has $NC = 8$.

For each possible value of ϵ, for each data set, 100 runs were made, and in each of those runs a different random seed is considered (seeds from 0 to 99), to set apart the impact of random decisions. This technique is similar to the analysis performed by previous works (Bauer and Kohavi 1999; Fernandes, Lopes, and Ruiz 2010) paying attention to the impact of randomness. For each of these experiments, a 10-fold stratified cross validation (Kohavi 1995) was performed in order to obtain an unbiased accuracy estimation.

The statistical relevance of results

The result of experiments was a reliable estimation of accuracy of each data set with different values of training error threshold (ϵ). To avoid the impact of randomness, the basic output (the average of the 10-fold cross validation) for each pack of 100 runs with different seeds the 5 highest and the five lowest accuracy results were discarded and the average of the remained 90 runs was considered as the result.

For the execution times a similar analysis was made. Each ϵ value for each data set was submitted to the 10-fold cross validation and the 100 runs with different seed values were performed. However, instead of discarding the 5 lowest and the five highest execution times, we kept the execution times of the same runs as in the accuracy results.

This decision was taken to keep the logic between the accuracy and execution time, since it was incorrect to consider to the average computation distinct runs for accuracy and for execution time. Therefore, the average values considered as numeric raw results in the next section are consistent with the median 90 experiments with respect to the accuracy.

Results Analysis

The numeric raw average results for accuracy and execution times were not included in this paper due to space restrictions. However, these result are depicted in Figure 1. The numeric values are available in the larger version of this paper at the address http://www.inf.pucrs.br/peg/pub/conferences/flairs2017long.pdf.

Is the threshold 0.5 adequate?

According to Freund and Schapire (1996), the training error threshold $\epsilon = 0.5$ is adequate for data sets with two classes. We will consider as the best ϵ value the one with the highest accuracy, and in a case of same accuracy, the one with the smaller execution time.

Table 2 shows the 14 data sets with $NC = 2$. Among these 14 data sets, the 0.5 value was never the best threshold. Table 3 shows the 18 data sets with $NC > 2$. Among these 18 data sets, the 0.5 value was never the best threshold.

The observation of these two grouped results clearly shows that the value 0.5 is not related at all to the number of

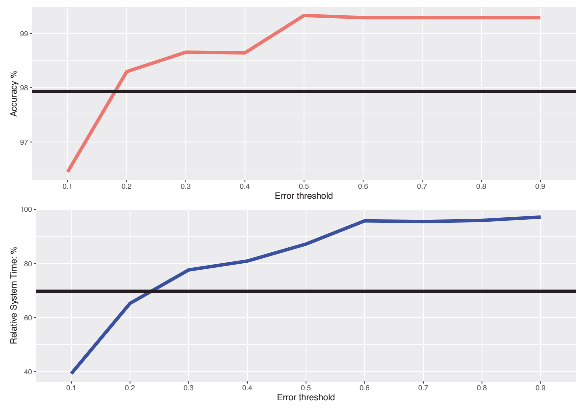

Figure 1: Average accuracy and execution time considering the 32 datasets. The horizontal straight line is the 1/NC value

Table 2: Best threshold for data sets with $NC = 2$.

data set	B5	B7	B8	B13	B14	B15	B16
ϵ	0.2	0.6	0.2	0.3	0.2	0.4	0.6
data set	B17	B18	B19	B22	B27	B28	B29
ϵ	0.1	0.6	0.4	0.3	0.1	0.1	0.1

Table 3: Best threshold for data sets with $NC > 2$.

data set	B1	B2	B3	B4	B6	B9	B10	B11	B12
ϵ	0.5	0.3	0.3	0.1	0.2	0.4	0.4	0.4	0.1
data set	B20	B21	B23	B24	B25	B26	B30	B31	B32
ϵ	0.1	0.5	0.1	0.2	0.1	0.1	0.2	0.1	0.1

classes. In all fairness, it is relevant to mention that rarely literature authors pay attention to execution time, and in many data sets the accuracy values did not change according to error training value. However, this fact alone shows that the arbitrary choice of value 0.5 is empirically unjustified.

Is the threshold $1/NC$ any better?

Observing the raw results, we notice that only for the data sets B9, B11 and B31 the use of $1/NC$ as training error threshold was adequate among the 32 experimented data sets. For data sets B23 and B32 a close accuracy and a small increase in execution let us enlarge to just 5 the number of data sets where the use of such way to compute ϵ value is somewhat effective. This result is even more clear than the previous one and it allow us to state that, based on our empirical evidences, estimate $\epsilon = 1/NC$ is not better than $\epsilon = 0.5$.

A consequence of this analysis is to discard, with all due respect to the previous work, the hypothesis that the training error threshold has any relation with the effectiveness of a weak learner in comparison to the random choice among the classes. We believe that such false claim was a *bona fide* mistake made in previous works (Freund and Schapire 1996; Zhu et al. 2009) because these authors were assuming a completely balanced data set, where the number of instances in each class is approximatively equal. However, this is often the case for the experimented 32 data sets in this paper, nor for the data sets in real applications.

Is it possible to suggest a better threshold?

A shallow analysis of the raw resuts indicates that the better threshold is achieved for 12 data sets with 0.1 (Table 4 shows the best thresholds for each data set). However, such analysis is purely an account data sets votes based on their best ϵ values, and the use of a threshold value of $\epsilon = 0.1$ would be very bad (at least less than 3% of accuracy) for data sets B1, B2, B3, B9, B10, B11, B13, B14, B15, B21, B22 and B24.

Note that such simplistic analysis is sufficient to discard threshold suggested values, which was done when we discarded $\epsilon = 0.5$ and $\epsilon = 1/NC$. In fact, such analysis is also the base to refute $\epsilon = 0.1$, since it only works fine for 12 data sets, while for 12 data sets it is not good enough.

Table 4: Best threshold for data sets.

threshold ϵ	data sets
0.1	B4, B12, B17, B20, B23, B25, B26, B27, B28, B29, B31, B32
0.2	B5, B6, B8, B14, B24, B30
0.3	B2, B3, B13, B22
0.4	B9, B10, B11, B15, B19
0.5	B1, B21
0.6	B7, B16, B18

Having in mind the empirical aspect of our analysis, it is rather prudent to refrain from a definitive suggestion about the best training error threshold. However, in the absence of further information, it seems that the threshold $\epsilon = 0.3$ was the best trade-off among the experimented options.

Final Remarks

As a preliminary work, we showed an empirical analysis of the AdaBoosting.M1 algorithm, the more popular and researched option of the Boosting method with respect to the choice of training error threshold. We used 32 data sets from different sources, and with different characteristics. Our experiments were conducted paying attention to the impact of randomness and the statistical relevance of average results.

We started showing that there was no practical reason to assume a threshold value of 0.5, even in the cases with only two possible classes. This is a common misconception that came from the seminal works proposing the algorithm, and carried out by the research community, probably due to a theoretical assumption that all data sets were balanced, which is far from the truth. Then, we had shown that trying to cope this problem using the threshold as a function of the number of classes ($1/NC$) was equally ineffective.

Finally, we had attempted to suggest a better threshold option, but the results were not completely conclusive, even thou our results indicate some advantage to adopt a threshold value of 0.3. Despite that, it is the authors' opinion that further studies with more data sets, and specially the search for correlations between the balance index of data sets and the threshold value could be an interesting and revealing future work.

References

Asuncion, A., and Newman, D. 2007. Uci machine learning repository. Available at https://ergodicity.net/2013/07/.

Bauer, E., and Kohavi, R. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning* 36(1-2):105–139.

Boetticher, G.; Menzies, T.; and Ostrand, T. 2007. The promise repository of empirical software engineering data. Available at http://promisedata.org/.

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24(2):123–140.

Fernandes, P.; Lopes, L.; and Ruiz, D. D. 2010. The impact of random samples in ensemble classifiers. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, 1002–1009. ACM.

Freund, Y., and Schapire, R. E. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, 23–37. Springer.

Freund, Y., and Schapire, R. E. 1996. Experiments with a new boosting algorithm. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, 148–156.

Freund, Y.; Schapire, R.; and Abe, N. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence* 14(771-780):1612.

Friedman, J. H. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38(4):367–378.

Han, J.; Pei, J.; and Kamber, M. 2011. *Data mining: concepts and techniques*. Elsevier.

Hansen, L. K., and Salamon, P. 1990. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.* 12(10):993–1001.

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1137–1145.

Milidiú, R. L.; Duarte, J. C.; do Exercito, C. T.; and da Informacao, D. d. T. 2009. Boosting at start. In *Proceedings of the IASTED International Conference*, volume 639, 028.

Quinlan, J. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

Schapire, R. E. 1990. The strength of weak learnability. *Machine learning* 5(2):197–227.

Tan, P.-N.; Steinbach, M.; and Kumar, V. 2006. *Introduction to data mining*. Pearson Education.

Witten, I. H., and Frank, E. 2005. *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition.

Zhu, J.; Zou, H.; Rosset, S.; and Hastie, T. 2009. Multi-class adaboost. *Statistics and its Interface* 2(3):349–360.