Empirical Comparison of Boosting Algorithms

Riadh Khanchel and Mohamed Limam

Institut superieur de gestion
41, Rue da la liberte, Le Bardo 2000 Tunis, Tunisia

Abstract. Boosting algorithms combine moderately accurate classifiers in order to produce highly accurate ones. The most important boosting algorithms are Adaboost and Arc-x(j). While belonging to the same algorithms family, they differ in the way of combining classifiers. Adaboost uses weighted majority vote while Arc-x(j) combines them through simple majority vote. Breiman (1998) obtains the best results for Arc-x(j) with $j = 4$ but higher values were not tested. Two other values for j, $j = 8$ and $j = 12$ are tested and compared to the previous one and to Adaboost. Based on several real binary databases, empirical comparison shows that Arc-x4 outperforms all other algorithms.

1 Introduction

Boosting algorithms are one of the most recent developments in classification methodology. They repeatedly apply a classification algorithm as a subroutine and combine moderately accurate classifiers in order to produce highly accurate ones. The first boosting algorithm, developed by Schapire(1990), converts a weak learning algorithm into a strong one. A strong learning algorithm achieves low error with high confidence while a weak learning algorithm drops the requirement of high accuracy. Freund (1995) presents another boosting algorithm, boost-by-majority, which outperforms the previous one.

Freund and Schapire (1997) present another boosting algorithm, Adaboost. It is the first adaptive boosting algorithm because its strategy depends on the advantages of obtained classifiers, called hypotheses. For binary classification, the advantage of a hypothesis measures the difference between its performance and random guessing. The only requirement of Adaboost is to obtain hypotheses with positive advantage. Furthermore, the final hypothesis is a weighted majority vote of the generated hypotheses where the weight of each hypothesis depends on its performance. Due to its adaptive characteristic, Adaboost has received more attention than its predecessors. Experimental results (Freund and Schapire (1996), Bauer and Kohavi (1999)) show that Adaboost decreases the error of the final hypothesis.

Breiman (1998) introduces the ARCING algorithm's family: **A**daptively **R**esampling and **C**ombining which Adaboost belongs to. In order to better understand the behavior of Adaboost, Breiman (1998) develops a simpler boosting algorithm denoted by Arc-x(j). This algorithm uses a different weight updating rule and combines hypotheses using simple majority vote. The best results of Arc-x(j) are obtained for $j = 4$. When compared to Adaboost, Breiman's results show that both algorithms perform equally well. Breiman (1998) argues that the success of Adaboost is not due to its way of combining hypotheses but on its adaptive property. He argues also that since higher values for j were not tested further improvement is possible.

In this paper, two other values for the parameter j of Arc-x(j) algorithm, $j = 8$ and $j = 12$, are tested and their performance compared to Adaboost and Arc-x4 in the subsampling framework using a one node decision tree algorithm.

In section two, the different boosting algorithms used are briefly introduced. In section three, the empirical study is described and the results are presented. Finally, section four provides a conclusion to this article.

2 Arcing algorithms

Adaboost was the first adaptive boosting algorithm. First, the general framework of boosting algorithms is introduced, then Adaboost and some of its characteristics are reviewed. Finally, arcing algorithm's family is discussed.

Given a labeled training set $(x_1, y_1), \ldots, (x_n, y_n)$, where each x_i belongs to the instance space X, and each label y_i to the label set Y. Here only the binary case is considered where $Y = \{-1, 1\}$. Adaboost applies repeatedly, in a series of iterations $t = 1, \ldots, T$, the given learning algorithm to a reweighted training set. It maintains a weight distribution over the training set. Starting with equal weight assigned to all instances, $D(x_i) = 1/n$, weights are updated after each iteration such that the weight of misclassified instances is increased. Weights represent instance importance. Increasing instance's weight will give it more importance and thus forcing the learning algorithm to focus on it in the next iteration. The learning algorithm outputs in each iteration a hypothesis that predicts the label of each instances $h_t(x_i)$. For a given iteration, the learning algorithm tends to minimize the error:

$$\epsilon_t = Pr[h_t(x_i) \neq y_i], \tag{1}$$

where $Pr[.]$ denote empirical probability on the training sample.

2.1 Adaboost

Adaboost requires that the learning algorithm outputs hypotheses with error less than 0.5. A parameter α_t is used to measure the importance assigned to each hypothesis. This parameter depends on hypothesis' performance. For the binary case this parameter is set to:

$$\alpha_t = \frac{1}{2} \ln(\frac{1 - \epsilon_t}{\epsilon_t}). \tag{2}$$

The weight distribution is updated using α_t (see Figure 2.1). This parameter is positive because Adaboost requires that the learning algorithm output hypotheses with error less than 0.5. At the end of the process, a final hypothesis is obtained by combining all hypotheses from previous iterations using weighted majority vote. The parameter α_t represents the weight of the hypothesis h_t generated in iteration t. The pseudocode of Adaboost for binary classification is presented in Figure 2.1.

Adaboost requires that the base learner performs better than random guessing. The error can be written as follows:

$$\epsilon_t = 1/2 - \gamma_t, \tag{3}$$

where γ_t is a positive parameter that represents the advantage of the hypothesis over random guessing. The training error of the final hypothesis is bounded by:

$$\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} \tag{4}$$

```
Given: $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in X, y_i \in Y = \{-1; +1\}$
1-Initialize $D_1(i) = 1/n$
2-For t = 1 to T:
    • Train the weak classifier using $D_t$ and get a hypothesis
      $h_t : X \mapsto \{-1; +1\}$
    • Compute        $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(x_i)$
    • If             $\epsilon_t \geq 0.5$ stop.
    • Choose:        $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$
    • Update:        $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
     where $Z_t$ is a normalization factor
3-output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$
```

Figure 2.1: Adaboost algorithm

This bound can be expressed in term of the advantage sequence γ_t:

$$\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq \exp(-2\sum_t \gamma_t^2). \tag{5}$$

Thus, if each hypothesis is slightly better than random guessing, that is $\gamma_t > \gamma$ for $\gamma > 0$, the training error will drop exponentially fast.

The bound of generalization error, or the error of the final hypothesis over the whole instance space X, depends on the training error, the size of the sample n, the Vapnik-Chervonenkis (VC, Vapnik 1998) dimension d of the weak hypothesis space and the number of boosting iterations T. The generalization error is at most:

$$Pr[H(x) \neq y] + \hat{O}\left(\sqrt{\frac{T.d}{n}}\right). \tag{6}$$

This bound depends on the number of iterations T and we would think that it will overfit as T becomes large but experimental results (Freund and Schapire (1996)) show that Adaboost continue to drop down generalization error as T becomes large.

2.2 Arcing family

Breiman (1998) used the ARCING term to describe the family of algorithms that Adaptively Resample data and Combine the outputted hypotheses. Adaboost was the first example of an arcing algorithm.

In order to study the behavior of Adaboost, Breiman developed an ad-hoc algorithm, Arc-x(j). This algorithm is similar to Adaboost but differs in the following:

- it uses a simpler weight updating rule:

$$D_{t+1}(i) = \frac{1 + m(i)^j}{\sum(1 + m(i)^j)}, \tag{7}$$

  where $m(i)$ is the number of misclassifications of instance i by classifiers $1, \ldots, t$ and j is an integer.
- classifiers are combined using simple majority vote.

Since the development of arcing family, Adaboost and Arc-x4 were compared in different framework and using different collections of datasets. Breiman (1998) and Bauer and Kohavi (1999) show that Arc-x4 has an accuracy comparable to Adaboost without using the weighting scheme to construct the final classifier. Breiman (1998) argues that higher values of j were not tested so improvement is possible.

In this empirical study, two other values of the parameter j, $j = 8$ and $j = 12$, are tested in the subsampling framework and compared to Adaboost and Arc-x4.

3 Empirical study

First, the base classifier and the performance measure used in the experiments are introduced then we the experimental results of each algorithm are presented. Finally, the performance of all algorithms are compared.

3.1 Base classifier and performance measure

Boosting algorithms require a base classifier as a subroutine that performs slightly better than random guessing. In our experiments, we use a simple algorithm, developed by Iba and Langley (1992), that induces a one node decision tree from a set of preclassified training instances.

In order to compare different boosting algorithms, we use a collection of binary data sets from UCI Machine learning Repository (Blakes et al. (1998)). Details of these data sets are presented in Table 2.

For each data set, we repeat the experiment 50 times. Each time, the data set is randomly partitioned into two equally sized sets. Each set is used once as a training set and once as a testing set. We run each algorithm for $T =$ 25 and 75 iterations and report the average test error.

Bauer and Kohavi (1999) measures of performance are used. For a fixed number of iterations, the performance of each algorithm is evaluated using test error averaged over all data sets. To measure improvement produced by a boosting algorithm, absolute test error reduction and relative test error reduction are used.

3.2 Results

Results are reported in Table 1 and interpreted as follows: for a fixed number of iterations, we evaluate the performance of each algorithm on the collection of data sets and on each data set. Then all algorithms are compared for 25 and 75 iterations using test error averaged over all data sets.

Table 1. Average test error for each algorithm for 25 and 75 iterations on each data set.

	base	Adaboost		Arc-x4		Arc-x8		Arc-x12	
Data	Classifier	25	75	25	75	25	75	25	75
Liv.	41.81 %	29.78%	29.35%	29.96%	28.94%	31.94%	29.24%	34.28%	29.60%
Hea.	28.96%	19.58%	20.38%	18.99%	18.93%	20.25%	19.41%	21.79%	20.07%
Ion.	18.93%	12.38%	11.32%	12.27%	11.83%	12.22%	11.21%	12.59%	11.01%
Bre.	8.32%	4.56%	4.62%	3.88%	3.87%	4.22%	4.14%	4.40%	4.26%
Tic.	34.66%	28.80%	28.68%	29.59%	28.43%	31.38%	28.82%	30.11%	29.36%
mean	26.54%	19.02%	18.87%	18.94%	18.40%	20.00%	18.56%	20.63%	18.86%

Adaboost results: Adaboost decreases the average test error by 7.52% for 25 iterations and by 7.67% for 75 iterations. All data sets have relative test error reduction higher than 15%. The results for 75 iterations are better than those obtained for 25 iterations except for breast cancer data and heart data.

Table 2. Data sets used in the experimental study

Data set	number of instances	number of attributes
Liver disorders(Liv)	345	7
Heart (Hea)	270	13
Ionosphere (Ion)	351	34
Breast cancer (Bre)	699	10
Tic tac toe (Tic)	958	9

Arc-x(j) results: All Arc-x(j) algorithms decrease the test error. The relative test error reduction is higher than 15% for all datasets except when Arc-x(j) algorithms are applied for 25 iterations on the tic tac toe dataset. Results produced for 75 iterations are better than those obtained for 25 iterations.

Comparing algorithms: When comparing the results of the different boosting algorithms for 25 and 75 iterations, we notice that:

- For 25 iterations, the lowest average test error is produced by Arc-x4 algorithm.
- The relative average error reduction between Arc-x4 and Adaboost is 0.43% which is not significant.
- The average error of Arc-x4 is better than the average error of Arc-x8 by 5.62% and by 8.96% for Arc-x12 which are significant at 5% level.
- Arc-x4 and Adaboost produce the lowest error on 2 databases, Arc-x8 outperforms the other algorithms on 1 data set.
- For 75 iterations, Arc-x4 outperforms all other algorithms.
- Adaboost and Arc-x12 performs equally well and less accurately than Adaboost and Arc-x8.
- Arc-x4 produces the lowest error on 4 data sets and Arc-x12 on 1 data set.
- The relative average error reduction between the lowest and the highest error is 2.55% which is not significant.

4 Conclusion

This empirical study is an extension to Breiman's (1998) study on the family of boosting algorithms, the ARCING family. Two extensions of arcing weight updating rules are tested and compared to the one used by Breiman (1998) and to Adaboost in the subsampling framework.

Our empirical study shows that, based on these empirical results, increasing the factor j of Arc-x(j) algorithm does not improve the performance of

Arcing algorithms. The absolute test error reduction is higher for the first 25 iterations than for the last 50 iterations. It is interesting to look for another way of combining classifiers which gives more weight to the first ones and thus producing lower test error.

References

BAUER, E. KOHAVI, R. (1999): An empirical comparison of voting classification algorithms: Bagging, boosting and variants.*Machine learning, 36(1), 105–142.*

BLAKES, C. KEOGH and E. MERZ, C.J. (1998): UCI repository of machine learning databases. *http://www.ics.uci.edu/ mlearn/MLRepository.html*

BREIMAN, L. (1998): Arcing classifiers.*The Annals of Statistics, 26(3), 801–849.*

FRIEDMAN, J., HASTIE, T. and TIBSHIRANI, R. (2000): Additive logistic regression: A statistical view of boosting. *The Annals of Statistics, 28(2), 337-407.*

FREUND, Y. (1995): Boosting a weak learning algorithm by majority. *Information and Computation, 121(2), 256–285.*

FREUND, Y. and SCHAPIRE, R.E. (1996): Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference, 148–156.*

FREUND, Y. and SCHAPIRE, R.E. (1997): A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55(1), 119–139.*

IBA, W. and LANGLEY, P. (1992): Induction of one-level decision trees. *Proceedings the ninth international conference on machine learning*, 233–240.

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

VAPNIK, V. (1998): *Statistical learning theory.* John Wiley & Sons INC., New York. A Wiley-Interscience Publication.