Empirical comparison of Arcing algorithms

Riadh Khanchel and Mohamed Limam

I . S . G
41, Avenue de la Liberté
Le Bardo,2000 Tunis, Tunisie
(e-mail: riadh.khanchel@isg.rnu.tn)

Abstract. Adaboost and Arc-x(h) are two ensemble algorithms that belong to Arcing family of algorithms. They use different weight updating rules and combine classifiers using different voting scheme. For $h = 4$, Arc-x(h) performs equally well as Adaboost but higher values of h were not tested. Previous methods used to compare algorithms are based on the performance over test sets. A different approach presented by [Nadeau and Bengio, 2003] takes into account variability in training and test sets. Using this approach, an empirical study is conducted to compare Adaboost and Arc-x(h) for different values of h. Results show that increasing h does not affect the performance of Arc-x(h) whichis comparable to Adaboost.
Keywords: Boosting, Arcing, Adaboost.

1 Introduction

Different classification algorithms have been proposed and used in fields like medicine, business and finance. However, the accuracy of these algorithms may be moderate when applied to complex classification tasks. Ensemble learning is a technique for improving their performance: a collection of moderately accurate and diverse classifiers are constructed then they are combined in order to output highly accurate ones. Different ensemble learning algorithms have been proposed: Adaboost [Freund and Schapire, 1997], Bagging [Breiman, 1996], and Arcing [Breiman, 1998].
Ensemble learning method is developed within the framework of probably approximately correct (P. A. C) model of learning where learning algorithm and hypothesis are used to refer to, respectively, classification algorithm and classifier. This model of learning is specified by a set of measurement space, a label space, an error parameter ϵ, a confidence parameter δ and other parameters that specify the size of the measurement space and the label space. After running for a polynomial time, the learning algorithm outputs a hypothesis which error is less than ϵ with probability $1 - \delta$: this is a P.A.C. hypothesis.
Two extensions to this learning model are strong and weak learning algorithms. Both algorithms run in a polynomial time. The strong learning algorithm outputs a hypothesis that is P.A.C while the weak one outputs a hypothesis with accuracy better than 0.5. The question of whether these two

notions are equivalent is referred to as the "hypothesis boosting problem" since in order to show this equivalence we must boost the accuracy of the weak learning algorithm.
The proof that these notions are equivalent is provided by [Schapire, 1990], who presents the first algorithm for converting a weak learning algorithm into a strong one. Based on this idea, Boost-By-Majority, a simpler and more efficient boosting algorithm, is developed by [Freund, 1995]. This algorithm suffers from practical problem for estimating parameters. Adaboost, the first adaptive boosting algorithm, is developed by [Freund and Schapire, 1997]. This algorithm does not require parameter estimation.
Arcing family of algorithms denotes algorithms that **A**daptively **R**esample data and **C**ombine classifiers [Breiman, 1998]. Adaboost belongs to this family. Arc-x(h) is an ad-hoc algorithm developed by [Breiman, 1998] to better understand the behaviour od Adaboost. This algorithm uses a simple weight updating rule and a different method for combining hypotheses. For $h = 4$ Arc-x(h) performs better than $h = 1$ or 2.When compared to Adaboost, Arc-x4 performs equally well. However Breiman argues that higher values for h are not tested so further improvement is possible [Breiman, 1998].
In this paper, different values for the parameter h of Arc-x(h) algorithm are tested and their performance are compared to Adaboost and Arc-x4 in the reweighting framework using C4.5 [Quinlan, 1993] as classification algorithm.
In section two, Adaboost and Arc-x(h) are introduced then results of previous empirical study are reviewed. In section three, the general framework of this empirical study is presented: classification and boosting algorithms, datasets and performance measure. Results are presented in section four. Finally, section five provides a conclusion to this work.

2 Arcing Algorithms

Adaboost and Arc-x(h) belongs to the Arcing family of algorithms. In this section, these algorithms are presented then results of previous empirical studies are reviewed.

2.1 Adaboost

Adaboost applies a classification algorithm to a dataset composed with labelled instances for a fixed number of iterations T. In each iteration t, $t = 1, \ldots, T$, a weight, $w_t(Z_i)$, is assigned to each instance $Z_i = (x_i, y_i), i = 1, \ldots, n$ in the dataset. It represents instance's importance. Based on this weight distribution, a classifier is outputted which predicts the class of each instance. Adaboost requires that the weighted error is less than 0.5. A parameter α_t is used to update the weights and to measure classifier's performance. The weight of misclassified instances is increased in order to force the algorithm to concentrate on them in the next iteration.

At the end of the process, a final classifier is obtained by combining classifiers from previous iterations using weighted majority vote. The parameter α_t represents the weight of classifier h_t generated in iteration t. The pseudocode of Adaboost for binary classification is presented in table 1.

Algorithm:Adaboost algorithm
Given: $\{Z_1 = (x_1, y_1), \ldots, Z_n = (x_n, y_n)\}$ where $x_i \in X, y_i \in Y = \{-1; +1\}$
1-Initialize $w_1(Z_i) = 1/n$ for $i = 1, \ldots, n$.
2-For t = 1 to T:
•Train the algorithm using w_t and get a classifier $h_t : X \mapsto \{-1; +1\}$
• Compute $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} w_t(Z_i)$
• If $\epsilon_t \geq 0.5$ stop.
• Choose: $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$
• Update: $w_{t+1}(Z_i) = \frac{w_t(Z_i) \exp(-\alpha_t y_i h_t(x_i))}{N_t}$
where N_t is a normalization factor
3-output the final hypothesis: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Table 1. Adaboost algorithm for binary classification

2.2 Arc-x(j)

Arc-x(h) algorithm is developed by [Breiman, 1998] to study the behaviour of Adaboost. It is different from Adaboost in the following:

- it uses a simpler weight updating rule:

$$w_{t+1}(Z_i) = \frac{1 + m(Z_i)^h}{\sum(1 + m(Z_i)^h)}, \tag{1}$$

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

2.3 Previous results

Empirical results show that Adaboost improves the performance of various classification algorithms, often by dramatic amount. Adaboost decreases the average error rate by 55.2% when applied to decision stump, a weak learning algorithm [Freund and Schapire, 1996]. Boosted decision stump performs equally well as C4.5 [Quinlan, 1993], a strong learning algorithm: Adaboost converts a weak learning algorithm into a strong one.
The ability of Adaboost to improve strong learning algorithm is investigated by [Freund and Schapire, 1996] and [Quinlan, 1996]. Experimental results show that Adaboost improves the average error rate.
Arc-x4 is tested on moderate and large data sets by [Breiman, 1998]. Results show that it improves the performance of CART [Breiman *et al.*, 1984] learning algorithm for all data sets.
Two different frameworks are considered by [Bauer and Kohavi, 1999] to test the performance Arc-x4: reweighting and subsampling. Subsampling uses the weight of instances to generate a different training set in each iteration while reweighting uses a fixed training set for all iterations. Arc-x4 produces a higher error reduction in the subsampling framework than in the reweighting framework.
Adaboost and Arc-x4 are compared in different framework and using different collections of datasets. Arc-x4 has an accuracy comparable to Adaboost without using the weighting scheme to construct the final classifier [Breiman, 1998] and [Bauer and Kohavi, 1999].
Arc-x(h) is tested for $h = 1, 2, 4$ by [Breiman, 1998]. However higher values of h are not tested so improvement is possible. Based on the performance measure used by [Bauer and Kohavi, 1999], increasing the factor h does not improve the performance of Arc-x(h) [Khanchel and Limam, to appear].

3 Empirical Study

In this section, the general framework of our empirical study in presented: classification algorithm, Arcing algorithms and data sets. The performance measure criterion is presented. Then performance of the different algorithms is compared.

3.1 General framework

C4.5 [Quinlan, 1993] is used as subroutine for the different boosting algorithms. In order to compare different boosting algorithms, a collection of data sets from UCI Machine learning Repository [Keogh and Merz, 1998] is used. Details of these data sets are presented in table 2.

Different values of the parameter h, $h \in \{5, 6, 8, 12\}$, are tested for the algorithm Arc-x(h). Results are compared to Adaboost and Arc-x4 in the reweighting framework . Boosting algorithms are applied for 25 iterations.

Data set	number of instances	number of attributes	number of classes
Liver disorders	345	7	2
Heart	270	13	2
Australian	690	14	2
Pima	760	8	2

Table 2. Data sets used in the empirical study

3.2 Performance measure

The performance boosting algorithms is usually evaluated using test error. This criteria takes into account only variability due to the choice of the test set. Comparison is often made without using rigorous statistical framework [Nadeau and Bengio, 2003]. Often it uses liberal estimators and leads to incorrect claims. A new method which takes into account variability due to the choice of training sets and test sets is presented by [Nadeau and Bengio, 2003]. The goal is to estimate the generalization error using the training data.

Given a data set Z^n of size n, a training set of size n_1 is generated from this data set. Using Z^{n_1} a classifier is generated. The loss incurred by this classifier on a new example Z_{n+1} can be expressed by $L(Z^{n_1}; Z_{n+1})$. We are interested in estimating ${}_{n_1}\mu = E[L(Z^{n_1}, Z_{n+1}]$.

To achieve this, we proceed as follows: suppose that the data set Z^n is composed with n labelled instances $Z^n = \{Z_1, \ldots, Z_n\}$. For $m = 1, \ldots, M$, Z^n is randomly splitted into 2 distinct subsets D_m and D_m^c each of size $n/2$. For each subset, we repeat the following process for $j = 1, \ldots, J$:

- Let S_j be a set of random index of size $n_1, n_1 = 4n/10$, chosen from $\{1, \ldots, n/2\}$ and let S_j^c of size $n_2 = n/10$ denote its complement.
- Let $Z_j = \{Z_i / i \in S_j\}$ be the training set and $Z_j^c = \{Z_i / i \in S_j^c\}$ be the test set.
- For $j = 1, \ldots, J$, use Z_j to generate a classifier, and let $L(j, i)$ be:

$$L(j, i) = Q_A(Z_j, i) - Q_B(Z_j, i) \tag{2}$$

  where Q_A (Q_B) is the loss observed on instance i when the algorithm A (B) uses Z_j to generate classifiers.
  For classification problem, this loss can be expressed as:

$$Q_A(Z_j, i) = \begin{cases} 1 \text{ if instance } i \text{ is incorrectly classified,} \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

- First we average over the test set Z_j^c of size n_2 to obtain:

$$\hat{\mu}_j = \frac{1}{n_2} \sum_{i \in S_j^c} L(j, i). \tag{4}$$

- Then we average over J to obtain:

$$ {}^{n_2}_{n_1}\hat{\mu}_J = \frac{1}{J}\sum_{j=1}^{J}\hat{\mu}_j \tag{5} $$

This process is repeated for D_m^c and for $m = 1, \ldots, M$. Each of the M split yields a pair $({}^{n_2}_{n_1}\hat{\mu}_J, {}^{n_2}_{n_1}\hat{\mu}_J^c)$ which can be denote as $(\hat{\mu}_m, \hat{\mu}_m^c)$.
The generalization error ${}_{n_1}\mu$ is estimated using ${}^{n_2}_{n_1}\hat{\mu}_J$ and its variance is estimated using:

$$ {}^{n_2}_{n_1}\hat{\sigma}_J^2 = \frac{1}{2M}\sum_{m=1}^{M}(\hat{\mu}_m - \hat{\mu}_m^c)^2. \tag{6} $$

Since ${}^{n_2}_{n_1}\hat{\mu}_J$ is the mean of Jn_2 loss $L(j,i)$, its distribution can be approximated by the normal distribution:

$$ \frac{{}^{n_2}_{n_1}\hat{\mu}_J - {}_{n_1}\mu}{\sqrt{{}^{n_2}_{n_1}\hat{\sigma}_J^2}}. \tag{7} $$

Using this assumption we can perform inference about the performance of boosting algorithms using confidence interval. A confidence interval for ${}_{n_1}\mu$ at confidence level $1-\alpha$ has the following form:

$$ {}_{n_1}\mu \in [{}^{n_2}_{n_1}\hat{\mu}_J - c\sqrt{{}^{n_2}_{n_1}\hat{\sigma}_J^2} \quad , \quad {}^{n_2}_{n_1}\hat{\mu}_J + c\sqrt{{}^{n_2}_{n_1}\hat{\sigma}_J^2}] \tag{8} $$

where c is a percentile from $N(0,1)$ distribution.

4 Results

For each pair of algorithms and for each dataset we construct a confidence interval at confidence level 95%. If this interval includes zero, we conclude that both algorithms have comparable performance. Confidence interval are presented in table 3. Algorithms producing the same error rate are omitted.
The important observations for this empirical comparison are:

- For 3 data sets: Pima, Heart and Australian, Arc-x(h) outputs the same test error in all iterations and for different values of the parameter h. When compared to Adaboost, the confidence interval is:
  - $[-0.1197, 0.0371]$ for the Australian data and $[-0.0689, 0.1775]$ for the heart data. For these 2 data sets, we conclude that all algorithms have comparable performance.
  - $[-0.0365, -0.0073]$ for Pima data. Adaboost performs slightly better that Arc-x(h) algorithms

data sets	algorithms compared	confidence intervals
Australian	Arc-x(h) - Adaboost,$h \in \{4, 5, 6, 8, 12\}$	[-0.1197, 0.0371]
Heart	Arc-x(h) - Adaboost, $h \in \{4, 5, 6, 8, 12\}$	[-0.0688, 0.1775]
Pima (diabetes)	Arc-x(h) - Adaboost, $h \in \{4, 5, 6, 8, 12\}$	[-0.0365, -0.0073]
Puba (liver disorder)	Arc-x(h)- Arc-x4, $h \in \{5, 6, 8, 12\}$	[-0.0440, 0.0990]
	Arc-x4 - Adaboost	[-0.0976, 0.0214]
	Arc-x(h) - Adaboost	[-0.0227, 0.0014]

Table 3. Confidence intervals for difference of generalization error for different Arcing Algorithms

- For Bupa data, Arc-x(h) outputs the same test error in all iterations for $h = 5, 6, 8$ and12. Arc-x4 outputs a slightly lower test error. This difference is not significant because the confidence interval at 95% confidence level is $[-0.044, 0.099]$. Adaboost has a comparable performance to the different arc-x(h) algorithm: the confidence interval when compared to Arc-x4 is [-0.0976, 0.0214] and [-0.0227, 0.0014] when compared to the other Arc-x(h) algorithms.

5 conclusion

This empirical study is an extension to Breiman's study [Breiman, 1998] of the family of Arcing algorithms. Different values of the parameter h used by Arc-x(h) algorithm in the weight updating rule are tested and compared to Adaboost in the reweighting framework. The approach proposed by [Nadeau and Bengio, 2003] is adopted: performance measures take into account variability due to the training sets and test sets and comparisons are made using confidence intervals.
Based on this empirical study, increasing the factor h used by Arc-x(h) in the weight updating rule does not improve performance. Arc-x(h) performs equally as Adaboost for different values of h. Adaboost performs slightly better for only one data set.
Comparable performance is obtained using two different methods for combining classifiers. This agree with Breiman's claim that the error reduction is due to the weight updating rule.
The size of the data sets used in this empirical study is moderate. The framework proposed by [Nadeau and Bengio, 2003] uses small fractions of these data sets as training and test sets. Also the process generates many training data then averages the performance. This can explain the comparable performance of the different boosting algorithm considered. It will be interesting to test these algorithms on large data sets where large training and test sets can be generated.

References

[Bauer and Kohavi, 1999]E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithm: Bagging, boosting and variants. *Machine Learning*, pages 105–142, 1999.

[Breiman *et al.*, 1984]L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Chapman anf Hall, London, 1984.

[Breiman, 1996]L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[Breiman, 1998]L. Breiman. Arcing classifiers. *The annals of statistics*, 26(3):801–849, 1998.

[Freund and Schapire, 1996]Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *machine learning: Proceedings of the thirteenth international conference*, pages 148–156. Morgan Kaufmann San Francisco, 1996.

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

[Freund, 1995]Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 12(2):256–285, 1995.

[Keogh and Merz, 1998]C. Blakes E. Keogh and C.J. Merz. Uci repository of machine learning databases http://www.ics.uci.edu/ mlearn/ mlrepository.html. 1998.

[Khanchel and Limam, to appear]R. Khanchel and M. Limam. Empirical comparison of boosting algorithms. Springer-Verlag, to appear.

[Nadeau and Bengio, 2003]C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(2):239–281, 2003.

[Quinlan, 1993]J.R. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann, 1993.

[Quinlan, 1996]J.R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730, Menlo Park, August4–8 1996. AAAI Press / MIT Press.

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