Review of Arcing Classifiers

Nusrat Jahan
Department of Mathematics and Statistics
Mississippi State University,Mississippi State, MS 39762 USA
email:njahan@ra.msstate.edu

Abstract

Some classification algorithms such as regression trees, neural networks are very sensitive to their training sets [2]. A minor change or perturbations may produce a significant change to the constructed predictor. In these cases of unstable classifiers, accuracy can be improved by aggregating multiple versions of the predictor generated by perturbing the original training set. Bagging and arcing (or boosting) are two well known methods for improving accuracy in terms of reduced test set error. In case of bagging, different versions of the classifiers are constructed from the original training set by resampling and then these multiple classifiers are combined by simple voting. In case of arcing, multiple classifiers are generated by adaptive resampling of the training set and then they are combined by weighted voting. Arcing algorithms have been found to be very efficient in reducing the test set error compared to bagging [2]. Two types of *arcing* algorithms are investigated in this paper. They are compared with each other and also with bagging. The error reduction ability of arcing has been attributed to the adaptive resampling technique.

1 Introduction

A classification algorithm operates on a training set to produce a classification rule that will provide low prediction error or low misclassification rate on a separate test set. If the algorithm is unstable, a small change either in the training set or in construction may inflate the prediction error [1]. The performances of unstable algorithms can be improved by bagging or boosting methods [2]. Both of these methods work with the purturbed versions of the training set repeatedly to produce multiple predictors and these are combined by a majority vote [3]. Bagging and boosting employ distinctly different methods to introduce perturbation in their training sets. Perturbations introduced by bagging are random and independent, where as the perturbations used by boosting are deterministic and serial (dependent on the previous step) [4]. Bagging uses bootstrap techniques to resample from the training set. Then classifiers are built from each sample and combined. In case of boosting adaptive resampling is used to resample from the training set. Classifiers are constructed from each sample, then they are combined. Because of the technique of "adaptive resampling and combining" boosting algorithm is referred to as "arcing algorithm"

Both bagging and arcing algorithms when applied to any unstable classification algorithm, improves efficiency of that algorithm, reduces variance. If they are compared with each other, in most cases arcing algorithm out performs bagging. Arcing algorithm was first introduced by Freund and Schapire [3]. Their particular algorithm is identified by arc-fs. It is found to produce very low variance on both synthetic and real world data sets [2]. The author in [2] claims that the variance reduction by arc-fs is due to the adaptive resampling and not because of the specific form of arc-fs. To prove his claim, Breiman

proposed arc-x4 - a simpler form of arc algorithm in [2]. The performacnces of arc-fs and arc-x4 are compared on several different simulated and real data sets [2]. They have been found to produce equivalent results in most cases.

In this article, section 2 presents how arc-fs and arc-x4 work and also their differences are highlighted. Section 3 discusses and compares the results from simulated and real data sets obtained in [2] for arc-fs and arc-x4. An overall summarization section ends this paper.

2 Arcing Algorithms

Bagging and arcing (boosting) techniques manipulate the training set to generate diverse classifiers. Then these classifiers are combined to produce the finel improved classifier. In this section procedures bagging, arc-fs and arc-x4 are explored.

Suppose the training set T has N cases, $n = 1, 2, ..., N$ with equal probabilities $p(n) = 1/N$ assigned to each case, the probability distribution of T given by $p(n)$ is P. Bagging would use these probabilities to sample with replacement (bootstrap technique) N times from T. This resampled training set ($T^{`}$) is iid from P, some cases in T may not appear in $T^{`}$ and others may appear more than once. The whole procedure is repeated to produce a sequence of independent bootstap training sets and from each set a classification predictor C_k is constructed by applying the same classification algorithm. Then these classifiers vote for each class. The constructed predictors are combined to produce the final prediction rule [2]. For any point x, the classifier $C_k(x)$ depends on the underlying probability distribution P_0 that the training sets are drawn from $C_k(x) = C_k(x, P_0)$. The classifier resulting from bagging is $C_k(x, P)$, which is an approximation to $C_k(x, P_0)$ [2]. This fact is the cause of variance reduction in bagging classfiers.

In case of arcing, $p(n) = 1/N$ is used to resample from T (the original training set) N times and a classification predictor is constructed from there. As the procedure is repeated, $p(n)$ is updated depending on the misclassifications up to that step [2]. $p(n)$ increases for most frequently misclassified cases. At the final stage classifiers are combined by weighted or simple voting. In [2], the author discussed two arcing algorithms namely arc-fs and arc-x4. The arc-fs algorithm is developed based on a boosting theorem given in [S], where as arc-x4 is an ad hoc invention [2]. The initial steps are the same as described above for both the arc-fs and arc-x4. They differ interms of how the probabilities are updated at each step and how the different classifiers are combined.

Arc − fs

At any step k the classifier C_k, constructed from the resampled training set is used to classify the cases of T (original training set). In [2] the author defined $d(n) = 1$ if the nth case is misclassified otherwise zero. Then

$$\epsilon_k = \sum_n p^{(k)}(n) d(n), \quad \beta_k = (1 - \epsilon_k)/\epsilon_k.$$

Therefore the updated probabilities for the next step is given by,

$$p^{(k+1)}(n) = p^{(k)}(n) \beta_k^{d(n)} / \sum_n p^{(k)}(n) \beta_k^{d(n)}.$$

After a predetermined number of steps, the $C_1, ..., C_k$ are combined using weighted voting, where the classifier C_k has the weight $log(\beta_k)$ [S]. It has been observed that if ϵ_k becomes greater than 1/2, the voting weights β_k becomes negative. In that case, setting all $p(n)$ equal and restarting the algorithm would yield better results [2]. Also if ϵ_k becomes zero the step should be made undefined, again setting all $p(n)$ equal and restarting would yield better results [2].

Arc − x4

The classifier C_k is used to classify T. At step k, $m(n)$ is defined as the number of misclassifications of the nth case by $C_1, ..., C_k$. the updated probabilities for the next step is given by,

$$p(n) = \left(1 + m(n)^4\right) / \sum \left(1 + m(n)^4\right).$$

After a fixed number of steps, the classifiers $C_1, ..., C_k$ are combined by unweighted voting [2].

If the three classification algorithms described above are compared, we see that all of them are methods of manipulating the training set to generate multiple classifiers. Just the manipulation techniques and the techniques of combining the multiple classifiers differ for each algorithm.

3 Discussion

This section discusses the performance of bagging and two arc algorithms over a number of data sets as described by [2]. The algorithms were run on both artificial and real data sets. Four artificial data sets were used, for each one 100 replicate training sets of size 300 were generated by Monte Carlo method. On each training set bagging and arc-fs were used 50 times with CART [3] as the classifier. This paper [2] analyzes the test set error in terms of bias and variance decomposition. Both methods reduced bias a little bit. But the performance of arcing was better than that of bagging in terms of variance reduction. As the number of combined classifiers was increased from 50, the arc-fs error rate decreased significantly and came close to Bayes minimum [6] (with 250 combined classifiers) [2].

In [2] six moderate and four larger real data sets were used to compare bagging, arc-fs and arc-x4. In addition to these another handwritten digit data set "U.S. Postal Service" was also used.

For the moderate sized data sets, 10% of the training sets was randomly selected to be used as a test set. Both arc-fs and arc-x4 were run 50 steps on the remaining 90% of the training sets. Then these 50 classifiers were combined to produce the final classifier. For the four larger data set and also the digit data, separate training sets and test sets were provided. Again each of the arcing algorithms was used to construct 50 classifiers and then these were combined to produce the final classifier. In case of the digit data 100 classifiers were used. When the test set errors were compared for the three algorithms bagging had the highest percentage of test set errors for almost all the data sets. Both arc-fs and arc-x4 were comparable in test set error percentages. It appeared that arc-x4 performed slightly better in smaller data sets and arc-fs performed a little better in larger data sets.

In order to understand arcing better, further experiments were conducted on the six moderate sized data sets and also on the artificial waveform data [2]. One thousand runs were made for both arc-fs and arc-x4 in each case. It has been observed that these two arcing algorithms were very different in terms of arcing techniques. In arc-fs, the constructed trees changed considerably from one step to the next, where as in case of arc-x4 changes were more gradual [2]. In these seven data sets, arc-x4 used 35% to 60% of the training set data, where as arc-fs used relatively smaller fraction of the data (lowest was 13% on breast cancer data). For arc-fs, the larger the value of $p(n)$ is, the more variable it is (standard deviations of $p(n)$ are large and increase linearly with the average), where as standard deviations for arc-x4 are quite small and increase slowly with average $p(n)$ [2]. It should also be noted that the range of $p(n)$ for arc-fs is 2 to 3 times greater than that of arc-x4. For both algorithms, the more frequently a case is misclassified, the more its probability of selection increased and it becomes more likely to be used in a training set.

Arc-fs reduces the training set error to zero very rapidly (on average at most five tree constructions), but it does not do the same for test set error [2]. In fact the accompanying test set error is higher than that of bagging. In order to obtain optimum reductions in test set error, arc-fs must continue to run beyond the point of zero training set error.

In case of arcing, instead of randomly sampling from the probabilities $p(n)$, the weighted probabilities for selection of each case are computed, so randomness is not involved. But in case of bagging, random sampling is critical, the selection probabilities for each case must remain equal and constant. It has also been observed

that arcing takes longer to attain its minimum error rate compared to bagging [2]. Breiman [2] also suggested that if error reduction properties of arcing are related to its steady-state behavior, then the longer reduction time may be due to the fact that the dependent Marcov property of the arc-fs algorithm takes longer to reach steady state than bagging. In case of bagging because of the i.i.d. bootstrap training sets, steady-state is achieved sooner (because of law of large numbers). In [7], arcing was compared with C4.5 (a tree-structured program [8] similar to CART), the arc-fs test set error was 20% larger than that of C4.5.

The effectiveness of bagging or arcing appears to be applicable only in case of unstable classification methods. It has been shown in [2] that, linear discriminant analysis - a stable low variance procedure which fits a simple normal parametric model, does not improve much with either bagging or arcing. Niether of the procedures can reduce the linear discriminant analysis test set error.

4 Summary

Breiman's work in [2] shows the effectiveness of aggregating classifiers for reducing classification error. Both bagging and arcing are methods of manipulating the original training set to generate diverse classifiers. From these classifiers simple or weighted voting will yield a better classifier, in the sense that the combined classifier will produce greater reduction in test set error. But this strategy works for only unstable classification algorithms. If an algorithm is stable, it will not change much with replication of training sets. The same cases tend to be misclassified even with the changing training sets.

The arc-fs and other arcing algorithms are innovative and highly efficient methods for reducing test set error for unstable classification algorithms. For a number of classification methods, arc-fs and other arcing algorithms can reduce the test set error to the point of most accurate. Arcing algorithms particularly arc-fs is very easy to use. Though there are some concerns about the error reduction properties of arc-fs, it still came out as the top error reducer in almost every data set [2]. The introduction of arc-x4 and its comparable performance to arc-fs indicates that further improvement of arcing technique is possible. This also supports the author's contention that error reduction property of arc-fs does not depend on the specific form of arc-fs, rather it depends on the adaptive resampling technique. There exists a huge scope of improvement and research in the arcing method.

References

[1] Breiman,L.Bagging Predictors. Machine Learning,Vol.26,pp.123-140,1996.

[2] Breiman,L.Arcing Classifiers.The Annals of Statistics,Vol.26,No.3,pp.801-824,1998.

[3] Breiman,L.,Friedman,J.,Olshen,R. and Stone,C.Classification and Regression Trees.Chapman and Hall,London,1984.

[4] Freund,Y. and Schapire,R.Experiments with a new boosting algorithm. In Machine Learning.Proceedings of the 13th International Conference. Morgan Kaufmann,San Francisco,1996.

[5] Freund,Y. and Schapire,R.A descision-theoretic generalization of on-line learning and an application to boosting.J.Comput. System Sci.Vol.55,pp.119-139,1997.

[6] Mackay,D.A practical bayesian framework for backpropagation networks. Neural Computation,Vol.4,pp.448-472,1992.

[7] Quinlan,J.R.Bagging,Boosting, and C4.5. In proceedings of AAAI'96 National conference on Artificial Intelligence. pp.725-730,1996.

[8] Quinlan,J.R. C4.5:Programs for Machine Learning. Morgan Kaufmann,San Francisco, 1993.