BOOSTED DECISION TREES, A POWERFUL EVENT CLASSIFIER

BYRON P. ROE^A, HAI-JUN YANG^A, AND JI ZHU^B
^{A} *Department of Physics,* ^{B} *Department of Statistics, University of Michigan,*
450 Church St., Ann Arbor, MI, 48109-1040
e-mail: byronroe@umich.edu

Boosted decision trees are compared with neural nets and various decision tree methods using the MiniBooNE experiment as a test bed. A discussion of methods for pruning variables and for increasing the speed of convergence are given.

1 Decision Trees and Boosting

Consider the problem of classification of events between signal and background, given a number of particle identification (PID) variables. A decision tree is a sequence of binary splits of the data. To train the tree a set of known training events is used. The results are measured using a separate set of known testing events. Consider all of the data to be on one node. The best PID variable and best place on that variable to split the data into separate signal and background is found. There are then two nodes. The process is repeated on these new nodes and is continued until a given number of final nodes (called "leaves") are obtained, or until all leaves are pure or until a node has too few events.

There are several popular criteria to determine the best PID variable and best place on which to split a node. The *gini* criterion is used here. Suppose that event i has weight W_i. The purity P of a node is defined as the weight of signal events on the node divided by the total weight of events on that node. For a given node: $gini = P(1-P)\sum_i W_i$. $gini$ is zero for $P=1$ or $P=0$. The best split is chosen as the one which minimizes $gini_{left} + gini_{right}$. The next node to split is chosen by finding that node whose splitting maximizes the change in $gini$. In this way a decision tree is built. Leaves with $P \geq 0.5$ are signal leaves and the rest are background leaves.

Decision trees are powerful, but unstable. A small change in the training data can produce a large change in the tree. This is remedied by the use of boosting. For boosting, the training events which were misclassified (a signal event fell on a background leaf or vice versa) have their weights increased (boosted), and a new tree is formed. This procedure is then repeated for the new tree. In this way many trees are built up. The score from the mth individual tree T_m is taken as $+1$ if the event falls on a signal leaf and -1 if the event falls on a background leaf. The final score is taken as a weighted sum of the scores of the individual leaves.

Two methods for boosting are considered here. The first is called AdaBoost. Define err_m = *weight misclassified/total weight for tree m.* Let $\alpha_m = \beta \log\left[(1-err_m)/err_m\right]$, where β is a constant. In the statistical literature β has been taken as one, but for the MiniBooNE experiment, $\beta = 0.5$ has been found to be the optimum value. The misclassified events have their weight multiplied by e^{α_m}. The weights are then renormalized so the sum of all of the training event weights is one. The final score is $T = \sum_{m=1}^{N_{tree}} \alpha_m T_m$.

The second method of boosting considered here is called ϵ-boost or, sometimes, shrinkage. Misclassified events have their weight multiplied by $e^{2\epsilon}$, where ϵ is a constant. For the MiniBooNE experiment, $\epsilon = 0.03$ has been optimum. (The results vary only mildly as β or ϵ are changed a bit.) The final score is $T = \sum_{m=1}^{N_{tree}} \epsilon T_m$.

ϵ-boost changes weights a little at a time, while AdaBoost can be shown to try to optimize each change in weights to minimize e^{-yT} where T is the score and y is $+1$ for a signal event and -1 for a background event. The optimum value is $T = \log prob/(1-prob)$, where $prob$ is the probability that $y=1$, given the observed PID variables. In practice, for MiniBooNE, the two boosting methods have performed almost equally well. Boosting is described as using many weak classifiers to build a strong classifier. This is seen in Figure 1. After the first few trees, the misclassification fraction for an individual tree is above 40%.

In the MiniBooNE experiment some hundreds of possible PID variables have been suggested. The most powerful of these have been selected by accept-

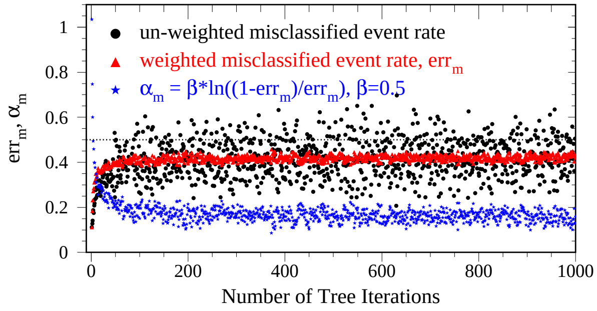


Figure 1. The unweighted, weighted misclassified event rate (err_m), and α_m versus the number of tree iterations for AdaBoost with $\beta = 0.5$ and signal purity threshold value of 50%.

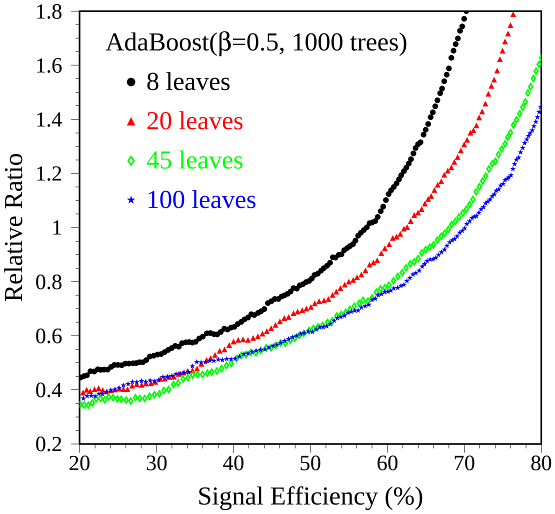
ing those which are used most often as splitting variables. Some care needs to be taken as sometimes a variable will appear unimportant for the first few trees, but then become important for later trees. Current MiniBooNE boosting trees have 80-100 PID variables. Use of more variables tends to slightly degrade the performance, probably because all of the useful information is already in the previous variables and noise without additional signal is being added. The performance has been examined varying the number of trees and the number of leaves/tree. This is shown in Figures 2. Here, relative ratio is *constant* × *fraction of background kept/fraction of signal kept* for a given signal efficiency. (Smaller is better!) Optimum results are obtained for MiniBooNE with about 1000 trees and with 45 leaves/tree. Different experiments should optimize these values for their particular data sets.

2 Tests of Boosting with Other Classification Methods

Boosting was compared with artificial neural nets (ANN), which the MiniBooNE collaboration had used previously. For Figure 3 only, the relative ratio is redefined as the fraction of background kept by ANN to that for boosting for a given fraction of signal events being kept. (Larger is better for boosting!) It is seen that boosting is better than ANN by a factor of 1.2-2 for MiniBooNE data.

AdaBoost and ϵ-boost were compared with various other similar methods. Space does not permit a description of these methods; Table 1 will be of most use to those already familiar with them.

It is seen that Adaboost, ϵ-boost, ϵ-LogitBoost,



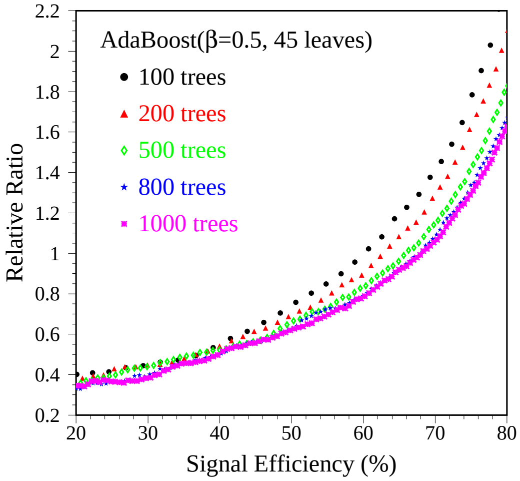


Figure 2. Top: tuning the number of leaves when using 1000 trees. Bottom: tuning the number of trees when using 45 leaves.

Boosting Algorithms	Parameters β,ϵ (N_{lv}, N_{tr})	Rel. ratios 50% sig. eff.
AdaBoost	0.5 (45,1000)	0.62
AdaBoost	0.8 (45,1000)	0.62
ϵ-Boost	0.03 (45,1000)	0.58
AdaBoost (b=0.5)	0.5 (45,1000)	0.60
ϵ-Boost (b=0.5)	0.03 (45,1000)	0.58
ϵ-LogitBoost	0.01 (45,1000)	0.61
ϵ-HingeBoost	0.01 (30,1000)	0.86
LogitBoost	0.1 (45,150)	0.62
Real AdaBoost	(45,1000)	0.69
Gentle AdaBoost	(45,1000)	0.67
Random Forests(RF)	(400,1000)	0.85
AdaBoosted RF	0.5 (100,1000)	0.66

Table 1. Relative error ratio versus signal efficiency for various boosting algorithms using MiniBooNE data. Differences up to about 0.03 are largely statistical. b=0.5 means the smooth scoring function described in Section 3.

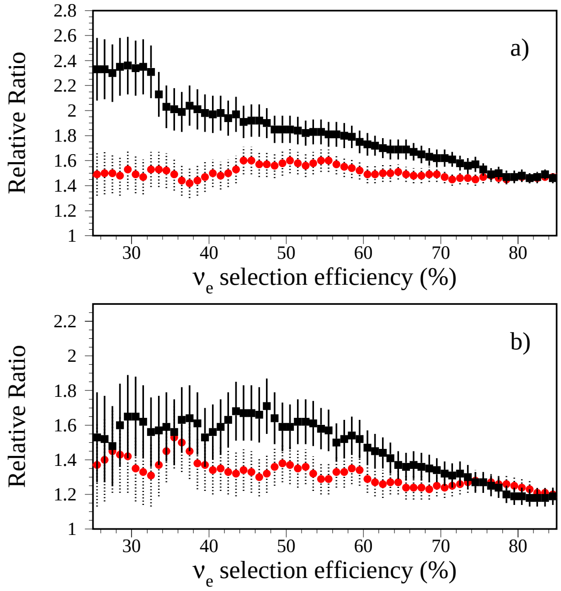


Figure 3. Comparison of ANN and AdaBoost performance for test samples. Relative ratio (defined as the number of background events kept for ANN divided by the number of background events kept for AdaBoost) versus the intrinsic ν_e charged current quasi-elastic event selection efficiency. a) All kinds of backgrounds are combined for the training against the signal. Dots show the relative ratios for 21 training variables and boxes show them for 52 training variables. b) AdaBoost trained by signal and neutral current π^0 background. Dots show the relative ratios for 22 training variables and boxes show them for 52 training variables. All error bars shown in the figures are for Monte Carlo statistical errors only.

and LogitBoost performed similarly. The Random Forest method uses no boosting, but uses a random fraction of the training events, chosen with replacement, for each tree and a random fraction of the PID variables for each node. For the tests in Table 1, all of the PID variables were used in each node. This option is also known as “bagging”. Bagging did poorly compared with AdaBoost, but had performance close to AdaBoost if boosting was added.

Post-Fitting is an attempt to reweight the trees when summing tree scores after all the trees are made. Two post-fitting attempts were made. They produced only a very modest (few percent), if any, gain.

For any of these methods, robustness, the resistance to small inaccuracies between data and training events, is important. In MiniBooNE this is being done by generating several dozen Monte Carlo event samples, each with some parameter varied by about one standard deviation. Individual PID variables which are strongly sensitive to variation are eliminated from the boosting variables. This procedure is not yet complete, but the initial results indicate that the boosting output is then quite robust.

In March 2005, a large change in the detector optical model was made requiring retuning of the reconstructions. The networks trained on the old model were tested on the new versions of the same variables. For a fixed background contamination of π^0 events, the fraction of signal kept dropped by 8.3% for boosting and by 21.4% for ANN.

ANN's tend, in practice, to be quite sensitive to a number of parameters. The temperature, hidden layer(s) size, the learning rate, feedback function, $\cdots$, must be chosen. If one multiplies one of the PID variables by two, or interchanges the order of two variables, or puts a variable in twice, the result is likely to change. For more than twenty–thirty PID variables, tuning is quite difficult and improvement in performance problematic.

For boosting many variables ($\approx$100) can be used. There are only a few parameters to optimize. The MiniBooNE experience is that once β, number of leaves, and number of trees are set, they remain about the same for all uses of boosting within the experiment. If a transformation of the PID variables x is made, $y = f(x)$, such that if $x_2 > x_1$, then $y_2 > y_1$, then the results remain identical, as they depend only on ordering. Interchanging variables or putting the same one in twice has no effect on the results.

3 Convergence Speed of Modifications to the Basic Boosting Algorithm

From Table 1, it is seen that none of the tested options for boosting proved superior to AdaBoost or ϵ-Boost for the MiniBooNE experiment. It is still possible to examine modifications to see if the computer time for convergence using the training set can be reduced. Empirically, reducing the correlations between variables has been found to speed convergence for the MiniBooNE experiment. As seen in Table 1, Random Forests with boosting does not do badly and, if optimized further, may become competitive with AdaBoost, while speeding up convergence.

In the method so far described, the score is taken as $+1$ if an event falls on a signal leaf and -1 if the event falls on a background leaf. This means that if

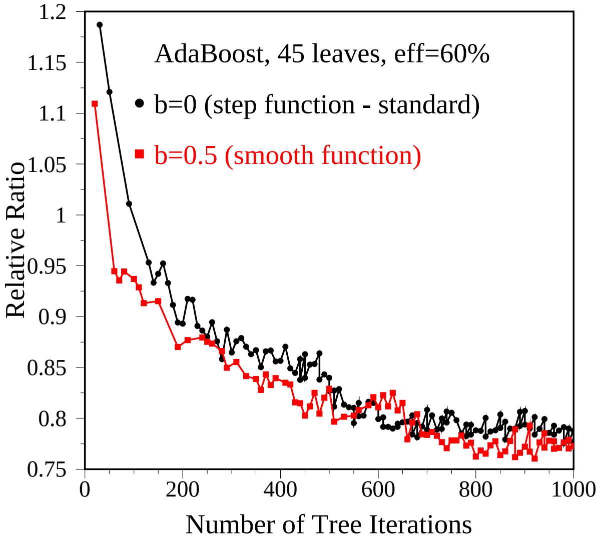


Figure 4. Performance of AdaBoost with $b = 0$ (step function) and $b = 0.5$ (smooth square root function), $\beta = 0.5$, 45 leaves per tree, versus tree iterations.

the event falls on a leaf with purity $P = 0.51$ it gets the same score as if it fell on a leaf with purity 0.91. A "soft scoring" method can be tried using some function of the purity. Empirically it was found that if $d = 2P - 1$, then $T_m = sign(d)|d|^b$, with $b = 0.5$ worked reasonably well. The results are shown in Figure 4. It is seen that the convergence is faster for soft scoring although the end result is about the same as the standard method. From testing a number of samples it appears that, on the average, the final result is about the same for AdaBoost. There is a hint that soft scoring might be slightly better for ϵ-Boost. Since there seems no disadvantage to using soft scoring, it should be considered when one is using boosting in an analysis.

4 Conclusions

Boosting seems very robust. Given enough iterations, AdaBoost or ϵ-Boost reach an optimum level of classification which is not bettered by any variant tried. For the MiniBooNE Monte Carlo samples, boosting was better than ANN's in our tests by factors between 1.2–2. There are ways, such as smooth scoring, to increase the rate of convergence of the algorithm.

Several techniques were tried for reducing the number of variables. Selecting the variables which were most used as splitting variables seemed to work as well as any of the other methods tried.

Downloads in FORTRAN or C++ are available from:

http://www.gallatin.physics.lsa.umich.edu/˜ roe/

References

1. R.E. Schapire "The strength of weak learnability", Machine Learning 5 (2), 197-227 (1990). First suggested the boosting approach for 3 trees taking a majority vote.
2. Y. Freund, "Boosting a weak learning algorithm by majority", Information and Computation 121 (2), 256-285 (1995) Introduced using many trees.
3. Y. Freund and R.E. Schapire, "Experiments with an new boosting algorithm", Machine Learning: Proceedings of the Thirteenth International Conference, Morgan Kauffman, San-Francisco, pp.148-156 (1996). Introduced AdaBoost.
4. J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", Annals of Statistics 28 (2), 337-407 (2000). Showed that AdaBoost could be looked at as successive approximations to a maximum likelihood solution.
5. T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning", Springer (2001). Good reference for decision trees and boosting.
6. B.P. Roe et. al., "Boosted decision trees as an alternative to artificial neural networks for particle identification", NIM A543, pp. 577-584 (2005).
7. Hai-Jun Yang, Byron P. Roe, and Ji Zhu, "Studies of Boosted Decision Trees for MiniBooNE Particle Identification", Physics/0508045, July 2005. Accepted NIM, Sept. 16, 2005.