2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

LogitBoost with Trees Applied to the WCCI 2006 Performance Prediction Challenge Datasets

Roman Werner Lutz

***Abstract*— We apply LogitBoost with a tree-based learner to the five WCCI 2006 performance prediction challenge datasets. The number of iterations and the tree size is estimated by 10-fold cross-validation. We add a simple shrinkage strategy to make the algorithm more stable. The results are very promising since we won the challenge.**

I. Introduction

In recent years a lot of new methods for classification have been proposed and therefore there is a need for a fair comparison. The WCCI 2006 performance prediction challenge (http://www.modelselect.inf.ethz.ch) offers a platform for that. The challenge consists of five binary classification problems. Each problem consists of a training set (input-output-pairs) to fit the model and a test set (inputs only) to predict on new data and evaluate the performance. The datasets are summarized in Table I.

TABLE I
The five challenge datasets

Dataset	Variables/ Features	Training Samples	Test Samples	Proportion of class +1
Ada	48	4562	41471	0.248
Gina	970	3468	31532	0.492
Hiva	1617	4229	38449	0.035
Nova	16969	1929	17537	0.285
Sylva	216	14394	130858	0.062

The performance measure is the balanced error rate (BER) which is the average of the errors on each class on the test set. An additional task on the prediction challenge is to predict the BER (make a BER guess) which will be realized on the test set (generalisation BER). The final test score combines the BER and the guess error.

We decided to use LogitBoost [3] in conjunction with trees. LogitBoost is a "statistical" version of Freund and Schapire's well known and successful AdaBoost [2] because it minimizes the negative binomial log-likelihood instead of the exponential loss. LogitBoost has already been applied successfully to high dimensional microarray data by Dettling and Bühlmann [1].

The most popular choice for the base learner are trees. They can easily model different degrees of interaction (model complexities) and no variable transformations are needed.

Roman Werner Lutz is PhD Candidate of the Seminar for Statistics, ETH Zurich, CH-8092 Zurich, Switzerland (phone: +41 44 632 34 35; fax: +41 44 632 12 28; email: lutz@stat.math.ethz.ch).

II. Methods

The terminology we are going to use is $x_i \in \mathbb{R}^p$ for the input of the i-th observation/sample and $y_i \in \{-1, +1\}$ for the label/output of the i-th observation ($i = 1, \ldots, n$; n is the sample size). We recode the y_i to $y_i^* = (y_i + 1)/2 \in \{0, 1\}$. The LogitBoost algorithm works in the logistic framework. This means we have a predictor function $F : \mathbb{R}^p \to \mathbb{R}$ and conditional probabilities $p(x) = P[Y^* = 1|X = x]$, which are linked by

$$p(x) = \frac{\exp(F(x))}{1 + \exp(F(x))} \quad \text{and} \quad F(x) = \log\left(\frac{p(x)}{1 - p(x)}\right).$$

Since the BER is used as performance measure, the misclassification of a sample belonging to the smaller class (always $+1$) is punished harder than the misclassification of a sample belonging to the bigger class (-1). Therefore, it is a good advice to classify the doubtful observations as $+1$. The best cut off is the proportion of class $+1$ in the data. For Ada for example, the sample i is classified as $+1$, if $p(x_i) > 0.248$.

A. LogitBoost

The LogitBoost algorithm uses Newton steps for fitting a logistic model by maximum binomial likelihood. The algorithm works as follows:

1) Start with $F^{(0)}(x_i) = 0$ and $p(x_i) = \frac{1}{2}$, $i = 1, \ldots, n$.
2) Repeat for $m = 1, \ldots, M$:
   a) Compute the weights and working response
   $$w_i = p(x_i)(1 - p(x_i)),$$
   $$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}.$$
   b) Fit the function $f^{(m)}(x)$, using the tree-based learner, by a weighted least-squares regression of z_i to x_i using weights w_i.
   c) Update $F^{(m)}(x_i) = F^{(m-1)}(x_i) + \nu f^{(m)}(x_i)$ and $p(x_i) = \exp(F^{(m)}(x_i))/(1 + \exp(F^{(m)}(x_i)))$.

An implementation protection is necessary: enforce thresholds on the weights and working responses:

$$w = \max(w, 10^{-15}),$$
$$z = \min(\max(z, -z_{max}), z_{max}).$$

Friedman, Hastie and Tibshirani [3] suggest to use values between 2 and 4 for z_{max}, but our experience is, that this is too small, especially for unbalanced data. So we use $z_{max} = 30$ for Hiva and $z_{max} = 10$ for the other datasets.

0-7803-9490-9/06/$20.00/©2006 IEEE

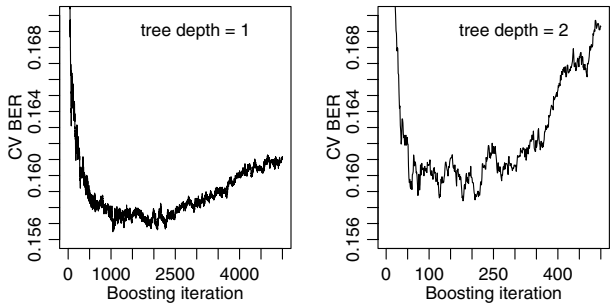

Fig. 1. CV BER as a function of the boosting iteration for Ada for $\nu = 0.3$ with trees of depth one (left) and depth two (right). Both curves show the usual behavior of first going down and then going up again because of overfitting. Trees of depth one are the better choice because we reach a smaller CV BER.

The number of iterations M is estimated by 10-fold cross-validation (CV). This means that the data is split into ten parts. One part is set aside and the algorithm is run for a large number of iterations on the other nine parts. At each iteration, we predict on the excluded part and calculate the BER. The procedure is repeated for each part and the ten resulting BER curves (as functions of the boosting iteration) are averaged to give the CV BER curve (see Figure 1). The stopping iteration M is the iteration which minimizes the CV BER. Actually, we did the cross-validation three times with different folds and averaged the results.

The learner (fitting method) in step 2b) is a regression tree of prefixed depth. The depth controls the degree of interaction of the fitted model. At each iteration the tree is grown without pruning to the prefixed depth and each split is allowed (even when only one observation remains in a node). We run the whole boosting algorithm five times. In each run we use another tree depth: in the first run, the learner is always a tree of depth one (stump); in the second run, the learner is always a tree of depth two (tree with four terminal nodes); and so on until depth five. The tree depth which gives the best CV BER curve is finally chosen. When different tree depths lead to approximately to the same CV BER, we take the smaller one to keep the model simple. Most of the time a tree depth of one or two is enough (see also Figure 1).

The ν in step 2c) is the so-called shrinkage factor. The natural value is one, but smaller values are often a better choice. This makes the algorithm slower, since more iterations are needed, but more stable, since the steps taken are smaller. Additionally, this often leads to some improvement of predictive power and only rarely to a deterioration. We suggest to start with $\nu = 1$ and look at the CV BER curve as a function of the boosting iteration (see Figure 2). If the curve is very rough around its minimum, or if the minimum occurs already after a few iterations, we reduce ν by a factor of approximately three and rerun the algorithm until the CV BER curve is smooth enough. We don't have a strict mathematical criterion for stopping and we choose ν by visual inspection. Most of the time a value of 0.3 or 0.1 is reasonable.

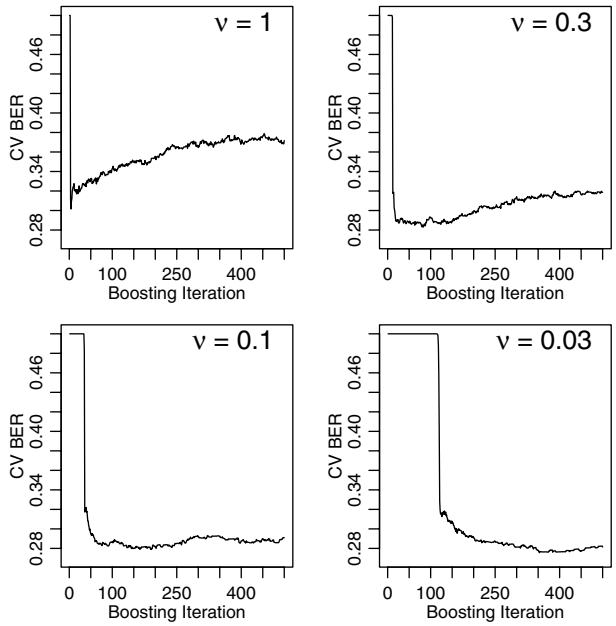

Fig. 2. CV BER as a function of the boosting iteration for Hiva with trees of depth two for $\nu = 1, 0.3, 0.1, 0.03$. For $\nu = 1$ the minimum occurs already after four iterations and the CV BER curve goes up again quickly. Smaller ν lead not only to smoother curves but also to smaller minima.

Our BER guess for the test set (generalisation BER) is the CV BER at the stopping iteration. On the one hand, this is a little bit too pessimistic, since the estimation of generalisation BER by CV is biased upward because only nine tenth of the data is used to fit the model. On the other hand, it is slightly too optimistic, since the stopping iteration is explicitly chosen to minimize the CV BER on the given training data. This effect however is most of the time small, since the CV BER curve is usually smooth (ensured by a small ν). Our hope is that the two effects cancel each other out.

B. Special Treatment of the Nova Dataset

Because the Nova dataset has a large number of variables and is sparse (each variable is binary and most of the values are zero), some preprocessing seems to be adequate. We drop the variables with only one or two entries equal to one (training and test set) and calculate the principal components with the remaining 16881 variables (with centered and scaled variables). It is important to use also the observations from the test set for the calculation. We take the first 400 principal components for LogitBoost.

C. Variable Pre-selection

To reduce the high dimensional datasets one could do a variable pre-selection with simple univariate tests. It is not clear whether this improves or worsens the performance of LogitBoost. At least it reduces the computation time a lot. We try both and apply LogitBoost with and without variable pre-selection. For the pre-selection we use the Wilcoxon test for continuous variables (the two groups are given by the

labels y) and the Fisher exact test for the binary variables (with the 2×2 contingency table constructed with the x-variable and the labels y). Each variable with a p-value above 0.1 is dropped. This threshold is chosen ad hoc and reduces the number of variables (principal components for Nova) for Ada from 48 to 38, for Gina from 970 to 482, for Hiva from 1617 to 686, for Nova from 400 to 237 and for Sylva from 216 to 81.

III. Results

A. LogitBoost without and with Variable Pre-selection

Table II contains the results without and Table III with variable pre-selection. The outcomes are very similar. In the challenge, these two submissions ranked fourth and second. For Ada and Sylva, a main effects model performed best. For Hiva and Nova, first order interactions were needed and for Gina, quite complicated trees of depth five were fitted. This is no surprise since Gina was originally a ten class classification problem. The two classes for the challenge were constructed by combining five original classes at a time. For Ada, Gina and Sylva, a shrinkage factor ν of 0.3 seems to be small enough, while for the very unbalanced Hiva dataset a smaller value is needed.

Our BER guess method worked quite satisfactorily. On Ada, Hiva and Sylva we were too optimistic, while on Gina and Nova we were too pessimistic.

TABLE II

Results without variable pre-selection. CV BER is the cross-validated BER on the training set at the stopping iteration and also our BER guess.

Dataset	Tree depth	ν	No. of iterations	CV BER = BER guess	BER on test set
Ada	1	0.3	1043	0.1565	0.1712
Gina	5	0.3	741	0.0415	0.0385
Hiva	2	0.03	353	0.2756	0.2888
Nova	2	0.1	294	0.0506	0.0491
Sylva	1	0.3	273	0.0058	0.0064
Average				0.1060	0.1108

TABLE III

Results with variable pre-selection. CV BER is the cross-validated BER on the training set at the stopping iteration and also our BER guess.

Dataset	Tree depth	ν	No. of iterations	CV BER = BER guess	BER on test set
Ada	1	0.3	979	0.1550	0.1708
Gina	5	0.3	1308	0.0388	0.0357
Hiva	2	0.03	249	0.2795	0.2946
Nova	2	0.1	365	0.0503	0.0469
Sylva	1	0.3	229	0.0062	0.0066
Average				0.1060	0.1109

B. A Mix: Predicted Probabilities Averaged

We made a third submission for which we averaged the predicted probabilities of LogitBoost with and without variable pre-selection (with acronym "LB tree mix"). Our BER guess for each data set is the minimum of the two individual BER guesses. The results are contained in Table IV. For Ada and Nova, the BER on the test set of the mix was better than the BER of the two individual methods. For the other data sets, the BER of the mix was always nearer to the BER of the better individual method. All in all, this gave a small improvement. In the challenge, this submission was fifth.

TABLE IV

Results for the mix (averaged probabilities of LogitBoost with and without variable pre-selection).

Dataset	BER guess	BER on test set	Guess error	Test score	Rank
Ada	0.1550	0.1705	0.0155	0.1859	7
Gina	0.0388	0.0361	0.0027	0.0386	5
Hiva	0.2756	0.2904	0.0148	0.3035	9
Nova	0.0503	0.0467	0.0036	0.0498	5
Sylva	0.0058	0.0064	0.0006	0.0070	15
Average	0.1051	0.1100	0.0074	0.1170	8.2

C. Intercept Adaptation

For our forth and challenge winning submission we made a last small adjustment by adapting the intercept on the logit scale (the acronym "LB tree mix cut adapted" is kind of misleading). We added the same constant to all $F(x_i)$ of the mixed submission so that the average of the resulting probabilities is exactly the proportion of class $+1$ in the data. This is a kind of bias correction and improved the BER of four data sets and had no effect for Gina. The results are given in Table V. For the BER guess we took the same values as for the mixed submission (except for Hiva where we hoped to be better and reduced the BER guess from 0.2756 to 0.27).

TABLE V

Results with "intercept adaptation".

Dataset	BER guess	BER on test set	Guess error	Test score	Rank
Ada	0.1550	0.1696	0.0146	0.1843	3
Gina	0.0388	0.0361	0.0027	0.0386	5
Hiva	0.2700	0.2871	0.0171	0.3029	8
Nova	0.0503	0.0458	0.0045	0.0499	8
Sylva	0.0058	0.0063	0.0005	0.0067	7
Average	0.1040	0.1090	0.0079	0.1165	6.2

IV. Conclusions

We have shown by winning the challenge that LogitBoost with trees can perform very well on high-dimensional classification problems.

There are three tuning parameters, which have to be chosen: the shrinkage factor ν, the tree depth and the number of iterations M.

Our choice of the shrinkage factor ν is somewhat arbitrary. We chose it by visual inspection of the CV BER curve. A simple alternative would be to take always $\nu = 0.1$ as a good default. Smaller values are seldom needed and $\nu = 1$ should not be used. With a small ν we can ensure that enough

iterations are performed and that all the relevant variables have been chosen during the iteration. The selection of the number of iterations M then becomes easier, too.

The tree depth and the number of iterations M can simply be chosen by CV.

ACKNOWLEDGMENT

The author would like to thank Prof. Isabelle Guyon for organizing the challenge and Prof. Peter Bühlmann for helpful discussion and comments and for the opportunity to participate in the challenge. I also thank Corinne Dahinden, Nicolai Meinshausen, Lukas Meier and Markus Kalisch for the good time we had working on the challenge and for stimulating in-house competition. A special thank goes to our compute server deb2 with 16 GByte RAM. Without her, the computation of the principal components for Nova would not have been possible.

REFERENCES

[1] M. Dettling and P. Bühlmann, "Boosting for tumor classification with gene expression data," *Bioinformatics*, vol. 19, pp. 1061–1069, 2003.

[2] Y. Freund and R. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *J. Comput. System Sciences*, vol. 55, pp. 119–139, 1997.

[3] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Ann. Stat.*, vol. 28, pp. 337–374, 2000.