Fast Committee Machines for Regression and Classification

Harris Drucker

Monmouth University
West Long Branch, NJ 07764
drucker@monmouth.edu

Abstract

In many data mining applications we are given a set of training examples and asked to construct a regression machine or a classifier that has low prediction error or low error rate on new examples, respectively. An important issue is speed especially when there are large amounts of data. We show how both classification and prediction error can be reduced by using boosting techniques to implement committee machines. In our implementation of committees using either classification trees or regression trees, we show how we can trade off speed against either error rate or prediction error.

Introduction

Boosting techniques [Drucker (1993, 1994, 1996a), Freund and Schapire (1996a,b), Schapire (1990)] allows one to obtain smaller prediction errors (in regression) and lower error rates (in classification) using multiple predictors. The ensemble of classifiers is often called a committee machine. As shown in Figure 1, the same set of input features (independent variables) is applied to what are termed the "weak learners" and each learner puts forth an hypothesis h_i with a confidence inversely related to β_i. For classification, h_i is either 0 or 1 (for the two class case) or the predicted value of the dependent variable (in the case of regression). In a serial machine, the outputs of the weak learners are output sequentially so as we move down the chain of learning machines, the execution speed decreases but generally the prediction error or classification error also decreases. Therefore, by choosing to use only i ($i \leq T$) of the T learners, we can choose the appropriate combination of execution speed and either classification accuracy or prediction accuracy. If parallel implementation is feasible (either each learner or blocks of N learners), then speed is governed by the slowest weak learner or the slowest block of N learners. Several studies of boosting in classification [Breiman (1996b), Freund and Schapire (1996a)] have shown the superiority of boosting over another committee machine approach termed bagging [Breiman (1994,1996a)] but this is the first experimental study where we also investigate combining regressors using boosting techniques. In boosting each machine is trained on different subsets of the training set. The first machine is trained on examples picked with replacement from the original training set. We then pass *all* the training patterns through this first machine and note which ones are in error (in classification). For regression machines, those patterns whose predicted values differ most from their observed values are defined to be "most" in error. For those patterns in error, the sampling probabilities are adjusted so that difficult examples are more likely to be picked as members of the training set for the second machine. Thus, different machines are better in different parts of the observation space. Regressors are combined using the weighted median while classification machines are combined using a weighted sum of the hypotheses. Details of these weighting scheme are discussed later.

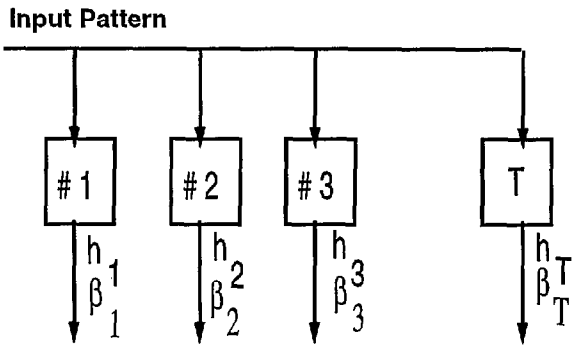

FIGURE 1. BOOSTING ENSEMBLE

For classification, the parameter of interest is the error rate, while for regression, the parameters of interest are the modeling error (ME) and the prediction error (PE). We discuss regression first: Suppose we are given a set of *observations*, $(y_i, \boldsymbol{x}_i)$ $i=1,...,N_1$ where N_1 is the number of training set observations, and $\boldsymbol{x}$ is an M-variate vector forming the features. The probability density function of $(y,\boldsymbol{x})$ is fixed but unknown. Based on these observations, we form a predictor $y^{(p)}(\boldsymbol{x})$. We define a sample modeling error (ME) and prediction

error (PE):

$$PE = \frac{1}{N_2}\sum_{i=1}^{N_2}[y_i - y_i^{(p)}(\boldsymbol{x}_i)]^2$$

$$ME = \frac{1}{N_2}\sum_{i=1}^{N_2}[y_i^{(t)} - y_i^{(p)}(\boldsymbol{x}_i)]^2$$

where $y_i^{(p)}(\boldsymbol{x}_i)$ is the prediction for the i'th test example, y_i is the i'th observation, and $y_i^{(t)}$ is the "truth". The parameters of $y^{(p)}(\boldsymbol{x})$ are obtained from the N_1 training set observations but the y_i and $\boldsymbol{x}_i$ in the above summations are obtained from a set of N_2 observations (the test set) never seen before. If the noise is additive, then $y_i = y_i^{(t)} + n_i$ where n_i is the i'th sample of the noise. Furthermore, if E[n] = 0 and $E[n_i n_j] = \delta_{ij}\sigma^2$, then we may take the expectation with respect to $(y, \boldsymbol{X})$ and obtain (Breiman and Spector, 1992):

$$E[PE] = \sigma^2 + E[ME]$$

This shows us that even if we know the model exactly, there is a minimum prediction error due to the noise. Our problem will be that the modeling error is also nonzero because we have to determine the model in the presence of noise. Since we don't know the probability distributions, we approximate the expectation of the ME and PE using the sample ME (if the truth is known) and sample PE and then average over multiple experiments. Similarly, for the classification case, we have a training error for those examples we train on and a test error for the test examples tested on a machine constructed using the training examples. Our implementation of each weak learner is a tree, either a regression tree or a classification tree because of their ease of implementation, small training time, and fast execution speed.

Boosting Algorithm

The theory behind the boosting algorithms are presented elsewhere(Freund and Schapire, 1996b). Here we present the boosting algorithm for regression:

Repeat the following while the average loss $\bar{L}$ defined below is less than .5 .

1. The probability that training sample i is in the training set is $p_i = w_i / \sum w_i$ where the summation is over all members of the training set. Initially, the w_i are set equal to unity. Pick N_1 samples (with replacement) to form the training set. This may be implemented by marking a line of length $\sum w_i$ and subsections of length w_i . A uniform number picked from the range $[0, \sum w_i]$ and landing in section i corresponds to picking pattern i.

2. Construct a regression machine t from that training set. Each machine makes a hypothesis: $h_t: x \rightarrow y$

3. Pass *every* member of the training set through this machine to obtain a prediction $y_i^{(p)}(\boldsymbol{x}_i)$ $i = 1, \ldots N_1$.

4. Calculate a loss for each training sample $L_i = L\left[\,|\, y_i^{(p)}(\boldsymbol{x}_i) - y_i \,|\,\right]$. The loss L may be of any functional form as long as $L \in [0,1]$. If we let

$$D = sup\,|\, y_i^{(p)}(\boldsymbol{x}_i) - y_i \,|\quad i = 1, \ldots, N_1$$

then we have three candidate loss functions we examined:

$$L_i = \frac{|\, y_i^{(p)}(\boldsymbol{x}_i) - y_i \,|}{D} \qquad (linear)$$

$$L_i = \frac{|\, y_i^{(p)}(\boldsymbol{x}_i) - y_i \,|^2}{D^2} \qquad (square\ law)$$

$$L_i = 1 - exp\left[\frac{-|\, y_i^{(p)}(\boldsymbol{x}_i) - y_i \,|}{D}\right] \qquad (exponential)$$

5. Calculate an average loss: $\bar{L} = \sum_{i=1}^{N_1} L_i p_i$

6. Form $\beta = \frac{\bar{L}}{1-\bar{L}}$. β is a measure of confidence in the predictor. Low β means high confidence in the prediction.

7. Update the weights: $w_i \rightarrow w_i \beta^{**}[1 - L_i]$, where ** indicates exponentiation. The smaller the loss, the more the weight is reduced making the probability smaller that this pattern will be picked as a member of the training set for the next machine in the ensemble.

8. For a particular input $\boldsymbol{x}_i$, each of the T machines makes a prediction h_t, $t = 1, \ldots, T$. Obtain the cumulative prediction h_f using the T predictors:

$$h_f = inf\left\{ y \in Y: \sum_{t:h_t \le y} \log(1/\beta_t) \ge \frac{1}{2}\sum_t \log(1/\beta_t) \right\}$$

This is the weighted median. Intuitively, the effect of varying the weight w_i to give more emphasis to "difficult" examples means that each subsequent machine has a disproportionately harder set of examples. Thus, the average loss tends to increase as we iterate through the algorithm and ultimately the bound on $\bar{L}$ is not satisfied and the algorithm terminates.

For classification, we replace steps 4 and 8 above. In step 4, the loss for pattern i becomes $L_i = |\, h_t(i) - c(i) \,|$ where $h_t(i)$ is the hypothesis of machine t (0 or 1) and $c(i)$ is the correct classification (either 0 or 1). The issue of multiclass classification will be discussed in the conclusion. In step 8, the final hypothesis is:

$$h_f(i) = \begin{cases} 1, & \sum_{t=1}^{T}(log\frac{1}{\beta_t})\, h_t(i) \ge \frac{1}{2}\sum_{t=1}^{T}\log\frac{1}{\beta_t} \\ 0, & otherwise \end{cases}$$

Note that the examples referred to in the algorithm above only refers to training examples and we usually observe convergence to zero classification error or zero prediction error on these training examples. However, we are interested in the performance on a test set. Therefore, at each stage of boosting, we pass the test set through the ensemble and use step 8 to obtain the test performance. How well we generalize to the test set will depend on our choice of weak learner, e.g., neural net, tree, nearest neighbor classifier.

Trees

Regression trees are based on Breiman's CART [Breiman, 1994]. In this implementation of regression trees, one starts (at the top) with a main node that examines a specified feature of the input pattern and compares its value to a "separator" value assigned to that node. If the value of the feature is less than or equal to the separator, we continue down the left side of the tree, else the right side. Each of the children of the main node has a specified feature and separator value for that feature. Each parent node has two children nodes and those children nodes become parents of their children. Depending on the values of the features for the input pattern, we continue down the tree, at each node going left or right until we reach a terminal leaf. At the terminal leaves, we assign a hypothesis that is the predicted value of the regression. For classification trees, the building criterion is based on an information theoretic basis found in C4.5 [Quinlan, 1993]. To generalize well to as an yet unseen test set, we then prune back the trees using a separate pruning set. Pruning [Mingers, 1989] performs two essential tasks (a) it increases the execution speed because the tree is smaller and (b) the generalization is better.

Experiments

We tried several cases of classification and regression using the techniques described above. The first regression problem was based on that of Friedman (1991), which is a nonlinear prediction problem which has 10 independent variables that are uniform in [0,1] Here we can determine both the modeling and prediction error.

$$y=10\sin(\pi x_1 x_2)+20(x_3-.5)^2+10x_4+5x_5+n$$

where n is normal (0,1). Therefore, only five predictor variables are really needed, but the predictor is faced with the problem of trying to distinguish the variables that have no prediction ability (x_6 to x_{10}) from those that have predictive ability (x_1 to x_5). For different training set sizes (pruning size is 20% of the training set size) and a test set of of size 10,000 we plot (Figure 2) execution speed (on a SparcStation 20) versus the prediction error on the test set. Each data point is the average of ten samples. Generally as we increase the number of trees in the ensemble (thereby decreasing the execution speed), the prediction error decreases. Because of the inherent noise floor in this prediction problem, the PE tends to asymptote with increasing number of trees and large enough number of examples. There is a continuum of prediction error versus speed trade-offs. We can show similar behavior on other regression and classification problems.

Discussion and Conclusions

Discussion and Conclusions" We have shown how to use boosting to reduce error rates in classification and prediction error in regression using classification trees and regression trees, respectively. Of critical importance is the pruning of the trees by using a separate set of pruning data after the tree has been initially grown using the training set. Pruning both increases the speed and improves the generalization to samples as yet unseen. By picking the appropriate number of trees in the ensemble, one can trade off speed versus performance. If the speed performance is not acceptable (but the error rate or prediction error is), then one choice is a parallel architecture. Generally, there are between forty and seventy trees in an ensemble and therefore the speed would increase by those factors (but only approximately since not all trees are the same size). If error rate or prediction error is the issue, then another possibility is to build a single neural network. Generally a single neural network is slower than these ensembles because trees just implement simple IF statements, while neural networks need to implement both multiplication and some sigmoid function as the transfer functions. However, these issues must be decided on a case-by-case basis.

Acknowledgements

Part of this work was performed under ARPA contract N00014-94-C-0186 while at Lucent Technologies.

References

Breiman, L. (1996),"Bias, Variance, and Arcing Classifiers Technical Report 460, Department of Statistics, University of California, Berkeley, CA. *Annals of Statistics* (to be published)

Breiman, L. and Spector, P., (1992). "Submodel Selection and Evaluation in Regression. The X-Random Case", *International Statistical Review*, **60**, 3, pp.291-319.

Breiman, L., Friedman, H.H., Olshen, R.A. and Stone, C.J. (1984), *Classification and Regression Trees*, Wadsworth International Group.

Breiman, L. (1996) "Bagging Predictors" *Machine Learning*, vol. 26, No. 2, pp. 123-140.

Breiman, L. (1996),"Bias, Variance, and Arcing Classifiers Technical Report 460, Department of Statistics, University of California, Berkeley, CA. *Annals of Statistics* (to be published)

Drucker, H., (1997), "Improving Regressors using Boosting Techniques", 1997 International Conference on Machine Learning.

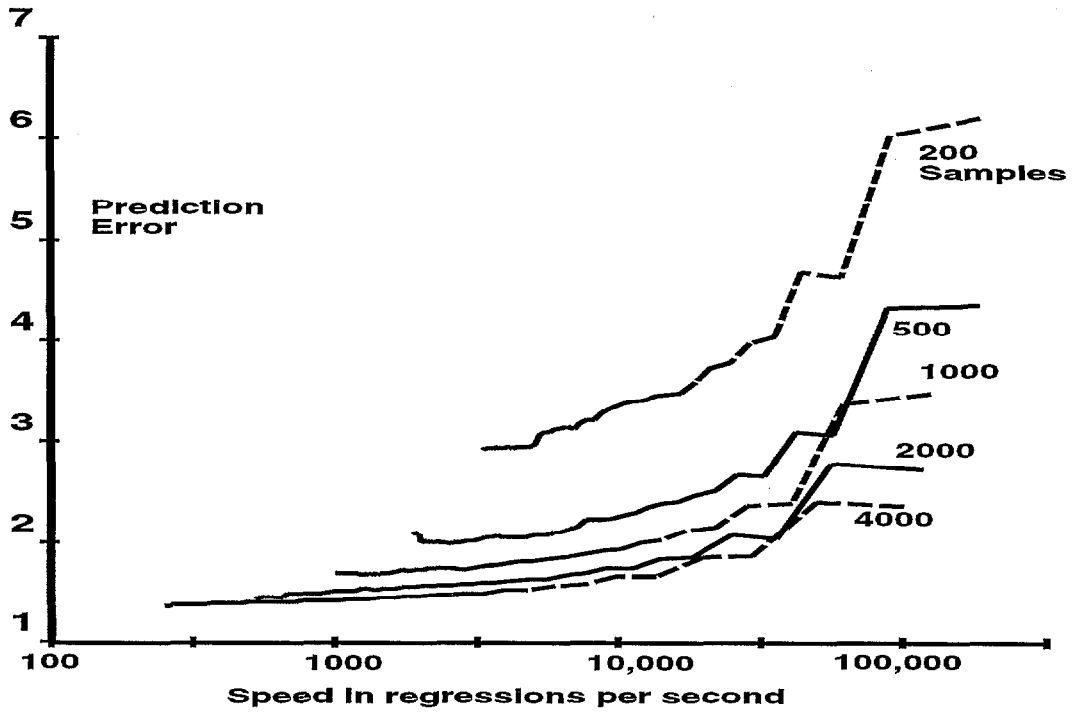

Figure 2. Prediction error versus speed for a regression problem

Drucker, H., Schapire, R.E., Simard, P., (1993), "Boosting Performance in Neural Networks", *International Journal of Pattern Recognition and Artificial Intelligence*, 7, 4, pp. 705-719.

Drucker, H., Cortes, C., Jackel, LD., LeCun, Y., Vapnik V., (1994). "Boosting and Other Ensemble Methods", *Neural Computation* , **6**, 6, pp. 1287-1299. pp. 1287-1299.

Drucker, H., (1996), "Fast Decision Tree Ensembles for Optical Character Recognition", *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval*

Drucker, H., (1996) and Cortes, C., "Boosting Decision Trees", *Neural Information Processing 8*, ed: D.S. Touretzky, M,C. Mozer and M.E. Hasselmo. Morgan Kaufmann, pp.479-485.

Drucker, H., (1997), "Improving Regressors using Boosting Techniques", submitted to the 1997 conference on Machine Learning.

Freund, Y., (1990), "Boosting a Weak Learning Algorithm by Majority", *Proceedings of the Third Workshop on Computational Learning Theory*, Morgan-Kaufmann, 202-216

Freund, Y. and Schapire, R.E. (1996a) "Experiments with a New Boosting Algorithm", *Machine Learning: Proceedings of the Thirteenth Conference*, ed: L. Saitta, Morgan Kaufmann, pp. 148-156.

Freund, Y. and Schapire, R.E. (1996b) "A decision-theoretic generalization of on-line learning and an application to boosting", at web site: http://www.research.att.com/~yoav. or http://www.research.att.com/orgs/ssr/people/~yoav. An extended abstract appears in the *Proceedings of the Second European Conference on Computational Learning Theory*, Barcelona, Springer-Verlag, March, 1995, pp. 23-37.

Friedman J., (1991), "Multivariate Adaptive Regression Splines", *Annal of Statistics*, vol 19, No. 1, pp. 1-141

J. Mingers (1989a), "An Empirical Comparison of Pruning Methods for Decision Tree Induction", *Machine Learning*, 4:227-243.

J.R.Quinlan (1993), *C4.5: Programs For Machine Learning*, Morgan Kauffman.

Schapire, R.E., (1990), "The Strength of Weak Learnability", *Machine Learning*, **5**, 2, pp. 197-227.