PERT – Perfect Random Tree Ensembles

Adele Cutler

Mathematics and Statistics
Utah State University
Logan, UT 84322

Guohua Zhao

Center for Basic Research
in the Social Sciences
Harvard University
Cambridge, MA 02138

Abstract

Ensemble classifiers originated in the machine learning community. They work by fitting many individual classifiers and combining them by weighted or unweighted voting. The ensemble classifier is often much more accurate than the individual classifiers from which it is built. In fact, ensemble classifiers are among the most accurate general-purpose classifiers available.

We introduce a new ensemble method, PERT, in which each individual classifier is a perfectly-fit classification tree with random selection of splits. Compared to other ensemble methods, PERT is very fast to fit. Given the randomness of the split selection, PERT is surprisingly accurate. Calculations suggest that one reason why PERT works so well is that although the individual tree classifiers are extremely weak, they are almost uncorrelated.

The simple probabilistic nature of the classifier lends itself to theoretical analysis. We show that PERT is fitting a continuous posterior probability surface for each class. As such, it can be viewed as a classification-via-regression procedure that fits a continuous interpolating surface. In theory, this surface could be found using a one-shot procedure.

Keywords

Boosting; Classification; Ensemble classifier; Tree classifier.

1 Introduction

Ensemble classifiers are built by fitting many individual classifiers (called "base learners") and combining them by weighted or unweighted voting.

It is now well known (for example, Breiman 1998) that ensemble classifiers often give more accurate predictions than the base learners from which they are built. Indeed, for many benchmark datasets, ensemble classifiers are among the most accurate classifiers available, and substantially more accurate than traditional statistical methods. The success of ensemble methods has provoked considerable interest in understanding why such methods succeed and identifying circumstances in which they can be expected to produce good results.

Ensemble methods differ in the way the base learners are fit and combined.

In bagging (Breiman, 1996), an ensemble classifier is constructed by generating bootstrap samples from the original data set, constructing a classifier from each bootstrap sample, and voting to combine. In Adaboost (Freund and Shapire 1996) and other arcing algorithms (Breiman 1998) the successive classifiers are constructed by giving increased weight to those points that have been frequently misclassified up to this time, and the classifiers are combined using weighted voting.

Both bagging and arcing perturb or reweight the training data to generate the individual classifiers for the ensemble. Another approach is to fix the training data and allow the base learner itself to have a random element. One such method is the randomized C4.5 tree (Dietterich 1998), which extends the popular C4.5 base learner (Quinlan 1993). The randomized C4.5 tree splits a node by finding the 20 best candidate splits and choosing one at random. In situations with little or no classification noise the randomized C4.5 tree was shown to be competitive with (and perhaps slightly superior to) bagging C4.5.

Breiman (1999) gives a general framework for tree ensembles in which each tree is determined by a random vector which is independently sampled from the same distribution ("random forests"). He shows that under reasonable conditions, the generalization error for a random forest converges a.s. as the number of trees increases. The accuracy depends on the strength of the individual classifiers and the dependence between them. Accordingly, Breiman (1999) studies ensembles of trees that are constructed by selecting the best split for a randomly chosen feature or linear combination of features. Random selection of features helps to reduce correlation, and optimizing over the split selection maintains the strength of the individual classifiers.

We take the randomization one step further, introducing a new technique (PERT) in which the base learner randomly chooses both the feature on which to split and the split itself. Because PERT does not optimize over either the feature or the location of the split, it is very easy to code and very fast to fit.

Earlier work (Mingers 1989, Buntine and Niblett, 1992) showed that random split methods similar to the PERT base learner did not work well, but these studies were based on using individual classifiers, not ensembles. We show that the PERT ensemble is competitive with Adaboost and random forests. Moreover, PERT lends itself to theoretical analysis. We show that PERT is actually fitting a continuous posterior probability surface for each class.

The PERT method is described in detail in Section 2. Timing comparisons with CART are presented in Section 3. While the timing results are attractive, our focus is not so much on computational efficiency but on understanding why a tree-based classifier that makes no attempt at finding optimal splits can produce competitive results. In Section 4, we show that PERT does indeed produce results that are competitive with bagging and arcing CART and random forests. Section 5 gives an explanation of PERT's performance in terms of strength and correlation. Section 6 contains our theoretical analysis, and Section 7 gives some concluding remarks.

2 Perfect Random Trees

The perfect random tree ensemble, PERT, is so named because the PERT base learner is a random tree classifier that fits the training data perfectly. The PERT base learner is constructed by first placing all data points in the root node. At each step of the tree construction, each nonterminal node is split by randomly choosing two data points from the node until these two belong to different classes. Let

Table 1: Number of Terminal Nodes and CPU Time.

		Number of Nodes		CPU Time (secs/100)	
Data Set	Sample Size	CART	PERT	CART	PERT
waveform	100	29.3	122.0	1.51	.03
	400	102.5	461.2	11.04	.13
	800	193.9	891.1	28.92	.27
twonorm	100	20.2	91.0	1.26	.03
	400	68.3	322.0	9.64	.11
	800	127.5	604.8	25.77	.23
threenorm	100	29.4	121.1	1.51	.03
	400	107.6	462.8	10.73	.14
	800	207.2	905.9	26.75	.28
ringnorm	100	23.5	115.0	1.63	.03
	400	74.4	427.8	13.78	.13
	800	133.6	830.2	40.04	.27

$\mathbf{x} = (x_1, \ldots, x_p)$ and $\mathbf{z} = (z_1, \ldots, z_p)$. If this is not possible, all the data points in the node must be from the same class and the node is terminal. Now, randomly choose a feature on which to split, say feature j, and split at $\alpha x_j + (1-\alpha) z_j$, where α is generated from a uniform(0,1) distribution.

Ties are taken care of by repeating the procedure until a definitive split is obtained. If no such split is found after 10 tries, the node is declared to be terminal (so in this case, the tree would not perfectly fit the data).

To form an ensemble, PERT base learners can be combined by simply fitting many PERT base learners to the entire training set and voting these classifiers. Alternatively, PERT base learners can be combined by bagging (Breiman 1996). In this case, the PERT base learner is fit to many bootstrap samples from the training data and these classifiers are combined by voting.

3 Timing

PERT is trivial to code and PERT ensembles are extremely fast to construct because there is no optimality criterion to compute. CART, for example, searches over all possible features and all possible splits to find the best way to split each node. Even random feature selection (Breiman, 1999) selects an optimal split for the given choice of feature(s). However, the speed of fitting PERT is somewhat offset by the fact that it produces very large trees (because it fits the training data perfectly), so it tends to be slow to evaluate.

Table **??** gives the average number of terminal nodes and CPU times for PERT and CART (using the Gini criterion) for four synthetic data sets. The CART tree was coded using fast update methods. Both trees were grown until they perfectly

Table 2: Test Set Errors (%).

Data Set	Adaboost CART	Forest-RI Multiple	Forest-RI Single	PERT
glass	22.0	20.6	21.2	21.4
breast c.	3.2	2.9	2.7	2.8
diabetes	26.6	24.2	24.3	24.5
sonar	15.6	15.9	18.0	16.2
vowel	4.1	3.4	3.3	2.3
ionosphere	6.4	7.1	7.5	6.8
vehicle	23.2	25.8	26.4	27.4
soybean	5.8	6.0	6.5	6.5
Ger. credit	23.5	24.4	26.2	24.9
image	1.6	2.1	2.7	2.6
ecoli	14.8	12.8	13.0	14.5
votes	4.8	4.1	4.6	4.4
liver	30.7	25.1	24.7	27.9
letters	3.4	3.5	4.7	4.4
sat-images	8.8	8.6	10.5	10.2
zipcode	6.2	6.3	7.8	7.6
waveform	17.8	17.2	17.3	17.8
twonorm	4.9	3.9	3.9	3.6
threenorm	18.8	17.5	17.5	17.6
ringnorm	6.9	4.9	4.9	12.8

classified the training data and results were averaged over 500 samples. Average CPU times are given (in hundredths of seconds) on a 3.36 MH Ultra Spark II processor.

The PERT trees are between 4 and 5 times as large as the CART trees for these problems. CART is generally around 2 orders of magnitude slower than PERT.

4 Accuracy

We compared the accuracy of PERT to published results from Breiman (1999) for Adaboost with CART as the base learner and random feature selection (Forest-RI).

For comparability, we followed the same procedure as Breiman (1999) to obtain the results for PERT. For the real datasets, 10% of the data were randomly chosen and set aside as a test set. The ensemble method was fit to the remaining data and then used to classify the test set. The entire procedure was repeated 500 times and the test set errors were averaged. For the synthetic data, a new training set of size 300 and test set of size 3000 were generated for each of the 500 repetitions. Most of the datasets are available from the UCI repository (Murphy and Aha, 1994). The synthetic datasets are described in Breiman (1996).

For the Adaboost results (from Breiman, 1999), 50 trees were used, and 100 trees

Table 3: Test Set Errors (%) for Larger PERT Ensembles.

Data Set	100	200	400	1600
waveform	17.8	17.3	17.1	16.8
twonorm	3.6	3.2	3.1	3.0
threenorm	17.6	16.4	15.8	15.3
ringnorm	12.8	11.8	11.1	10.7

Table 4: Strength and Correlation for bagged CART and PERT.

	Strength, s		Correlation, $\bar{\rho}$	
Data Set	CART	PERT	CART	PERT
waveform	.37	.20	.22	.06
twonorm	.54	.50	.14	.07
threenorm	.31	.20	.13	.04
ringnorm	.48	.27	.14	.05

were used for all the other ensembles, with the exception of the zipcode data, for which these numbers were doubled.

The results (Table **??**) are surprisingly good for a method as simple and fast as PERT. For most of the examples, PERT falls within the range of the other three methods. The PERT results for vowel and twonorm are somewhat better than those for the other three methods, while PERT's results for vehicle are somewhat worse. The only striking difference is for ringnorm, where PERT does not do well. It turns out that we can get slightly better results if we combine more classifiers (Table **??**), but the results for ringnorm are still not good.

We believe that PERT is suffering from the curse of dimensionality. Ringnorm is a mixture of two multivariate normal distributions. The covariance matrix for class 1 is four times the identity, while that for class 2 is the identity. The Bayes rule classifies a region around the class 2 mean as class 2, with the entire surrounding region as class 1. A closer inspection of the results shows that PERT tends to make mistakes in this tail region.

5 Strength and Correlation

For obvious reasons, a single PERT tree is not a good classifier. Compared to, say, CART, it works very poorly. However, in ensembles it appears to be competitive with some of the best methods available. In order to help understand why, consider Breiman's bound on the generalization error:

$$PE^{\star} \leq \bar{\rho}(1 - s^2)/s^2,$$

where s is a measure of strength of the individual classifiers in the forest and $\bar{\rho}$ is a measure of the correlation between them in terms of the raw margin (for details, see Breiman 1999). While the bound is loose, it suggests that a reasonable strategy for random forests is to find trees that give the correct class on average (high s) but make mistakes in different places (low $\bar{\rho}$).

We calculated s and $\bar{\rho}$ for PERT and bagged CART for the four synthetic datasets and obtained the results given in Table **??**. These results suggest that part of the reason why PERT performs well is that although the individual PERT classifiers are relatively weak (low values of s), they have very low correlation (low values of $\bar{\rho}$). Presumably, if we could strengthen the PERT classifier without substantially increasing the correlation, an even better ensemble classifier could be developed.

6 Posterior Probabilities

Following Breiman (1999), consider an ensemble of random tree classifiers (a "random forest"), $h(X, \theta_1), \ldots, h(X, \theta_N)$, where $\theta_1, \ldots, \theta_N$ are independent and identically distributed. For PERT, θ denotes the vector of random numbers used to select random features and random splits.

The posterior probability that a random tree predicts class j at X, given the training data $(x_i, y_i), i = 1, \ldots, n$, is

$$Q_j(X) = P_\theta \left(h(X, \theta) = j \right).$$

Note that h depends on the training data. In practice, we estimate Q_j using

$$\hat{Q}_j(X) = \frac{1}{N} \sum_{k=1}^{N} I \left(h(X, \theta_k) = j \right),$$

where I denotes the indicator function. The ensemble predicts the class at X by

$$\hat{h}(X) = \arg\max_j \hat{Q}_j(X).$$

The definition of Q_j can be extended to non-random classifiers by defining Q_j to be 1 if the classifier predicts class j and 0 otherwise. For example, a single-nearest-neighbor classifier results in Q_j being 1 in a voronoi region around each class j example from the training data, and zero elsewhere.

Most ensemble classifiers are so complicated that very little can be said about Q_j. However, PERT has a very simple structure which allows some theoretical development. It is clear that because PERT correctly classifies the training data we have

$$Q_j(x_i) = \begin{cases} 1 & \text{if } y_i = j \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, we can show (Zhao, 2000) that for PERT, Q_j is continuous over the feature space. In fact, if we let $x_{(1),k}, \ldots, x_{(n),k}$ denote the sorted values of feature k in the training data, then in any hyper-rectangle

$$I(i_1, \ldots, i_m) = [x_{(i_1),1}, x_{(i_{1+1}),1}] \times \ldots \times [x_{(i_p),p}, x_{(i_{p+1}),p}]$$

$Q_j(X)$ is a multilinear function, i.e. a linear function of each of its arguments, holding the other arguments fixed. These multilinear functions are continuous on the boundaries of the hyper-rectangles $I(i_1, \ldots, i_m)$.

In fact, given training data, Q_j could (in theory) be computed directly using a recursive equation given in Zhao (2000). In practice, it is faster to simply estimate Q_j using the ensemble, but PERT is one of only a few competitive ensemble classifiers that can also be be fit in a "one shot" manner.

To better understand PERT, we consider a generic method for classification using linear or nonlinear regression. The linear version dates back to Fisher (1936) and the nonlinear version to Breiman and Ihaka (1984). For a 2-class situation, let

$$y_i = \begin{cases} 0 & \text{class 1} \\ 1 & \text{class 2.} \end{cases}$$

Now consider fitting a linear or nonlinear regression to the training data, and classifying a new observation x as class 2 if $\hat{y} \geq 0.5$, where $\hat{y}$ is the regression estimate of y at x.

One way to think about PERT is that instead of using linear or nonlinear regression, PERT is fitting a blockwise multilinear interpolating surface. For regression purposes, such a surface would drastically overfit the data, but for classification purposes it appears to work very well. As Friedman (1997) points out, the best methods for regression do not usually translate to the best methods for classification, and PERT is an extreme example of this phenomenon. The natural question is whether there are other ways of interpolating that give even faster, more accurate, one-shot classifiers.

7 Concluding Remarks

PERT is a fast, easily coded ensemble classifier that gives results comparable to some of the best classifiers available. As well as being useful in its own right, it also helps in understanding a little more about ensemble classifier behavior. An interesting avenue for further research is to revisit the classification-through-regression paradigm to see whether better one-shot classifiers can be found.

One difficulty with PERT and other ensemble classifiers is that they are impossible to interpret. For some problems in machine learning, this is not too critical, but it is often vital in other research areas. However, even for methods that are generally considered "interpretable", there are problems. Traditional methods such as LDA may be interpretable but are often so inaccurate that one has to question the value of any interpretations that are made. Flexible methods such as CART appear to be interpretable but the interpretability can be misleading due to instability (change the data slightly, and completely different interpretations emerge). While interpreting ensemble methods is nontrivial, we believe it is the next major step that needs to be taken if their use is to become widespread.

Acknowledgements

Many thanks to Leo Breiman for frequent helpful discussions.

References

Breiman, L. & Ihaka, R. (1984), "Nonlinear Discriminant Analysis via ACE and Scaling," *Technical Report* **40**, Statistics Department, U.C.Berkeley.

Breiman, L. (1996), "Bagging Predictors," *Machine Learning* **26**, pp 123–140.

Breiman, L. (1998), "Arcing Classifiers," *The Annals of Statistics* **26**, pp 801–849.

Breiman, L. (1999), "Random Forests - Random Features," *Technical Report* **567**, Statistics Department, U.C.Berkeley, September 1999.
[ftp://ftp.stat.berkeley.edu/pub/users/breiman].

Buntine, W. and Niblett, T. (1992), "A Further Comparison of Splitting Rules for Decision-Tree Induction," *Machine Learning* **8**, pp 75–85.

Cutler, A. (1999), "Fast Classification Using Perfect Random Trees," *Technical Report* 5/99/99, Department of Mathematics and Statistics, Utah State University, May 1999.

Dietterich, T. (1998), "An Experimental Comparison of Three methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning* 40(2) pp 139–158.

Fisher, R. A. (1936), "The use of multiple measurements in taxonomic problems," *Annals of Eugenics* 7, pp 179–188.

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

Friedman, J. H. (1997), "On Bias, Variance, 0/1 Loss, and the Curse of Dimensionality," *J. Data Mining and Knowledge Discovery* 1, p 55.

Mingers, J. (1989), "An Empirical Comparison of Selection Measures for Decision-Tree Induction," *Machine Learning* 3, p 319–342.

Murphy, P. M., & Aha, D. W. (1994), *UCI Repository of machine learning databases* [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Quinlan, J. R., (1993), *C4.5: Programs for Empirical Learning*, Morgan Kaufmann, San Francisco, CA.

Zhao, G. (2000), *A New Perspective On Classification*, PhD Dissertation, Utah State University.