C4.5 Decision Forests

Tin Kam Ho
Bell Laboratories, Lucent Technologies
700 Mountain Avenue, 2C-425, Murray Hill, NJ 07974, USA

Abstract

Much of previous attention on decision trees focuses on the splitting criteria and optimization of tree sizes. The dilemma between overfitting and achieving maximum accuracy is seldom resolved. We propose a method to construct a decision tree based classifier that maintains highest accuracy on training data and improves on generalization accuracy as it grows in complexity. Trees are generated using the well-known C4.5 algorithm, and the classifier consists of multiple trees constructed in pseudo-randomly selected subspaces of the given feature space. We compare the method to single-tree classifiers and other forest construction methods by experiments on four public data sets, where the method's superiority is demonstrated. A measure is given to describe the similarity between trees in a forest, and is related to the combined classification accuracy.

1 Introduction

A persistent problem in using decision trees for classification is how to avoid overfitting a set of training data while achieving maximum accuracy. Some previous studies attempt to prune back a fully-split tree even at the expense of the accuracy on the training data. Other probabilistic methods allow descent through multiple branches with different confidence measures. None of these methods are optimized simultaneously for both training and testing sets.

In [5] we show that combining multiple trees constructed in randomly selected subspaces can achieve nearly monotonic increase in generalization accuracy while preserving perfect accuracy on training data that have no intrinsic ambiguity. This offers a better way to overcome the apparent dilemma of accuracy optimization and over-adaptation. In [5] we show that the method works well with binary trees that at each internal node split the data to two sides of an oblique hyperplane.

The validity of the *random subspace* method does not depend on the particulars of tree construction algorithms. Other types of splitting functions, including single-feature splits (axis-parallel hyperplanes), supervised or unsupervised clustering, distribution maps [7], and support vector machines can be readily applied. For easy repetition of results by others, in this paper we chose to use the C4.5 algorithm [13] (Release 8) to construct the trees. An advantage of the C4.5 algorithm is its versatility for multiple data types. This makes it easy to construct trees in selected subspaces for problems with binary, discrete, continuous, or categorical features. We will compare the accuracies of forests constructed by our method to those of single trees and forests constructed by other methods.

2 Methods for Systematic Construction of a Forest

The random subspace method is another example of improving accuracy through combining the power of multiple classifiers [8]. Here each classifier is a decision tree, and we call the combined classifier a *decision forest*.

Some previous attempts to construct multiple trees rely on certain heuristic procedures or manual intervention [12][15][16]. The number of different trees they can obtain is often severely limited. We emphasize that arbitrarily introduced differences between trees do not guarantee good combination performance – zero error on training data and a monotonic decrease in generalization error as the forest grows in the number of trees. For continuous improvement through combination, it is important to have a systematic procedure that can produce as many trees as desired while sufficient differences between the trees are maintained.

Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space. In each pass, such a selection is made and a subspace is fixed where all points have a constant value (say, 0) in the unselected dimensions. All samples are projected to this subspace, and a decision tree is constructed using the projected training samples.

For a given feature space of n dimensions, there are 2^n such selections that can be made, and with each selection a decision tree can be constructed. More different trees can be constructed if the subspace changes within the trees, that is, if different feature dimensions are selected at each split. The use of randomization in selecting the dimensions is merely a convenient way to explore the possibilities.

The trees constructed in each selected subspace are fully-split using all training data. They are hence perfectly correct on the training set by construction, assuming that there is no intrinsic ambiguity in the samples. When two samples cannot be distinguished by

the selected features, there will be ambiguities, but if no decision is forced in such cases, this does not introduce errors.

With each tree, the classification is invariant for points that are different from the training points only in the unselected dimensions. Thus each tree generalizes its classification in a different way. The vast number of subspaces in high dimensional feature spaces provides more choices than needed in practice. Hence, while most other classification methods suffer from the *curse of dimensionality*, we can take advantage of the high dimensionality that is often encountered in a real-world problem. And contrary to the Occam's Razor, our classifier improves on generalization accuracy as it grows in complexity. This property of our classifier is shared by all methods derived from the theory of stochastic discrimination (SD) [10] [11]. While some other SD methods start with highly projectable classifiers with minimum enrichment and seek optimization on uniformity, with decision forests, one starts with guaranteed enrichment and uniformity, and seeks optimization on projectability [9].

Another previously explored idea to build multiple classifiers originates from the method of cross-validation. There, random subsets are selected from the training set, and a classifier is trained using each subset. Such methods are also useful since overfitting can be avoided to some extent by withholding part of the training data. Two subsampling methods have been proposed: *bootstrapping*[2] and *boosting*[3]. In bootstrapping subsets of training samples are generated independently according to a uniform distribution, whereas in boosting they are generated sequentially, and a probability distribution is introduced to prefer those samples on which previous classifiers are incorrect. In boosting final decisions are combined with weights determined by accuracies of individual trees. Both these methods have been known to be superior to single C4.5 trees [14]. In our experiments we will compare accuracies of forests built by our *subspace* method to those built by such *training set subsampling* methods.

3 Decision Combination

The decisions of n_t individual trees are combined by averaging the conditional probability of each class at the leaves. For a point x, let $v_j(x)$ be the terminal node that x falls into when it descends down tree T_j ($j = 1, 2, ..., n_t$). Given this, let the probability that x belongs to class c ($c = 1, 2, ..., n_c$) be denoted by $P(c|v_j(x))$.

$$P(c|v_j(x)) \;=\; \frac{P(c, v_j(x))}{\sum_{k=1}^{n_c} P(c_k, v_j(x))}$$

can be estimated by the fraction of class c points over all points that are assigned to $v_j(x)$ (in the training set). Notice that in this context, since the trees are fully split, most terminal nodes contain only a single class (except for abnormal stops that may occur in some tree construction algorithms) and thus the value of the estimate $\hat{P}(c|v_j(x))$ is almost always 1. The discriminant function is defined as

$$g_c(x) \;=\; \frac{1}{n_t} \sum_{j=1}^{n_t} \hat{P}(c|v_j(x))$$

and the decision rule is to assign x to class c for which $g_c(x)$ is the maximum. For fully-split trees, the decision obtained using this rule is equivalent to a majority vote among the classes decided by each tree.

It is obvious that the discriminant preserves 100% accuracy on the training set that has no intrinsic ambiguity. For an unseen point, $g(x)$ averages over the posterior probabilities that are conditioned on reaching a particular terminal node. Geometrically, each terminal node defines a neighborhood that includes all points assigned to that node in the chosen subspace. By averaging over the posterior probabilities in these neighborhoods (decision regions), the discriminant approximates the posterior probability for a given x in the original feature space. This is similar to other kernel-based techniques for estimating posterior probabilities, except that here the kernels are of irregular shapes and sizes, and do not necessarily nest.

Because of this, the discriminant is applicable to any algorithm that partitions the feature space into regions containing only or mostly points of one class (for instances, partitions by learning vector quantization [4] and nearest-neighbor matching [6]). The analytical properties of the function and its several variants have been studied extensively by Berlind [1]. Essentially, accuracy of the function is shown to be asymptotically perfect (as number of component classifiers increases) provided that a number of conditions on enrichment, uniformity, and projectability are satisfied.

4 Independence Between Trees

For a forest to achieve better accuracy than individual trees, it is critical that there should be sufficient independence or dissimilarity between the trees. There have been few known measures for correlation between decision trees. The difference in combined accuracy of the forest from those of individual trees gives strong but indirect evidence on their mutual independence.

A simple measure of similarity between two decision trees can be the amount that their decision regions overlap. On fully split trees each leaf represents a region labeled with a particular class. On trees that are not fully split the regions can be labeled with the dominating class (ties broken arbitrarily). Given that, regardless of the structure of the trees, we can consider trees yielding the same decision regions equivalent (Figure 1). The similarity of two trees is then the total volume of the regions labeled with the same class by both trees.

Given two trees t_i and t_j, let their class decisions for a point x be $c_i(x)$ and $c_j(x)$ respectively, we define

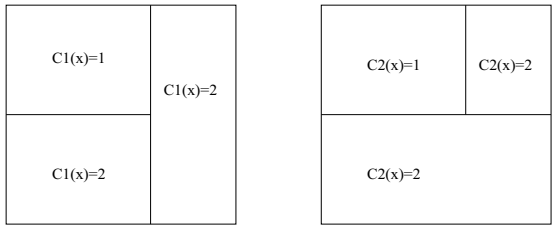


Figure 1: Two trees yielding the same decision regions in a 2-dim feature space ($Ci(x) = j$ if tree i decides that x is in class j).

tree agreement $s_{i,j}$ to be

$$s_{i,j} = \int_R p(x)dx,$$

where $R = \{x|c_i(x) = c_j(x)\}$, and $p(x)$ is the probability density function of x.

Given two decision trees, the volume of the overlapping decision regions is determined and in theory can be calculated exactly. However, depending on the form of the splitting function and the dimensionality of the space, the calculation could be difficult to carry out. Alternatively, one may get an approximation by Monte-Carlo integration, i.e., generating pseudo-random points to a sufficient density in the feature space and measure the agreement of their classification by the trees.

We consider an alternative estimate by using a fixed set of points. Assuming that the testing samples are representative for the given problem, we will use them to measure the tree agreement. This makes the estimation more computationally feasible and allows us to limit our concern within the neighborhood of the given samples.

It should be noted that severe bias could result if the samples for estimation are not chosen carefully. For instance, if the training samples are used in this context, since the trees are fully split and tailored to these samples, by construction the estimate of tree agreement will always be 1, and this will most likely be an overestimate. On the contrary, since the decision boundaries can be arbitrary in regions where there is no representative sample, if one uses pseudo-randomly generated points distributed over regions far outside those occupied by the given samples, one could risk severely underestimating the tree agreement that is relevant to the problem.

Using a set of n fixed samples and assuming a uniform distribution, the estimate $\hat{s}_{i,j}$ can be written as

$$\hat{s}_{i,j} = \frac{1}{n}\sum_{k=1}^{n} f(x_k),$$

$$\text{where} \quad f(x_k) = \begin{cases} 1 & \text{if } c_i(x_k) = c_j(x_k) \\ 0 & \text{otherwise.} \end{cases}$$

5 Experimental Results

A series of experiments were carried out using publicly available data sets provided by the Project Statlog [ftp.ncc.up.pt: pub/statlog/datasets]. We used all the four data sets that have separate training and testing data ("dna", "letter", "satimage", and "shuttle"). In each of these data sets, both training and testing samples are represented by feature vectors with integer components. Though some of the feature components are not necessarily numerical by nature (say, the features in the "dna" set are binary codings of a 4-element alphabet), the samples are treated as points in real-valued spaces. There are no missing values of any features in all four data sets. Table 1 lists the sizes of the training and testing sets provided in each problem.

Table 1: Specifications of data sets used in the experiments (c = number of classes; d = number of feature dimensions; |TR| = number of training samples; and |TE| = number of test samples).

dataset	c	d	|TR|	|TE|
dna	3	180	2000	1186
letter	26	16	15000	5000
satimage	6	36	4435	2000
shuttle	7	9	43500	14500

5.1 Forest built in subspaces versus a single decision tree

We first compared the accuracies of the forests constructed using our method against the accuracy of a single decision tree constructed using all features and all training samples.

In the experiments the C4.5 algorithm was essentially untouched. The forest construction procedure was implemented external to the C4.5 algorithm. That means, the features were randomly selected before the data were input to the algorithm. To mask off the unselected features, the corresponding entries in the attribute specification file (with a suffix ".names") were marked "ignore". Decision combination, in this case approximated by simple majority voting, was done using class decisions extracted from the output of the algorithm. Along with the use of publicly available data, these experiments can be easily duplicated.

In each run, we first used the original C4.5 algorithm to build an unpruned tree using all the features, and applied the tree to the testing set. Secondly a pruned tree was built (using the default settings) and tested similarly. We then constructed a decision forest, using the same C4.5 algorithm, but for each tree using only a half of the features that were randomly selected. Each tree was added to the forest and the accuracy of the forest on the testing set was measured. We continued until 100 trees were obtained in each forest. Figure 2 compares the accuracies of the forests against those obtained using a single (pruned or unpruned) C4.5 tree with the full feature vectors. Obvi-

ously the forests yield superior accuracies.

It should be noted that in some of these data sets the number of features is not very large (dna: 180; letter: 16; satimage: 36; shuttle: 9). But the method still works well with such relatively low-dimensional spaces. The only exception is that with the "shuttle" data, there are so few features to choose from that the forest does not display a great advantage over the single-tree classifier. To apply the method to very low dimensional data, we suggest first expanding the feature vectors by using certain simple functions of the original features. In many cases we found that the inclusion of pairwise sums, differences, products, or Boolean combinations (for binary and categorical features) of the raw features served as a good expansion.

5.2 Comparison to forests constructed by training set subsampling

In this experiment we compared our method of forest building to other methods where each tree was constructed using a randomly selected subset of the training data with all the features. We experimented with both bootstrapping and boosting methods that are of this category. In bootstrapping, subsets of the same size as the original training set were created by sampling with replacement in uniform probability. A tree was constructed with each subset. Boosting was done following the AdaBoost.M1 procedure described in [3].

Again, all trees were constructed by the C4.5 algorithm as in the previous experiment, and the sample selection procedures were implemented externally. Figure 2 shows the accuracies of such forests versus those built by random subspace selection. It can be seen that although the individual trees (say, the first tree in each forests) built by training set subsampling are sometimes more accurate, with more trees, the combined accuracies of the random subspace forests are superior. The results suggest that there is much redundancy in the feature dimensions in these data sets, but the samples are relatively sparse. Interestingly the results also indicate that the bootstrapping and boosting methods are very similar in performance.

Table 2: Average tree agreement of forests constructed by each method.

construction method	data set			
	dna	letter	satimage	shuttle
random subspace	0.7540	0.6595	0.8228	0.9928
bootstrapping	0.9081	0.8197	0.8294	0.9998
boosting	0.8969	0.7985	0.8205	0.9996

Tree Agreement

The differences in tree agreement in forests built using the subspace method and those built using the other two methods can be shown by our measure $\hat{s}_{i,j}$ estimated using the testing data. Table 2 lists the average tree agreement for each forest. Each estimate is averaged over all 4950 pairs among 100 trees. For the dna and letter data, the subspace method yielded very dissimilar trees compared to the other methods. For the other two data sets the differences are not as obvious. Again, differences between the bootstrapping and boosting methods are insignificant.

6 Conclusions

We described a method to systematically construct a decision forest. The method relies on a pseudo-random procedure to select components of a feature vector, and decision trees are constructed using only the selected feature components. Each tree generalizes classification to unseen points in a different way by invariances in the unselected feature dimensions. We also gave a decision combination function and a measure of similarity between trees.

We tested the method using a tree construction algorithm given in the C4.5 software packpage. Experiments were conducted using four publicly available data sets. Accuracies were compared to those of single trees constructed using the same tree construction algorithm with all the samples and full feature vectors. Significant improvements in accuracy were obtained using our method. Furthermore, it is clear that as the forests grow in complexity (measured in the number of trees), their generalization accuracy do not decrease, and maximum accuracies on the training sets are preserved. The results were also shown to be superior to those of forests built using training set subsampling including bootstrapping and boosting. For some datasets, significantly lower similarity between the trees resulted in the forests built by the subspace method.

References

[1] R. Berlind, *An Alternative Method of Stochastic Discrimination with Applications to Pattern Recognition*, Doctoral Dissertation, Department of Mathematics, State University of New York at Buffalo, 1994.

[2] L. Breiman, Bagging Predictors, *Machine Learning*, **24**, p.123, 1996.

[3] Y. Freund, R.E. Schapire, Experiments with a New Boosting Algorithm, *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[4] T.K. Ho, Recognition of Handwritten Digits by Combining Independent Learning Vector Quantizations, *Proceedings of the Second International Conference on Document Analysis and Recognition*, Tsukuba Science City, Japan, October 20-22, 1993, 818-821.

[5] T.K. Ho, Random Decision Forests, *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, Canada, August 14-18, 1995, 278-282.

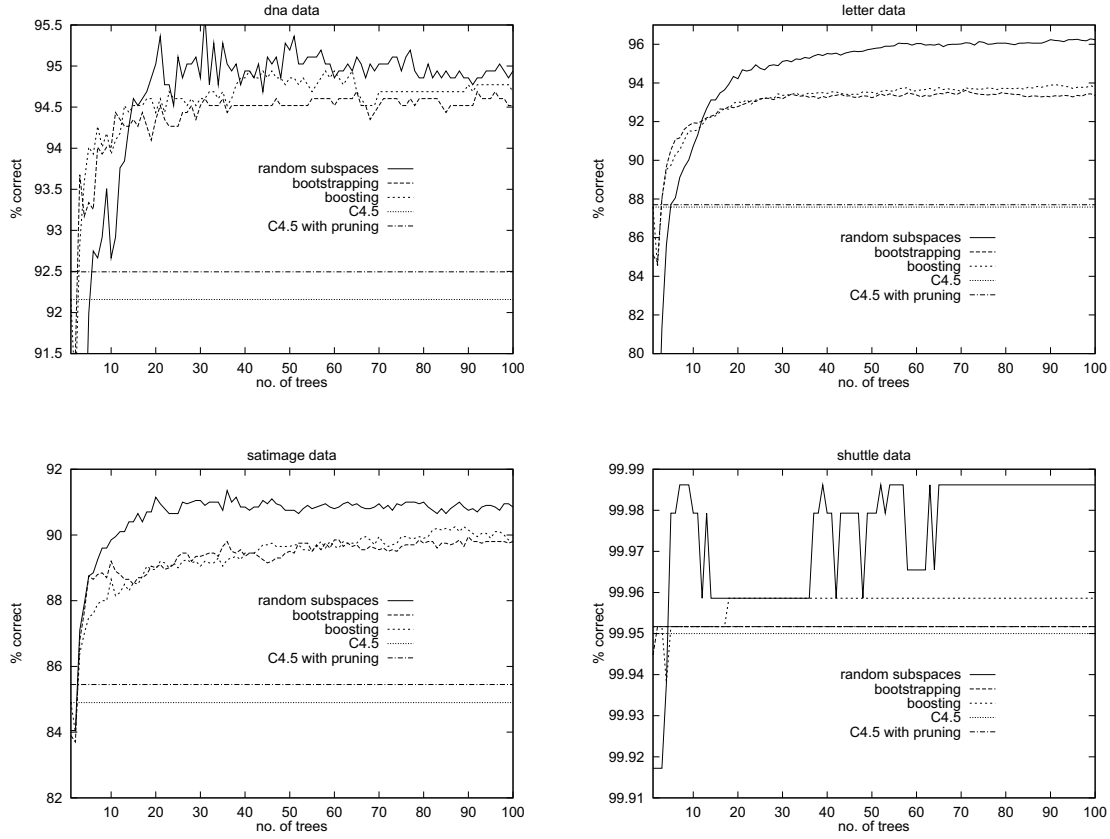


Figure 2: Comparison of accuracies: a single C4.5 tree, forests built by the random subspace method, and forests built by bootstrapping and boosting.

[6] T.K. Ho, Nearest Neighbors in Random Subspaces, *Proceedings of the 2nd International workshop on Statistical Techniques in Pattern Recognition*, Sydney, Australia, August 11-13, 1998.

[7] T.K. Ho, H.S. Baird, Pattern Classification with Compact Distribution Maps, *Computer Vision and Image Understanding*, **70**, 1, April 1998.

[8] T.K. Ho, J.J. Hull, S.N. Srihari, Decision Combination in Multiple Classifier Systems, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **16**, 1, January 1994, 66-75.

[9] T.K. Ho, E.M. Kleinberg, Building Projectable Classifiers of Arbitrary Complexity, *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, Austria, August 25-30, 1996, 880-885.

[10] E.M. Kleinberg, Stochastic discrimination, *Annals of Mathematics and Artificial Intelligence*, **1**, 1990, 207-239.

[11] E.M. Kleinberg, An overtraining-resistant stochastic modeling method for pattern recognition, *Annals of Statistics*, **4**, 6, 1996, 2319-2349.

[12] S.W. Kwok, C. Carter, Multiple Decision Trees, in R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer (eds.), *Uncertainty in Artificial Intelligence*, **4**, Elsevier Science Publishers B.V. (North-Holland), 1990.

[13] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[14] J.R. Quinlan, Bagging, Boosting, and C4.5, *Proceedings of 13th National Conference on Artificial Intelligence*, Portland, Oregon, August 4-8, 1996.

[15] S. Shlien, Multiple Binary Decision Tree Classifiers, *Pattern Recognition*, **23**, 7, 1990, 757-763.

[16] S. Shlien, Nonparametric Classification Using Matched Binary Decision Trees, *Pattern Recognition Letters*, **13**, February, 1992, 83-87.