CLASSIFICATION OF SIGNAL DATA USING DECISION TREES

Audrey Le, Julie Ngan, and Janna Shaffer

Decision Tree Group
Department of Electrical and Computer Engineering
Mississippi State University, Mississippi State, Mississippi 39762
{le, ngan, shaffer}@isip.msstate.edu

ABSTRACT

Abstract - **In this project, we present the use of a decision tree as a method of classifying different signal data. Two specific type of signal data that we attempt to classify are the scenic beauty values of forestry images and the pronunciations of proper nouns. The software to generate decision trees is developed, and decision trees are constructed for each set of the data using three decision tree styles: Bayesian, C4, and CART. Evaluations are conducted on the two data sets using each of these three tree styles. The results of the experiments are compared with IND [1], a decision tree software package that is available for research purpose.**

1. BACKGROUND

Binary trees give an interesting and often illuminating way of looking at data in classification or regression problems. Unlike many other statistical procedures which were moved from pencil and paper to calculators and computers, the use of trees was unthinkable before computers. In the last few decades, there have been much research on the use of decision trees to solve classification problems. One important feature of decision trees is their capability to break down a complex decision-making or classifying problem into a set of simpler problems. The goal of a decision tree classifier is to draw a conclusion through the breaking down and solving of these sub-problems that would resemble the intended desired solution.

1.1. Historical Background

The idea of using decision trees to identify and classify items or cases was originated to the work of Hunt in the pioneering book *Experiments in Induction* [2] that describes extensive experiments with several implementations of concept learning systems. The first real system was developed in the early 1960s when Morgan and Sonquist [3] developed the AID (Automatic Interaction Detection) program at the Institute for Social Research, University of Michigan. The ancestor classification program is THAID [4], developed at the institute in the early 1970s by Morgan and Messenger.

During the same time, both Breiman [5] and Friedman [6] independently started their work on using tree methods in classification. Later, they joined force with Stone and Olshen in the development of a tree structured methodology and theoretical framework in classification. CART (Classification and Regression Trees) was written in conjunction of their efforts [7] to solve classification problems.

In the meantime, based on Hunt's idea of concept learning systems, Quinlan developed a decision tree software package called ID3 (Induction of Decision Tree) [8] in 1979. ID3 is a recursive partitioning greedy algorithm which made use of information theoretic approaches. In 1992, the basic principles that underpin ID3 have evolved into the development of C4.5, which has incorporated some new discoveries and ideas developed after ID3 was released.

IND is another decision tree software package written by Wray Buntine [9] in 1992. It is developed as part of a NASA project to semi-automate the development of data analysis and modeling algorithms using artificial intelligence techniques. IND includes standard algorithms from Brieman's CART and Quinlan's ID3 and C4. It also introduces the use of Bayesian and minimum length encoding methods for growing trees and graphs.

Another decision tree package is OC1 (Oblique Classifier 1) which is developed by Sreerama Murthy in 1993 [10]. It is designed for applications where instances have numeric continuous feature values.

1.2. Applications

Depending on the problem, the basic purpose of a classification study can be either to produce an accurate classifier or to uncover the predictive structure of the problem. Decision trees are used in many disciplines and in various application domains for data exploration and classification. They are capable in approximating global complex decision regions (especially in high-dimensional spaces) by the union of simpler local decision regions at various levels of the tree [11]. Moreover, in a decision tree classifier, a sample is tested against only certain subsets of classes, thus eliminating unnecessary computations in many conventional single-stage classifiers [12].

Because of its flexibility and versatility, decision tree classifiers have been used in many research related to classification problems. In astronomy, decision trees are used in star-galaxy classification [13] and identification of comic rays [14]. In medicine, decision trees are used for detecting thyroid disorders [15] and breast cancer diagnosis [16]. In object recognition, tree-based classification has been used for recognizing three dimensional objects [17]. In physics, decision trees have been used for the detection of physical particles [18].

1.3. Motivation

In the last few years, the Institute of Signal and Information Processing (ISIP) has been conducting research to estimate scenic beauty values for forestry images [19] and to automatically generate proper noun pronunciations [20]. Different classification methods have been applied to attempt to improve the current solution to these problems. However, currently existing methods still have very high error rates and are not satisfactory.

The versatility and usefulness of decision trees in various disciplines motivates us to investigate the functionality of decision trees for these problems. However, not many public domain software packages are available, and those that are available have many limitations. In particular, IND is available for research purposes but it has difficulties deciphering the various phonetic symbols used in the proper noun data.

These factors lead us to the decision to implement our own decision tree software which would accommodate the data for the scenic beauty estimation and for the proper noun generation project. Our final software would be able to handle more complicated data types and be released as a public domain software package on our website.

2. INTRODUCTION

2.1. Terminology

Before the discussion of decision tree construction, we briefly introduce some basic definitions and terminology used in the discussion so that it will facilitate the reader’s understanding.

A tree is a connected, acyclic, undirected graph, with a root node. An ordered tree is a tree in which the children of each node ordered (normally from left to right) [21].

A binary tree is an ordered tree such that each child of a node is distinguished either as a left child or a right child and no node has more than one left child nor more than one right child.

A decision tree is built from a training set, which consists of objects. Each object is completely described by a set of attributes and a class label. Attributes can have ordered or unordered values. An example of an ordered value is an integer or a real value, while a Boolean value is an example of an unordered value.

A decision tree contains a root node, zero or more internal nodes (all nodes except the root and the leaves), and one or more leaf nodes (terminal nodes with no children). For a binary decision tree, the root node and all internal nodes have two child nodes. All non-terminal nodes contain splits.

A decision node is any non-terminal which contains some questions to be asked on a single or multiple attribute values, with one branch and subtree for each possible outcome of the test. A decision tree can be used to classify a case by starting at the root of the tree and moving through it until a leaf is encountered.

At each nonleaf decision node, the case's outcome for the test at the node is determined and attention shifts to the root of the subtree corresponding to this outcome. This process proceeds until a leaf is encountered. The class that is associated with the leaf is the output of the tree.

A class is one of the categories which cases are to be assigned at each leaf node. The number of classes is finite and their values must be established beforehand. The class values must be discrete.

Attributes are a collection of properties containing all the information about one object or case. Unlike class, each attribute may have either discrete or continuous values. A decision tree is built based on the attribute values of the training data. Therefore, the attribute values of all cases might not be the same, but the cases should have the same attributes.

A univariate decision tree is one in which the test at each internal node splits the node using a single attribute. A multivariate decision tree is one in which the test at each internal node splits the node using several attributes.

An object is misclassified by a tree if the class label output by the tree does not match the object's class label. The proportion of objects correctly classified by the tree is called accuracy and the proportion of objects incorrectly classified by the tree is called error [21].

2.2. Basics of Decision Tree Construction

To construct a decision tree, the tree is first grown to completion so that the tree partitions the training sample into terminal regions of all one class [22]. Tree construction uses the recursive partitioning algorithm, and its input requires a set of training examples, a splitting rule, and a stopping rule.

The partitioning of the tree is determined by the splitting rule and the stopping rule determines if the examples in the training set can be split further. If a split is still possible, the examples in the training set are divided into subsets by performing a set of statistical tests defined by the splitting rule. The test that results in the best split is selected and applied to the training set which divides the training set into subsets. This procedure is recursively repeated for each subset until no more splitting is possible.

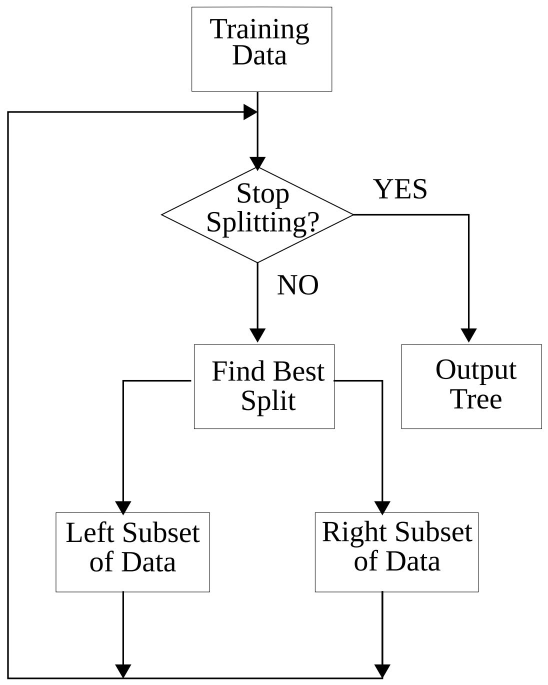

Figure 1. Flow chart Used for Splitting.

The splitting rules usually involve an exhaustive search in finding the best split. A statistical value is obtained for every possible split of all attributes at each node.Generally, the goal in splitting is to maximize or minimize the results from the statistical tests to find the best split. Figure 1 summarizes the tree building process.

Our implementation is based on the conventional method where a test involves just one attribute. We choose this method because this makes the tree easier to understand and sidesteps the combinatorial explosion that results if multiple attributes can appear in a single test [23]. The standard test on a discrete attribute for a binary tree is to generate a branch for cases containing one value of the attribute and the other branch for other cases containing other values, which is also known as the boolean combinations of attributes [24]. A more complex test, based on a discrete attribute, can be applied where the possible values are allocated to a variable number of groups with one outcome for each group rather than each value [25].

The test on continuous attributes is quite similar to discrete values even though they contain arbitrary thresholds. Since there are finite number of values in the training cases, we can sort the values for a particular attribute in order and split the attributes using the midpoint of each interval [26]. Then the same statistical testing can be applied to the splitting and determine where the best split lies. In the selection of a threshold for a continuous attribute A. If there are N distinct values of A in the set of cases D, there are $N-1$ thresholds that could be used for a test on A. Each threshold gives unique subsets D_1 and D_2, so that the value of the splitting criterion is a function of the threshold.

One important feature with continuous attributes is its ability to choose the threshold t so as to maximize this value gives a continuous attribute A an advantage over a discrete attribute (which has no similar parameter that adjusts the partition of D), and also over other continuous attributes that have fewer distinct values in D. That is, the choice of a test will be biased towards continuously attributes with numerous distinct values.

For some algorithms, such as C4.5, instead of using the midpoint, the largest value exists in the data that does not exceed the midpoint of each interval is chosen as the threshold.

Stopping rules vary from application to application but multiple stopping rules can be used across different applications. One stopping rule is to test for the purity of a node. For instance, if all the examples in the training set in a node belong to the same class, the node is considered to be pure [7], and no more splitting is possible.

Another stopping rule is by looking at the depth of the node. The depth of a node in a tree is the length of the path from the root to that node [27]. If the splitting of the current node will produce a tree with a depth greater than a pre-defined threshold, no more splitting is allowed. Another common stopping rule is the example size. If the number of examples at a node is below a certain threshold, then splitting is not allowed.

It has shown that a fully grown tree always suffer from over-fitting, in the sense that nodes near the bottom of the tree represent noise in the sample. It is because a complete grown tree usually results in the memorization of all the training instances of the data of the tree. The removal of some subtrees can often increase predictive accuracy [28]. To overcome this problem, a second process is used to prune back the tree so that the tree represents a more generalized description of the classification problem.Some popular algorithms include the use of resampling or hold-out methods [29], or to approximate significance tests [30], or minimum encoding [31].

3. SPLITTING ALGORITHMS

The idea of finding splits of nodes gives rise to purer descendant nodes. Splitting rules are defined by specifying a goodness of split function $\phi(s,t)$, where s is the split and t is the object in the current node, defined for every node t, and $s \in S$, where S is a set of binary splits. At every t, the split adopted is the split s^* which maximizes $\phi(s,t)$ [7], or minimize the error rate. There are many algorithms that are used to split a set of objects. We will limit our discussion to 1) twoing criterion, 2) Bayes splitting rule, 3) information gain, and 4) gain ratio.

3.1. Twoing Criterion

The **twoing criterion** requires the selection at every node be divided into two superclasses so that the problem can be considered as a two-class problem. This criterion attempts to group together large numbers of classes that are similar in some characteristics near the top of the tree and attempts to isolate single classes near the bottom of the tree. It is an intuitive criterion that attempts to inform the user of class similarities [7].

The twoing criterion for any node t and split s into a left node, t_L, and right node, t_R is defined by

$$\Phi(s, t) = \frac{p_L p_R}{4}\left[\sum_j \left|p(j|t_L) - p(j|p_R)\right|\right]^2 \qquad (1)$$

The split that maximizes the twoing criterion at a node is determined as the best split for this node. For a discrete attribute, twoing investigates each possible combination of values resulting in two superclasses. For continuous attributes, the data is sorted and the midpoint between each data sample is used as the sample split. Once the twoing criterion is maximized, the split defined by this function is applied to the node to create two subsets of the data.

3.2. Bayesian Splitting Rule

Bayesian splitting rule is based on Buntine's Bayesian splitting algorithm [22]. It uses the standard recursive partitioning algorithm to divide the training sample space into subsets based on some attribute. For a set of possible tests, each test is applied to the current node, and the tree is split using these tests. The posterior probability contributed by the new leaves is calculated. The test that is chosen is the test that yields the maximum posterior probability W. The posterior probability is calculated by:

$$Max\left(Pr_{jk}(W_V) = \sum_{k=1}^{V}\sum_{j=1}^{C} n_{jk}\log\phi_{jk}\right) \tag{2}$$

where C is the number of classes, V is the number of partitions, and

$$\phi_{jk} = \frac{n_{jk}}{\sum_{j=1}^{C} n_{jk}} \tag{3}$$

and

$$n_{jk} = \frac{\text{number of class j in partition k}}{\text{number of elements in partition k}} \tag{4}$$

The calculation is done in logarithmic probability to avoid underflow.

3.3. Information Gain

Information gain is another method used in hierarchical partitioning of the feature space. There is a simple method proposed by Sethi and Sarvarayudu [32] for the hierarchical partitioning of the feature space. The method is non-parametric and based on the concept of average mutual information. More specifically, let the average mutual information obtained about a set of classes C_k from the observation of an event X_k, at a node k in a tree T be defined as

$$\begin{aligned} &I_k(C_k;X_k) \\ &= \sum_{C_k}\sum_{X_k} p(C_{ki};X_{kj})\log\left[\frac{p(C_{ki}/X_{kj})}{p(C_{ki})}\right] \end{aligned} \tag{5}$$

Event X_k represents the measurement value of a feature selected at node k and has two possible outcomes; these measurement values are compared with a threshold associated with that feature at that node.

Then, the average mutual information between the entire set of classes, C, and the partitioning tree, T, can be expressed as

$$I(C;T) = \sum_{k=l}^{L} p_k \cdot I_k(C_k;X_k) \tag{6}$$

where p_k is the probability of the class set C_k and L is the number of internal nodes in the tree T.

The probability of misclassification, p_e, of a decision tree classification T and the average mutual information $I(C;T)$ are also related as [32]

$$\begin{aligned} I(C;T) \le -\sum_{j=1}^{m} & [p(C_j)\cdot\log p(C_j)] \\ & + p_e\cdot\log p_e \\ & + (1-p_e)\cdot\log(1-p_e) \\ & + p_e\cdot\log(m-1) \end{aligned} \tag{7}$$

with equality corresponding to the minimum required average mutual information for a prespecified probability of error. Then a goal for design of the tree could be to maximize the average mutual information gain at each node k. The algorithm terminates, when the tree average mutual information, $I(C;T)$, exceeds the required minimum tree average mutual information specified by the desired probability of error. An alternative stopping criterion proposed by Talmon [33] is to test the statistical significance of the mutual information gain that results from further splitting a node.

Even though information gain provides quite good results, it has a deficiency of a strong bias in favor of tests with many outcomes. In other words, when one of the attributes contains unique information for all of the data, partitioning any set of training cases on the values of this attribute will lead to a large number of subsets, each containing just one case. Since all of these one-case subsets necessarily contain cases of a single class, the information gain will be maximal, but yet quite useless [23].

3.4. Gain Ratio

Gain ratio is used instead of information gain as the splitting rule in many algorithms. Gain ratio is basically a normalization of the information gain where the apparent gain attributable to tests with many outcomes is adjusted. In order to generate a normalization, we define the potential information generated by splitting a decision tree T into n subsets as split information, with the class set C_i, can be expressed as,

$$Split(C;T) = -\sum_{i=1}^{n} [p(C_i) \cdot \log p(C_i)] \qquad (8)$$

Then the split information conveys the information relevant to classification that arises from the division.

As a result, the gain ratio, $GainRatio(C;T)$, expresses the proportion of information generated by the split that would be useful in the classification, and it is defined as,

$$GainRatio(C;T) = \frac{Gain(C;T)}{Split(C;T)} \qquad (9)$$

Therefore, when this ratio is maximized, the information gain must be large compared to the average gain over all test examined.

Gain ratio has advantages over information gain in that the information gained by a test is strongly affected by the number of outcomes and is a maximal when there is one case in each subset [34]. On the other hand, the potential information obtained by partitioning a set of cases is based on knowing the subset into which a case falls. Therefore, the split information tends to increase with the number of outcome of a test. The gain ratio criterion assesses the desirability of a test as the ratio of its information gain to its split information. The gain ratio of every possible test is determined and, among those with at least average gain, the split with maximum gain ratio is selected.

4. PRUNING AND SMOOTHING ALGORITHMS

The recursive partitioning method of constructing decision tree subdivide the set of training cases until each subset in the partition contains cases of a single class. The resulting tree is often very complex and “overfits the data” by inferring more structure than is justified by the training classes. Therefore, the idea of tree pruning is introduced.

Pruning of the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. In the case of noisy data, zero probability can be found in leaf nodes. To obtain a better classification tree, sometimes smoothing is used instead of pruning. Smoothing of the decision tree is done by building multiple trees and averaging their values. In this project, we have limited our discussion of pruning and smoothing algorithms to the following:

4.1. Cost-Complexity Pruning

Cost-complexity pruning is also known as error complexity pruning. The idea behind cost-complexity pruning is to find the best compromise between tree complexity and its cost. The process begins by growing a tree until all nodes are pure. For example, consider a tree $T_{unpruned}$. Let t_L and t_r be any two terminal nodes in $T_{unpruned}$ coming from the same parent node. If the cost of the parent node is equal to the sum of the cost of each child, then prune this node.

$$R(t_{parent}) = R(t_L) + R(t_R) \qquad (10)$$

The cost of a node is defined as

$$R(t) = \frac{\sum_n x_n \notin j}{n} \tag{11}$$

Continue pruning this way until no pruning is possible. Call the resulting tree T_1. For all nodes in T_1, find the one node that minimizes the following equation:

$$\alpha = \frac{R(t) - R(T_1)}{|\tilde{T}_t| - 1} \tag{12}$$

The cost of a subtree is defined as

$$R(T_t) = \sum_{t \in T} R(t) \tag{13}$$

and its complexity as

$$|\tilde{T}| = \sum t_{terminal} \in T \tag{14}$$

The node that minimizes this cost-complexity parameter α will be the "weakest link" of subtree T_1. This subtree should be pruned, with T_2 being the resulting tree. Continue in this same manner until only the root node is left. A decreasing sequence of subtrees will result such that $T_1 > T_2 > \ldots > T_n$.

Once the sequence of subtrees is found by consecutively pruning off the weakest link, the problem is reduced to choosing the optimum subtree. Cross-validation is used to select this optimum subtree. In cross-validation, the original test data is divided by random selection into V subsets [7]. For a k-th fold cross-validation, the training set is divided into k subsets and each subset is used as the testing data to evaluate the performance of the tree built using the combination of the other sets as the training data. V trees are grown using the partition of the data containing $(V-1)/V$ test cases.

For each value of the cost-complexity value, α, let $T^{(v)}(\alpha)$ for $v = 1 \ldots V$, be the tree pruned using the specified cost-complexity parameter. There exists a test set of $1/V$ cases that each tree has not seen. This test set will be used to determine which tree to use. For each fixed value of the cost-complexity parameter α, determine the value of the honest estimate for misclassification with the equation:

$$R^{CV}(T(\alpha)) = \frac{1}{N} \sum_{i,j} C(i|j) N_{ij} \tag{15}$$

N_{ij} is the number of test cases j, classified as i. $C(i|j)$ is the cost for misclassifying class j as class i. The value of alpha for which this function is a minimum is considered the maximum cost-complexity parameter. The "right sized" tree will be the tree grown from the whole data set that corresponds to this value of α.

4.2. Bayesian Smoothing

The standard approach for classifying an example using a class probability tree is to send the example down to a leaf and then return the class probability at the leaf [35]. The Bayesian classifier uses a smoothing technique as described by Bahl, etc.[36] and Chou [37]. A further technique from Kwok and Carter [38] is to build multiple trees and use the benefits of averaging to arrive at possibly more accurate class probability estimates.

In the smoothing approach, the class probability vectors encountered at interior nodes along the way are averaged [39]. Given a particular tree structure T' grown as described by Bayes splitting rule, and given a pruned tree structure $pruned(T')$ obtained by pruning the tree structure T' in all possible ways, if the space given by the pruned tree structure $pruned(T')$ is restricted and the posterior on the tree structure is a muliplicative function over nodes in the tree, then the sum can be recursively calculated using the distributive law. The sum is computable in a number of step linear in the size of the tree. The sum takes the form of an average calculated along the branch traversed by the new example.

$$E_{T,\Phi|(\vec{x},\vec{c})}(Pr(c = d_j)|(x, T, \Phi_T))$$
$$\sum_{n \in tr(x,T')} \begin{bmatrix} Pr(\text{n is leaf}|(\vec{x}, \vec{c},\ \text{T'pruned})) \\ \times \dfrac{n_{j,n} + \alpha_j}{n_{0,n} + \alpha_0} \end{bmatrix} \tag{16}$$

where $traversed(x, T')$ is the set of nodes on the path traversed by the example x as it falls to a leaf, and $Pr(\text{n is leaf}|(\vec{x}, \vec{c}, \text{pruning of T'}))$ is the posterior probability that the node n in the tree T' will be pruned back to a leaf given that the "true" tree is a pruned subtree of T'. It is given by

$$\begin{aligned} & Pr(\text{n is leaf}|\vec{x}, \vec{c}, \text{ T'pruned}) \\ = & (CPr(leaf(n), \vec{x}, \vec{c}))/SPr(T', \vec{x}, \vec{c}) \\ & \times \Pi CPrnode(O)) \cdot SPr(B, \vec{x}, \vec{c})) \end{aligned} \tag{17}$$

where ancestors $ancestors(T', n)$ is the set of ancestors of the node n in the tree T', $child(O, x)$ is the set of children trees of the node O, [40]

$$SPr(T, \vec{x}, \vec{c}) = \sum_{S \in pruned(T)} Pr(S(T, \vec{x}, \vec{c})) \tag{18}$$

4.3. Pessimistic Pruning

Pessimistic pruning is developed by Quinlan [23], based on the idea of statistical correction. The resubstitution error, which is the error rate on pruning a subtree using the observation on the training set from which the tree was built, is estimated and adjusted to reflect this estimate's bias. Therefore, based on the estimated and adjusted resubstitution error, the tree is pruned. This is done by examining each nonleaf subtree, starting from the bottom of the tree. If replacement of this subtree with a leaf, or with its most frequently used branch, would lead to a lower predicted error rate, then the tree is pruned accordingly. Since the error rate for the whole tree decreases as the error rate of any of its subtrees is reduced, this process will lead to a tree whose predicted error rate is minimal with respect to the allowable forms of pruning.

More specifically, the pessimistic estimate is described as follows. Consider a leaf covering N training cases, with E of them classified incorrectly. The resubstitution error rate for this leaf is then E/N. If we define this result as the probability of error over the entire population of cases covered by this leaf, for a given confidence level CF, the upper limit on this probability can be found from the confidence limits for the binomial distribution, denoted by $U_{CF}(E, N)$. On the argument that the tree has been constructed to minimize the observed error rate, this upper limit is then equated as the predicted error rate at a leaf.

To simplify the accounting, error estimates for leaves and subtrees are computed assuming that they were used to classify a set of unseen cases of the same size as the training set. So, a leaf covering N training cases with a predicted error rate of $U_{CF}(E, N)$ would give rise to a predicted $N \times U_{CF}(E, N)$ errors. Therefore, for a subtree t with k leaves, and each of the leaves cover n training cases with none of them classified incorrectly, the predicted error for k_i would be $n \times U_{CF}(0, n)$. The predicted error E_t for the subtree would be

$$\sum_{i=1}^{n} k_i \tag{19}$$

Then if the subtree is replaced by a single leaf, it would cover the same number of training cases, N, but with error E, so the corresponding predicted errors $E_t{}^*$ would be $N \times U_{CF}(E, N)$. If $E_t{}^* \geq E_t$, then the existing subtree has a higher number of predicted errors, and it is pruned to a leaf.

5. ALGORITHM DESCRIPTIONS

5.1. CART

CART refers to the software created in 1984 by Breiman, et al. It is an acronym for classification and regression trees. CART constructs a binary decision tree by recursively partitioning the training data. Its intent is to grow a large tree to cover all of the training cases, and then prune down the tree to balance the error rate with size of the tree [40]. CART uses the twoing criterion for splitting and cost-complexity cross-validation for pruning.

5.2. Bayesian Classifier

Bayesian classifier is based on the assumption that all of the relevant probability values are known. The apriori probabilities are assumed to be known. The random variable X can be determined to what class it belongs to based on a decision rule of probabilities. In our implementation of the Bayes' classifier, we use Bayes splitting rule to build multiple trees and use smoothing to average the trees.

5.3. C4

C4 is a decision tree algorithm which as its origins in Hunt's [2] Concept Learning Systems by way of ID3. It is introduced by Quinlan [23] for inducing classification models from data. A typical C4 algorithm generates a decision tree using the gain ratio as splitting rule and the pessimistic pruning as the pruning rule.

The different splitting and pruning/smoothing rules used for each algorithm is summarized in Table 1.

Decision Tree Algorithm	Splitting Rule	Pruning/ Smoothing Rule
Bayesian	Bayesian	Bayesian Smoothing
CART	Twoing	Cost-complex-ity
C4	Gain Ratio	Pessimistic

Table 1: Summary of Decision Tree Algorithms.

6. DATA DESCRIPTION

6.1. USFS Scenic Beauty Estimation Database

The database contains 638 images obtained from the USFS (the United States Forestry Service) for algorithm research and development purpose. These images are taken from various sites over the US in the past few years and during various seasons. The database contains forty-two features extracted from the images. These features are the distributions of red, green, and blue in different hues ranging from 0 to 255 divided into ten subgroups, the percentage of short lines and long lines in each image, and the entropy of color distribution (red, green, and blue) in each image.

These features describe the scenic beauty values of the images. There are three classes of scenic beauty values: high scenic beauty estimate (HSBE), medium scenic beauty estimate (MSBE), and low scenic beauty estimate (LSBE). These categories are determined by the subjective scenic beauty estimates (SBEs) that were obtained from human subjects judging the scenic value of the images. The SBEs of the images were averaged and the standard deviation was computed. Low scenic beauty falls below one standard deviation from the mean, medium scenic beauty falls between one standard deviation from the mean, and high scenic beauty falls one standard deviation above the mean.

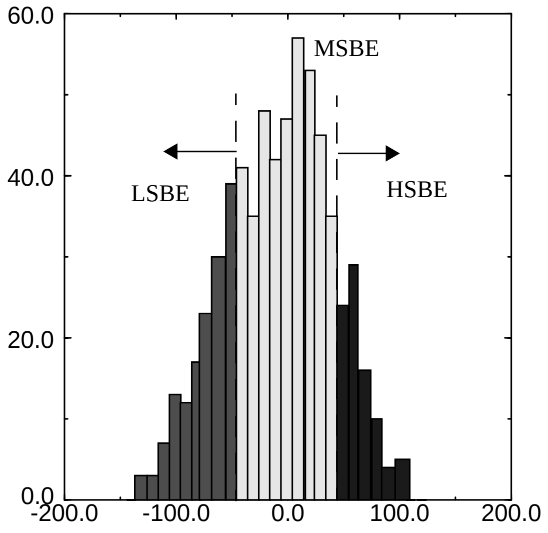

Figure 2. Scenic Beauty Estimation Histogram.

6.2. Proper Noun Pronunciation Database

The proper noun data consists of a comprehensive public domain pronunciation dictionary of people's last names (surnames). These last names includes 18,494 surnames from a diversity of ethnic origins and 25,648 corresponding pronunciations. The data, collected from a variety of sources, represents a reasonable mix of commonly found surnames, surnames with infrequent occurrence, and surnames that are known to present problems for letter-to-sound conversion due to complex morphology or difficult stress assignments [41].

The phonetic transcription was performed by hand using the Worldbet standards. Each surname was transcribed to a combination of phonemes to obtain all the correct pronunciations possible. Transcription of name pronunciations was a difficult task as the surnames derive from dozens of source languages having different stress patterns. A number of foreign names have both ethnic as well as anglicized pronunciations and individual pronunciations are often peculiar in defying any kind of typical text-to-speech rules.

7. IND Decision Tree Package

IND is a commercial decision tree package that was developed at NASA Ames Research Center by Wray Buntine. The first version was released in 1991. Earlier versions of IND were made available for research purposes only.The current IND package is available for $520 for domestic use and $1,040 for international use [42].

IND includes a collection of standard decision tree algorithms and offers a new set of decision tree algorithms. It re-implements the splitting and pruning algorithms used in Breiman's CART, Quinlan's ID3, and C4. It also introduces two new decision tree algorithms: Bayesian and MML. IND allows sophisticated options giving user direct and interactive control of the tree growing process.

IND grows the decision tree from data using a recursive partitioning algorithm. The training set consists of classes, each described by a set of attribute values. The class values can be alphanumeric and the attribute values can be alphanumeric or may be omitted. Prediction can then be done on new data or the decision tree printed out for inspection.

IND consists of four basic kinds of routines: data manipulation routines, tree generation routines, tree testing routines, and tree display routines. The data manipulation routines are used to partition a single large data set into smaller training and test sets. The generation routines are used to build classifiers. The test routines are used to evaluate classifiers and to classify data using a classifier. The display routines are used to display classifiers in various formats.

The routines in IND are written in C several of which controlled by shell scripts. IND has UNIX man entries for the routines.

Although IND allows sophisticated option controls giving direct control of the tree growing process, IND has many limitations. It is unable to handle more than 127 classes or more than 245 attributes, Moreover, each attribute can only have 254 values. The number of attributes can be modified but this involves manually modifying and re-compiling the source code. Furthermore, it can only handle alphanumeric characters and is unable to handle special characters such as &, >, @, ^, etc. which are used extensively in the proper noun phoneme data base. IND is fast and efficient, but readability is very low. The code is hard to read and poorly documented.

8. EXPERIMENTS

8.1. Digit Recognition Example

The digit recognition example is a good learning tool for understanding the operation of decision trees [7]. There are seven attributes indicating whether a light is on or off in the seven line digital display. These attributes are illustrated in Figure 3.

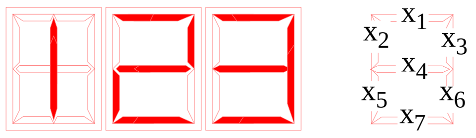

Figure 3. Attributes for the LED Problem.

Ten different classes are possible from the combination of these attributes. The training data used for the decision trees comes from Breiman's example [43]. The data contains two hundred test cases taken from a faulty digital display.

The following trees were grown on the digits data.

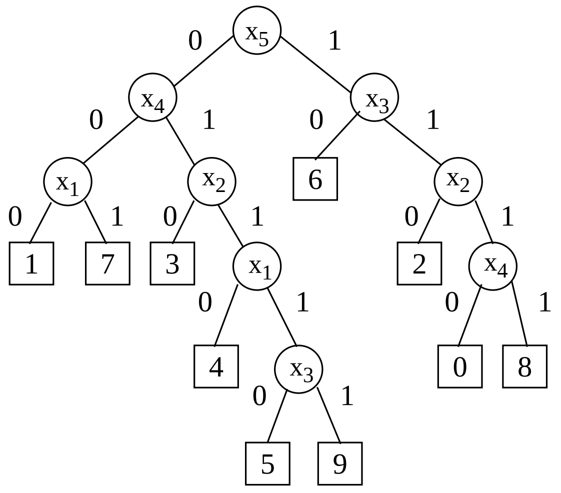

Figure 4. CART Digit Tree.

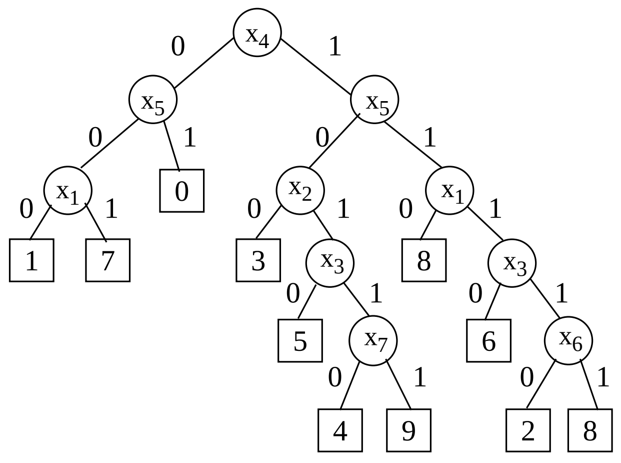

Figure 5. Bayes Digit Tree.

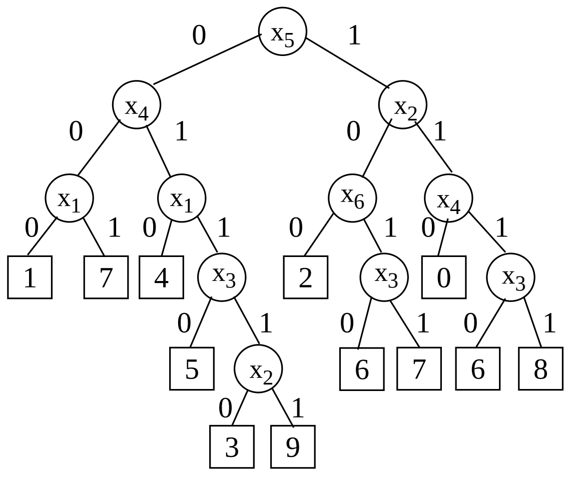

Figure 6. C4 Digit Tree.

8.2. Image Data -RGB+LL+ENT

The best system using other classification methods for the image data uses RGB, long lines and entropy as the feature space. In order to compare the results with the best system, we have used the same features, same training and testing data in our evaluation of our decision tree system. The result, in terms of error rate is then compared with the result of the current system and the results generated from using the IND decision tree software package.

8.3. Image Data - RGB+LL

In the evaluation of the performance of IND, we found that the best system results from using the RGB values and the long line as our feature space. Therefore, the same features and data sets were used to compare the results using the system we implemented.

9. EXPERIMENT RESULTS

In this section, we present the results obtained from the experiment we described previously using the IND decision tree software package and the decision tree software we have implemented.

9.1. LED

The LED data set comes from Breiman's [7] test data on the digit recognition problem. It consists of two hundred sample instances. Fifty samples were randomly chosen from this pool of two hundred samples, and the remaining one hundred fifty samples were used for training. Each sample belongs to one class and has seven attributes. The class represents one of the ten digits in the seven segment display 0-9, and the attributes indicate the on or off position of each segment for a total of seven segments. The data set was contrived such that 10% of the samples are faulty.

The data was trained using the three tree styles that were proposed in this paper and compared against the results using IND with the same algorithms. The results are shown in Table 2.

Algorithm	IND	DT
Bayes	41.5%	35.8%
CART	32.1%	35.8%
C4	28.3%	37.7%

Table 2: Misclassification Rate for the LED Problem.

Our decision trees perform comparably on the LED problem as those of IND. The difference between the performance of our trees and that of IND is not significant because the size of the training set is relatively small and the system is unable to learn determine the noisy data.

9.2. Image Data -RGB+LL+ENT

The image data set comes from USFS Scenic Beauty Database described above. The training and test set came from the official USFS training and test set one. The criterion for choosing the test set is to include at least one plot from each of the block, cover all treatments, and have a representative proportion of the number of LSBE, MSBE, HSBE images. The training set corresponding to the test set contains images from all the plots in the database excluding those in the test set. There are four training and test set pairs. The first training and test set pair was arbitrarily chosen to evaluate the performance of the tree. The test set contains 160 images and the training set contains 478 images.

Thirty-two features were extracted from the forty-two feature set. The features chosen include ten hues of red, ten hues of green, ten hues of blue, the percentages of long lines, and entropy.

Three decision tree styles were trained on this training data set and evaluated. The results are shown in

Algorithm	IND	DT
Bayes	63.1%	70.0%
CART	61.3%	70.0%
C4	59.3%	65.6%

Table 3: Misclassification Rate for the Image (RGB+LL+ENT) Problem.

For the image problem, all of our trees performed worse than those of IND. Like the LED problem the difference in performance between our trees and those of IND is not significant. The image features that were used as the attributes do not describe the class of the image.

9.3. Image Data - RGB+LL

The training and test set are the same as that described in section 10.2. except only thirty-one features were used for the attributes. The features are ten hues of red, ten hues of green, ten hues of blue, and the percentages of long lines. The three trees were built using the training data set and the test data was used to evaluate the performance of the tree. The results are given in Table 4.

Algorithm	IND	DT
Bayes	45.0%	35.6%
CART	41.9%	36.9%
C4	42.5%	81.9%

Table 4: Misclassification Rate for the Image (RGB+LL) Problem.

10. CONCLUSION

We have presented a simple decision tree package that performs comparable to IND but has significant advantages over IND.

Our decision tree package can handle much larger amount of training data as well as more classes than IND. The number of attributes and the values of each attribute are not restricted to a very small number. These requirements depend rather on the user’s system and memory availability. Our decision tree package allows tagging of attribute values and classes, enabling each attribute to be selected from the attribute file without having to reformat the training data. Therefore, our software is much more flexible in data selection and control than IND.

Our package is designed using object-oriented concept. All of the procedures can be readily transferred over to a different platform.

Our major goal of this work is to present an overview of the decision tree concepts and to implement public domain decision tree software. Our software is freely available for download and modification.

In this project, we described the decision tree software package we have implemented to classify signal data. Because of the limited availability of current decision tree software and many disadvantages in the existing systems, we aim to develop an integration of various decision tree algorithms with added flexibility. We have shown experimentally that our software can produce comparable results with the IND classifiers.

11. FUTURE WORK

The results of our work have raised additional issues that needed to be considered in future research directions.

11.1. Evaluation on Proper Noun Problem

The performance of our decision tree package has not been evaluated on the proper noun problem. The next step in our work is to evaluate our implementation of the decision tree on the proper noun pronunciation generation problem and compare the results with other systems.

11.2. Improvements to Current Implementation

Although our decision tree software is written in objected-oriented style and does not have many of the limitation that IND has, our system is significantly slower than that of IND. New data structures and faster implementations of the algorithms will be investigated to improve speed.

We will also investigate new methods to handle real valued attributes. In the original C4.5 for splitting algorithm for the continuous attribute values, it differs in choosing the largest value of the attribute in the entire training set that does not exceed the midpoint between two attributes, rather than the midpoint itself as the threshold. In later version of our software, we will include an option to choose the averaging method for continuous attributes.

11.3. Investigating New Methodology

The main goal of any classification task is to achieve a low misclassification rate on the new samples. Misclassification can be a result of an overtrained tree. Many a time, a decision tree is trained to fit a certain set of training examples, it lacks the generality to classify unseen samples. It is in our plan to explore various pruning methods and how they affect classification performance.

In addition, we will investigate further on smoothing as an alternative to pruning. We have observed from our results that if a class is poorly represented in the training samples, the probability of that class will be very low. It is not reliable to evaluate the result with this low probability. Moreover, we do not know if it is because the data is noisy or the data is rare. In the case of noisy data, the probability should be reduced zero whereas in the case of rare data, the probability should be low. We plan to investigate if the use of smoothing can possibly differentiate between noisy data and rare data by averaging over competing splits and over different training sets.

Currently the architecture of our decision tree is a univariate decision tree. It would be incorporated in our future research direction to explore a multivariate decision tree and evaluate its performance on our classification and identification problems.

12. REFERENCES

[1] W. Buntine and R. Caruana. “Introduction to IND Version 2.1 and Recursive Partitioning,” software manual, NASA Ames Research Center, Moffet Field, CA, 1992.

[2] E. B. Hunt, J. Marin, and P. J. Stone, *Experiments in Induction*, Academic Press, 1966.

[3] J. N. Morgan and J. A. Sonquist, “Problems in the Analysis of Survey Data, and a Proposal,” *Journal of Amer. Statist. Assoc.*, vol. 58, pp. 415-434. 1963.

[4] J. N. Morgan and R. C. Messenger, *THAID: a Sequential Search Program for the Analysis of Nominal Scale Dependent Variables.* Institute for Social Research, University of Michigan, Ann Arbor, 1973.

[5] L. Breiman, “Description of Chlorine Tree Development and Use,” technical report, Technology Service Corporation, Santa Monica, CA, 1978.

[6] J. H. Friedman, “A Recursive Partitioning Decision Rule for Nonparametric Classification,” *IEEE Trans. Computers,* vol C-26, pp 404-408. 1977.

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* Wadsworth International Group, 1984.

[8] D. Michie, “Discovering Rules by Induction from Large Numbers of Examples: a Case Study,” *Expert Systems in the Micro-electronic Age*. Edinburg University Press, Edinburgh, UK, 1979.

[9] W. Buntine, “Tree Classification Software,” *The Third National Technology Transfer Conference and Exposition*, Baltimore, MD, December 1992.

[10] S. K. Murthy, S. Kasif, and S. Salburg, *A System for Induction of Oblique Decision Tree*, vol. 2, pp. 1-32, 1994.

[11] L. N. Kanal, “Problem-Solving Methods and Search Strategies for Pattern Recognition,” *IEEE Trans. Patt Anal. Mach. Intell*., vol. PAMI-1, pp. 193-201, 1979.

[12] G. Landeweerd, T. Timmers, E. Gersema, M. Bins, and M. Halic, “Binary Tree Versus Single Level Tree Classification of White Blood Cells,” *Pattern Recognition*, vol. 16, pp. 571-577, 1983.

[13] N. Weir, U. Fayya, and S. Djorgovski, “Automated Star/Galaxy Classification for Digitized POSS-II,” *The Astronomical Journal*, vol. 109, pp.2401, 1995.

[14] S. Salzberg, R. Chandar, H. Ford, S. K. Murthy, and R. White, “Decision Trees for Automated Identification of Cosmic Rays in Hubble Space Telescope Images,” *Publications of the Astronomical Society of the Pacific*, 1994.

[15] P. E. File, P. I. Dugard, and A. S. Houston, “Evaluation of the Use of Induction in the Development of a Medical Expert System,” *Computers and Biomedical Research*, vol. 27, pp. 383-395, October 1994.

[16] O. Mangasarian, R. Setiono, and W. Wolberg. “Pattern Recognition via Linear Programming: Theory and Application to Medical Diagnosis,” In *SIAM Workshop on Optimization*, 1993.

[17] L. Spirkovska, “Three Dimensional Object Recognition Using Similar Triangles and Decision Trees,” *Pattern Recognition*, vol. 26, pp. 727, May 1993.

[18] D. Bowser-Chao and D. Dzialo, “Comparison of the Use of Binary Decision Trees and Neural Networks in Top Quark Detection,” *Physical Review D: Particles and Fields*, vol. 47, pp. 1900, March 1993.

[19] N. Kalidindi, A. Le, L. Zheng, H. Yaquin, V. Rudis, and J. Picone, “Scenic Beauty Estimate of Forestry Images,” *Proceedings of IEEE Southeastcon*, pp. 337-339, Blacksburg, Virginia, USA, April 1997.

[20] N. Deshmukh, M. Weber, and J. Picone, “Automated Generation of N-Best Pronunciations of Proper Nouns,” *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 283-286, Atlanta, Georgia, USA, May 1996.

[21] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1989.

[22] W. Buntine, *A Theory of Learning Classification Rules*, Ph.D. thesis, University of Technology, Sydney, 1991.

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

[24] G. Pagallo and D. Haussler, “Boolean Feature Discovery in Empirical Learning,” *Machine Learning*, vol. 5, pp. 71-99, 1990.

[25] P. E. Utgoff and C. E. Brodley. “Linear Machine Decision Trees,” *COINS Technical Report 91-10*, University of Massachusetts, Amherst, MA, 1991.

[26] A. Paterson and T. B. Niblett, *ACLS Manual*, Intelligent Terminals Ltd., Edinburgh, UK, 1982.

[27] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms,

Addison-Wesley Publishing Company, Reading, MA, 1974.

[28] J. Quinlan, “Induction of Decision Trees,” *Machine Learning*, vol. 1, pp. 81-106, 1986.

[29] S. Crawford, “Extensions to the CART algorithm,” *International Journal of Man-Machine Studies*, vol. 31, pp. 197-217, 1989.

[30] J. Mingers, “An Empirical Comparison of Pruning Methods for Decision-Tree Induction,” *Machine Learning*, vol. 4, pp. 227-243, 1989.

[31] J. Quinlan and R. Rivest, “Inferring Decision Trees Using the Minimum Description Length Principle,” *Information and Computation*, vol. 80, pp. 227-248, 1989.

[32] I. K. Sethi and G. Sarvarayudu, “Hierarchical Classifier Design Using Mutual Information,” *IEEE Trans Patt. Anal. Mach. Intell.*, vol. PAMI-4, pp. 441-445, 1982.

[33] J. L. Talmon, “A Multiclass Nonparametric Partitioning Algorithm,” in E. S. Gelsema and L. N. Kanal, Eds., *Pattern Recognition in Practice II*. Elsevier Science, Amsterdam, Netherlands, 1986.

[34] J. R. Quinlan, “Improve Use of Continuous Attributes in C4.5,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 77-90, 1996.

[35] S. K. V. Murthy, *On Growing Better Decision Trees from Data*, Ph.D. thesis, Johns Hopkins University, Baltimore, Maryland, 1996.

[36] L. Bahl, P. Brown, P. de Souza, and R. Mercer, “A Tree-Based Language Model for Natural Language Speech Recognition,” *IEEE Transactions Acoustic Speech and Speech Processing*, vol. 37, pp. 1001-1008, 1989.

[37] P. Chou, “Optimal Partitioning for Classification and Regression Trees,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp.170-198, 1991.

[38] S. Kwok and C. Carter, “Multiple Decision Trees,” in R. Shachter, T. Levitt, L. Kanal, and J. Lemmer, Eds, *Uncertainty in Artificial Intelligence 4*, North-Holland, 1990.

[39] P. C. Taylor and B. W. Silverman, “Block Diagrams and Splitting Criteria for Classification Trees,” *Statistics and Computing*, vol. 3, pp 147-161, Dec. 1993.

[40] W. Buntine, “Learning Classification Trees,” *Statistics and Computing*, vol. 2, pp 63-73, 1992.

[41] S. R. Safavin and D. Landgrebe. “A Survey of Decision Tree Classifier Methodology,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 3, pp. 660-674, May/June 1991.

[42] W. Buntine, “IND-the IND Decision Tree Package,” URL: http://cognac.cosmic.uga.edu/abstracts/arc-13188.html.

[43] Wei-Yin Loh and Nunta Vanichsetakul, “Tree-Structured Classification Via Generalized Discriminant Analysis,” *Journal of the American Statistical Association*, vol. 83, pp.715-725, 1988.