TREE PRUNING ALGORITHMS

SUBMITTED BY ARNAB DEB(510514047)
ISHITA MANDAL(510514066)
MADHUMITA PAUL(510514042)

MENTORED BY PROF. SOMNATH PAL

INTRODUCTION

Pruning- A machine learning technique that removes part of the tree which adds very little to the classification power of the tree.

Why pruning is required?

- Tight stopping criteria can cause underfitted tree.
- Loose stopping criteria causes overfitting to training sets.
- ‘Trading accuracy for simplicity’ - produce sufficiently accurate compact decision trees.

REDUCED ERROR PRUNING

Reduced Error Pruning

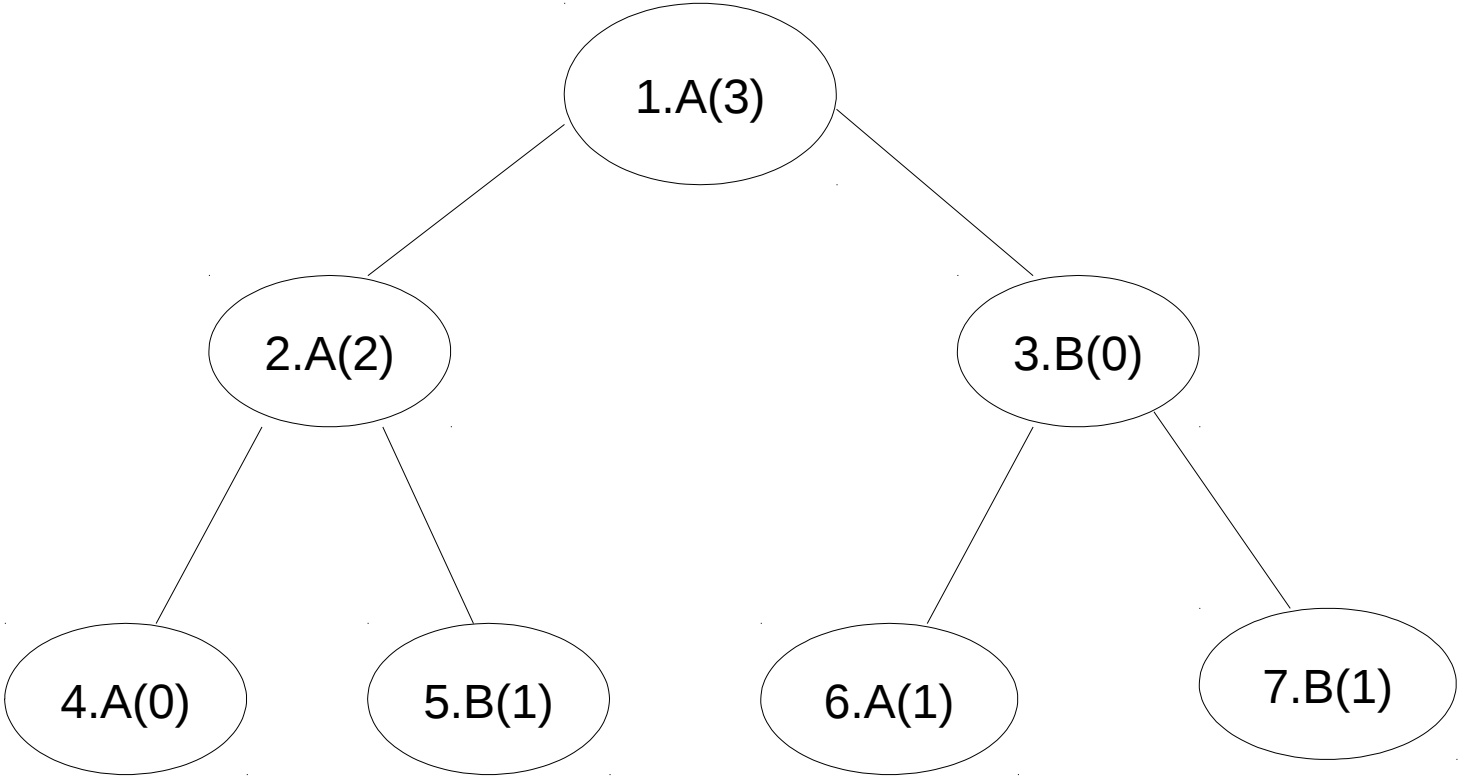
- Suggested by Quinlan.
- Its a bottom-up approach.
- Traversing through each internal node from bottom to determine whether replacing it with most frequent class reduces the error or not.
- Dataset divided into 3 sets: training set, pruning set and test set.

How it works?

- For a subtree S of the tree, if replacing S by a leaf does not make more prediction errors on the pruning set than the original tree, replace S by a leaf.
- This step is followed until no subtree of S possesses the above mentioned property.
- This procedure ends with the smallest accurate sub-tree with respect to a given pruning set.

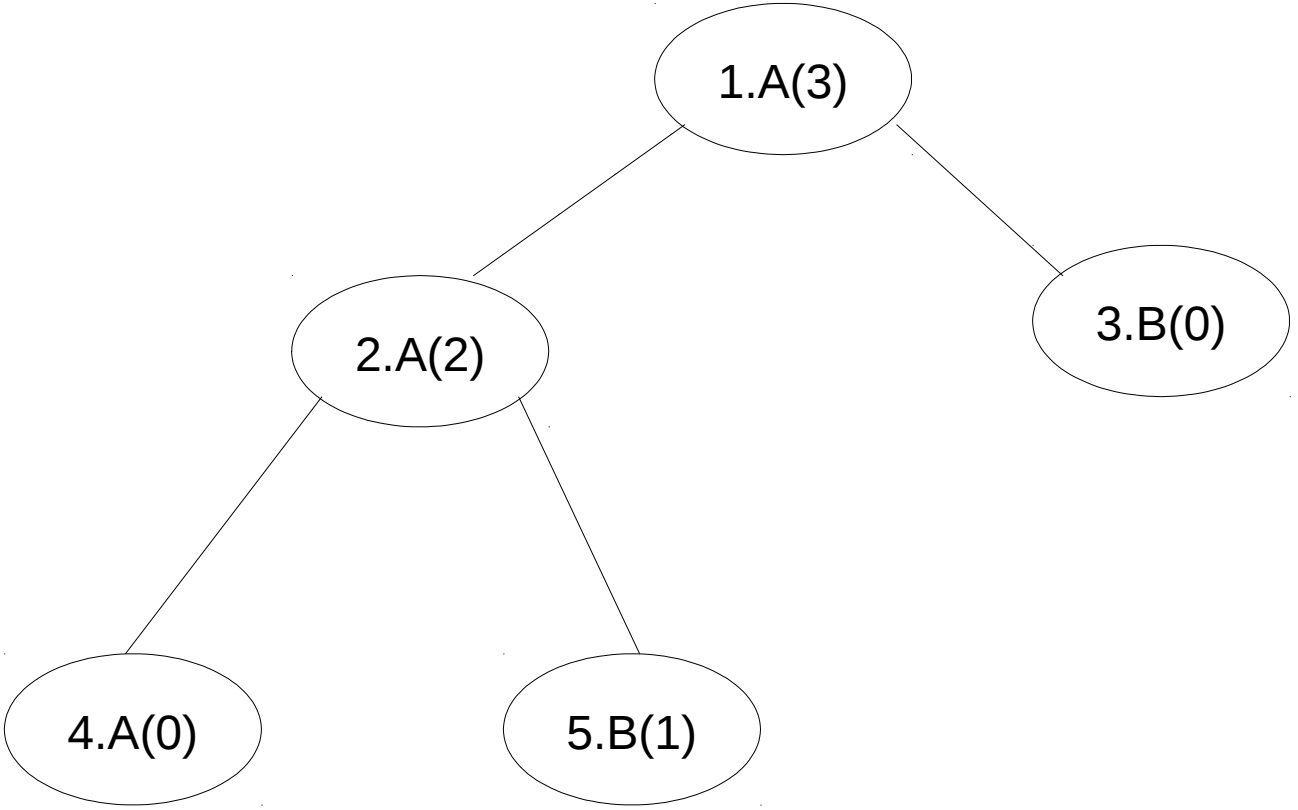
Example

- Let the A and B denote the classes. Number of misclassifications over pruning set at each node shown in brackets.



- Starting from the bottom, we see that node 2. as a leaf, makes 2 misclassifications, while as a subtree, the two branches 4. and 5. makes no or less error than 2. Thus, node 2. is not pruned.
- Looking at node 3., as a leaf, it makes no error in classification whereas as a subtree, the two branches 6. an 7. make errors. So, here subtree can be pruned to a leaf at 3.

- The pruned tree is:



COST COMPLEXITY PRUNING

INTRODUCTION

1. It is also known as weakest link pruning or error complexity pruning or CART (Classification and Regression Trees) Pruning.

2. Post-pruning algorithm proposed by Breiman,Olshen, Stone.

3. It takes into account both the number of errors and the complexity of the tree.

4. The size of the tree is used to represent the complexity of the tree.

Why cost complexity pruning??

- The problem we face is that the number of pruned subtrees may become very large and in that case it may not be feasible to compare them all on a test set.
- Remark: More precisely, let $|N_T|$ denote the number of leaf nodes of binary tree T . Then the number of pruned subtrees of T is floor(1.5028369^$|N_T|$) . So, for a tree with 25 leaf nodes (which is not unusually large) we would already have to compare 26,472 pruned subtrees

The basic idea of cost-complexity pruning is not to consider all pruned subtrees, but only those that are the “best of their kind”.

It consists of two steps:

1. Selection of a parametric family of subtrees of { T_0;T_1;…..; T_L } denoted by $T_{max(\alpha)}$, according to some heuristics.
2. Choice of the best tree according to an estimate of the true error rates of the trees in the parametric family.

- T_{i+1} is obtained from T_i by pruning those branches that show the lowest increase in α or apparent error rate per pruned leaf.
- Compute this value of α for each node t in T_i, and then select the “weakest links” i.e. the nodes for which α is the smallest.

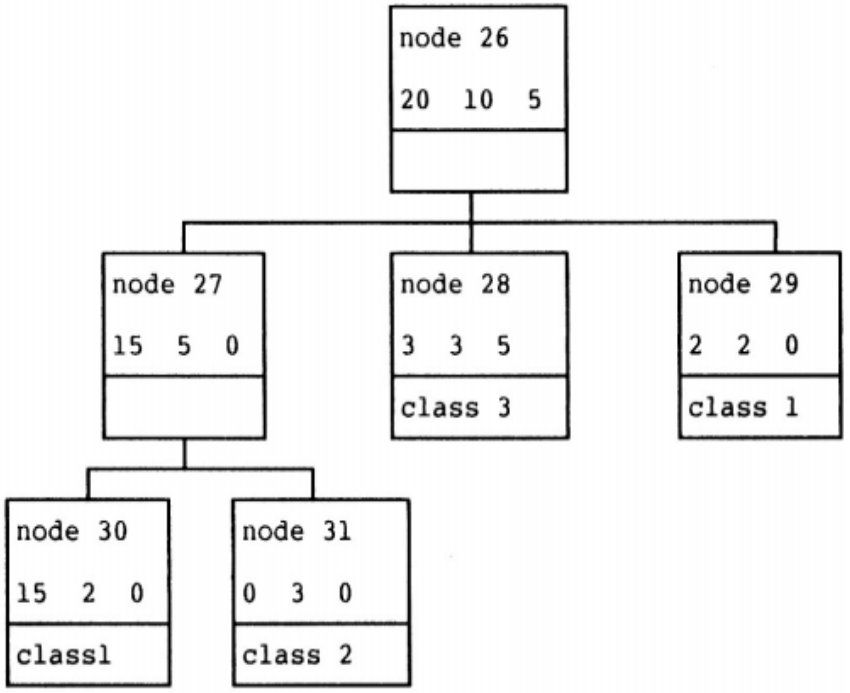
$$\alpha(t) = (R(t) - R(T_t))/(|N_T| - 1)$$

where, R(t)=error cost of node t

$R(T_t)$=error cost of subtree rooted at node t

.

Example



- Let t be node 26. The sub-tree rooted at node 26, T_{26} has 4 leaves, $N_T = 4$. If this is pruned, node 26 becomes a leaf of class 1, and so 15 of the 35 examples are then wrongly classified.
- Therefore, the error rate r(t) = 15/35. The proportion of data at t is p(t) = 35/200, so the error cost of node t is

$$R(t) = r(t)p(t) = \frac{15}{35} \times \frac{35}{200} = \frac{15}{200}$$

- If the node is not pruned, the error cost for the sub-tree is $R(T_t) = \Sigma R(i)$, for i = sub-tree leaves

$$= \frac{2}{17} \times \frac{17}{200} + 0 + \frac{6}{11} \times \frac{11}{200} + \frac{2}{4} \times \frac{4}{200} = \frac{10}{200}$$

- The cost complexity is the cost of one extra leaf in the tree, α. Then the total cost of the subtree is $R(T_t) + \alpha N_T$
- and of the node, if the sub-tree is pruned $R(t) + \alpha$.
- These are equal when

$$\alpha = \frac{R(t) - R(T_t)}{N_T - 1} = \frac{15/200 - 10/200}{4 - 1} = 5/600$$

- The two distinct ways of estimating the true error rate of each tree in the family: one based on cross-validation sets, and the other on an independent pruning set.

- If the given dataset is large enough, it is broken into a training set and a pruning set. The trees are constructed using the training set and evaluated on the pruning set.

- If the given dataset is not large enough, cross-validation methodology is used despite the computational complexity implications.

Minimum Error Pruning

A decision tree pruning method

Introduction

- Niblett and Bratko [1986] have developed a method to find the single tree which should, theoretically, give the minimum error rate when classifying independent sets of data.

Idea of Expected Error Rate

Assume a set of data with k classes; assume also that we have observed n examples, of which the greatest number, n_c are in class c. If we predict that all future examples will be in class c, what is the expected error rate—that is, proportion of wrong classifications. Niblett and Bratko show that the expected error rate, E_k is given by :

$$E_k = \frac{(n - n_c + k - 1)}{(n + k)}$$

Note that this assumes that all the classes are equally likely.

Algorithm

- At each (non-terminal) node in the tree, calculate the expected error rate if that sub tree is pruned so that the node becomes a leaf, using the numbers in each class at the node.

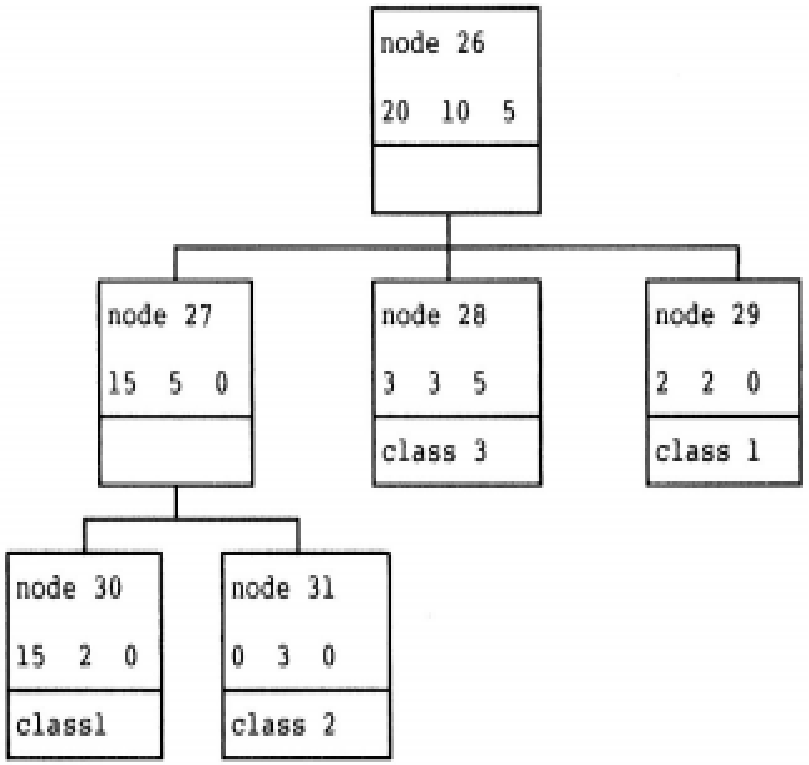
- Then calculate the expected error rate if the node is not pruned using the error rates for each branch, combined by weighting according to the proportion of observations along each branch.

Algorithm (cont.)

- The procedure is recursive since the error rate for a branch cannot be calculated until you know if the branch itself is to be pruned. If pruning the node leads to a greater expected error rate, then keep the sub tree; otherwise, prune it. The result should be a pruned tree that minimizes the expected error rate in classifying independent data.

An Example

Let us consider the case for Node 27



$$E_k = \frac{20 - 15 + 3 - 1}{20 + 3} = 0.304$$

$$E_k = \frac{17}{20}\left(\frac{17 - 15 + 3 - 1}{17 + 3}\right) + \frac{3}{20}\left(\frac{3 - 3 + 3 - 1}{3 + 3}\right) = 0.220 .$$

The expected error from pruning is greater, so do not prune.

Disadvantages

Theoretically, this is the ideal solution, since it minimizes the total expected error and does not require a separate testing set. However, there are several problems.

- The assumption of equally likely classes is seldom true in practice, although the effect of this is not clear.
- The procedure produces only a single tree. This is a disadvantage in the context of expert systems, where it is helpful if several trees, pruned to different degrees, are available.
- As the results will show, the number of classes strongly affects the degree of pruning, leading to unstable results.

Pessimistic Pruning

A decision tree pruning method

Introduction

- This is another method due to Quinlan[1986], which aims to avoid the necessity of a separate test data set.
- As has been seen, the misclassification rates produced by a tree on its training data are overly optimistic and, if used for pruning, produce overly large trees.
- Quinlan suggests using the continuity correction for the binomial distribution to obtain a more realistic estimate of the misclassification rate.

If $N(t)$ = number of training set examples at node t, and $e(t)$ = number of examples misclassified at node t, then

$$r(t) = \frac{e(t)}{N(t)}$$

is an estimate of the misclassification rate. The rate with continuity correction is,

$$r'(t) = \frac{e(t) + 1/2}{N(t)}.$$

For a subtree T_t the misclassification rate will be,

$$r(T_t) = \frac{\Sigma\, e(i)}{\Sigma\, N(i)}$$

where *i* covers the leaves of the sub-tree. Thus the corrected misclassification rate will be,

$$r'(T_t) = \frac{\Sigma\, (e(i) + 1/2)}{\Sigma\, N(i)} = \frac{\Sigma\, e(i) + N_T/2}{\Sigma\, N(i)}$$

where N_T is the number of leaves.

Here, $N(t) = \sum N(i)$ as they refer to the same set of examples. Therefore, the rates can be simplified to numbers of misclassifications:

$$n'(t) = e(t) + 1/2$$ for a node

$$n'(T_t) = \Sigma\, e(i) + N_T/2$$ for a subtree

Standard Error

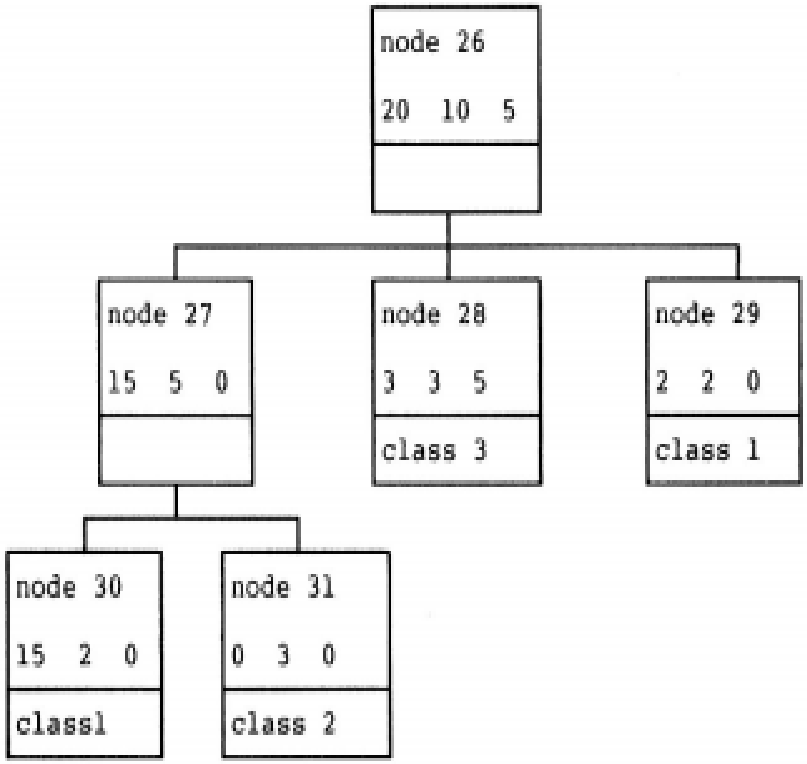
- With the training data, the subtree will always make fewer errors than the corresponding node, but this is not so when the corrected figures are used, since they depend on the number of leaves, not just the number of errors.
- However, it is likely that even this corrected estimate of the number of misclassifications made by the subtree will be optimistic. So the algorithm only keeps the sub-tree if its corrected figure is more than one standard error (as defined earlier) better than the figure for the node.

- The standard error for the number of misclassifications is derived from that given earlier for the rate of mis-classifications :

$$SE(n'(T_t)) = \sqrt{\frac{n'(T_t) \times (N(t) - n'(T_t))}{N(t)}}$$

So, Quinlan suggests pruning the subtree unless its corrected number of misclassifications is lower than that for the node by at least one standard error. The algorithm evaluates each node starting at the root of the tree. This means that it does not need to consider nodes that are in sub-trees which have already been pruned.

An Example


node 26
20 10 5
node 27
15 5 0
node 28
3 3 5
class 3
node 29
2 2 0
class 1
node 30
15 2 0
class1
node 31
0 3 0
class 2

For the example, number of corrected misclassifications at node 26,

$$n'(t) = 15 + 1/2 = 15.5,$$

and number of corrected mis-classifications for sub-tree,

$$n'(T_t) = (2 + 0 + 6 + 2) + 4/2 = 12.0$$

$$SE = \sqrt{\frac{12 \times (35 - 12)}{35}} = 2.8 .$$

Since, 12.0 + 2.8 = 14.8, which is less than 15.5, the sub-tree should be kept and not pruned.

ERROR BASED PRUNING

Overview

- It is an evolution of the pessimistic pruning.
- It is implemented in the well-known C4.5 algorithm.
- Uses all of the available labeled data for training and based on statistical confidence estimates.
- As in pessimistic pruning, the error rate is estimated using the upper bound of the statistical confidence interval for proportions.
- Allows to prune several subtrees by a leaf as well as replace the subtree by one of its branch as and when required.

Statistical Confidence Interval

$$\varepsilon_{UB}(T,S) = \varepsilon(T,S) + Z_{\alpha} \cdot \sqrt{\frac{\varepsilon(T,S) \cdot (1 - \varepsilon(T,S))}{|S|}}$$

where, T is the tree and S is the no. of available instances ,
ε UB (T, S) is the true error rate of the tree T and ε(T, S) is the observed error rate of the tree T.
Z is the inverse of the standard normal cumulative distribution and
α is the desired significance level.
Z is a factor that depends on α.

- The significance or confidence level is the Certainty Factor(CF).
- A higher CF, means the more certain we are that the true error rate lies in the observed error rate.
- A lower CF, means more errors than that occurred in training data, thus pruning is done.
- The default setting of CF is 25.

Technique

Let subtree(T,t) denote the subtree of T rooted at node t, S_t denote the no. of instances of S passing through the node t. Pruned(T, t) denotes the tree obtained by replacing the node t in T with a suitable leaf. Let maxchild(T, t) denote the most frequent child node of t.

We are going to consider 3 cases:

1. subtree(T,t) is left as it is.
2. subtree(T,t) is pruned.
3. subtree(T,t) is replaced by the subtree rooted at maxchild(T,t).

This technique traverses in a bottom-up approach through all the internal nodes and find the true error rate for each of the cases for a particular confidence level ,i.e.:

1. ε_{UB} (subtree(T,t), S_t)
2. ε_{UB} (pruned(subtree(T,t),t) , S_t)
3. ε_{UB} (subtree(T,maxchild(T,t)) , $S_{maxchild(T,t)}$)

The one with the lowest value is taken. According to that, a tree is either left as it is, or pruned at that node, or that node is replaced by the subtree rooted at the maxchild(T,t).

MINIMUM DESCRIPTION LENGTH PRUNING

Introduction

1. MDL was introduced by Jorma Rissanen in 1978.
2. It states that the best “theory” to infer from the set of data is the one which minimizes the sum of
    - Length of the theory
    - Length of the data when encoded using the theory as predictor for data.
3. This method measures the size of a decision tree by means of the number of bits required to encode the tree.

When a subtree is pruned?

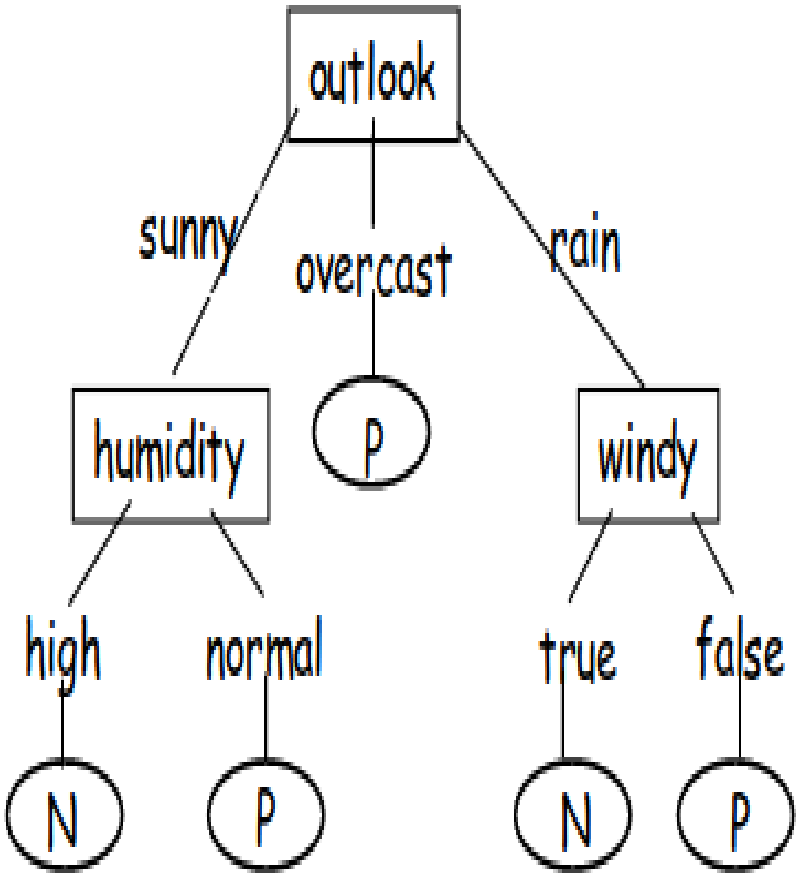
A subtree is pruned if

the description length of the classification of the subsets of instances in all the leaves of the subtree together with the description lengths of each path in the subtree >= the description length of the classification of the instances that fall in the current node.

The algorithm proceeds bottom-up from the leaves towards the root of the complete tree and for each of the internal nodes makes a decision whether to prune the subtree or not.

Data set from Quinlan(86)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N



Encoding Technique:

- We are interested in the problem of transmitting a string of 0's and 1's so that it will be cheaper to transmit strings which have only few 1's. (The ones will indicate the location of the exceptions.)
- We assume that the string is of length n, out of which k symbols are 1's and (n - k) of the symbols are 0's.
- $k \leq b$, where b is known as apriori upper bound on k. Typically we will either have $b = n$ or $b = (n + 1)/2$.

The procedure is:

- Transmit the value of k. This requires $\lg(b + 1)$ bits.
- Now that k is known, both know that there are only $({}^{n}C_{k})$ strings possible. Since all these possible strings are equally likely a priori, only $\lg(({}^{n}C_{k}))$ additional bits are needed to indicate which string actually occurred.
- The total cost for this procedure is thus

$$L(n,k,b) = \lg(b + 1) + \lg(({}^{n}C_{k})) \text{ bits.}$$

- We may consider coding in this manner the string in the last column of previous Table:

N, N, P, P, P, N, P, N, P, P, P, P, P, N

- Treating N as 0 and P as 1, we have n = 14, k = 9 and b = 14, for a total of

$$L(14,9,14) = \lg(15) + \lg(2002) = 14.874 \text{ bits.}$$

- This is larger than the “obvious” cost of 14 bits; **this coding scheme can save substantially when k is small.**

The generalization of this method to non binary classification problems would assign a cost of

$$L(n; k_1,k_2,k_3,\ldots\ldots k_t) = \lg(({}^{n+k-1}C_{k-1}).({}^{n}C_{k1,k2,k3,\ldots\ldots kt}))$$

to a string of length n containing k_t objects of class t, where $k = k_1 + \ldots + k_t$.

References

- Patil, Dipti D., V. M. Wadhai, and J. A. Gokhale. "Evaluation of decision tree pruning algorithms for complexity and classification accuracy." International Journal of Computer Applications 11.2 (2010).
- Esposito, Floriana, et al. "A comparative analysis of methods for pruning decision trees." IEEE transactions on pattern analysis and machine intelligence 19.5 (1997): 476-491.
- Rokach, Lior, and Oded Maimon. Data mining with decision trees: theory and applications. World scientific, 2014.
- Mingers, John. "An empirical comparison of pruning methods for decision tree induction." Machine learning 4.2 (1989): 227-243.
- Quinlan, J. Ross. "Simplifying decision trees." International journal of man-machine studies 27.3 (1987): 221-234.
- Quinlan, J. Ross, and Ronald L. Rivest. "Inferring decision trees using the minimum description length principle." *Information and computation* 80.3 (1989): 227-248
- Kononenko, Igor. "The minimum description length based decision tree pruning." Pacific Rim International Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 1998.