Unit – III
DECISION TREE LEARNING

3.1 INTRODUCTION

- Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.
- Learned trees can also be re-represented as sets of if-then rules to improve human readability. These learning methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

3.2 DECISION TREE REPRESENTATION

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.
- An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node. Figure *1* illustrates a typical learned decision tree. This decision tree classifies Saturday mornings according to whether they are suitable for playing tennis. For example, the instance

(Outlook = ***Sunny, Temperature*** = ***Hot, Humidity*** = ***High, Wind*** = ***Strong)***

would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance (i.e., the tree predicts that ***PlayTennis*** = ***no).***

- In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.
- For example, the decision tree shown in Figure **1** corresponds to the expression

(Outlook = ***Sunny*** □ ***Humidity*** = ***Normal)***
V ***(Outlook*** = ***Overcast)***
v ***(Outlook*** = ***Rain*** □ ***Wind*** = ***Weak)***

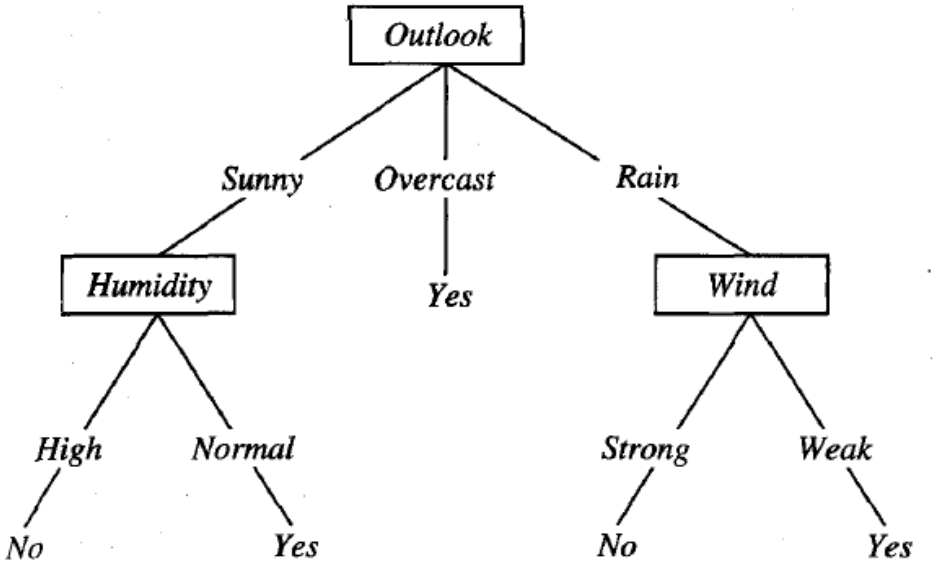


Fig 1: A decision tree for the concept *PlayTennis*.

3.3 APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

Decision tree learning is generally best suited to problems with the following characteristics:

a. Instances are represented by attribute-value pairs. Instances are described by a fixed set of attributes (e.g., ***Temperature)*** and their values (e.g., ***Hot).*** The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., ***Hot, Mild, Cold).***

b. The targetfunction has discrete output values. The decision tree in figure 1 assigns a boolean classification (e.g., ***yes*** or ***no)*** to each example. Decision tree methods easily extend to learning functions with more than two possible output values. A more substantial extension allows learning target functions with real-valued outputs, though the application of decision trees in this setting is less common.

c. Disjunctive descriptions may be required. As noted above, decision trees naturally represent disjunctive expressions.

d. The training data may contain errors. Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.

e. The training data may contain missing attribute values. Decision tree methods can be used even when some training examples have unknown values (e.g., if the ***Humidity*** of the day is known for only some of the training examples).

3.4 THE BASIC DECISION TREE LEARNING ALGORITHM

- Many algorithms have been developed for constructing the decision trees.
- The basic decision tree learning algorithm is ID3 which follows the top-down approach for constructing the decision tree.

3.1 Which Attribute Is the Best Classifier?

- The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree.
- For this, ID3 algorithm uses information gain measure. The attribute with the highest information gain is chosen for testing at a node.
- ***Information gain*** measures how well a given attribute separates the training examples according to the target classification.
- In order to define information gain precisely, a measure called ***entropy*** is used.
- Entropy measures the impurity of an arbitrary collection of samples. (i.e. it measures the homogeneity of samples).
- Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this Boolean classification is

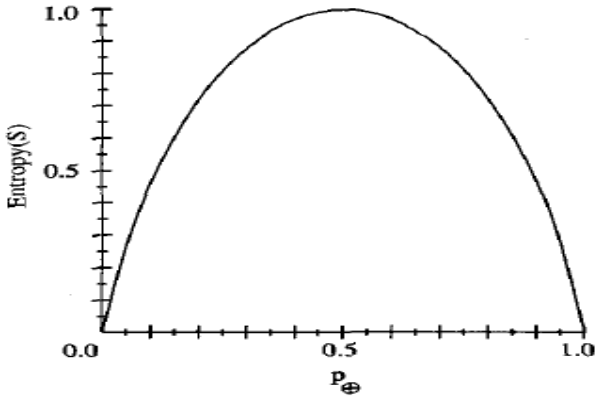
$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

where $p_{\oplus}$, is the proportion of positive examples in S and $p_{\ominus}$, is the proportion of negative examples in S.

- To illustrate, suppose S is a collection of 14 examples of some Boolean concept, including 9 positive and 5 negative examples. Then the entropy of S relative to this Boolean classification is

$$Entropy([9+, 5-]) = -(9/14) \log_2(9/14) - (5/14) \log_2(5/14)$$
$$= 0.940$$

- Notice that the **entropy is 0** if all members of S belong to the same class. Also note that the **entropy is 1** when the collection contains an equal number of positive and negative examples.
- If the collection contains unequal numbers of positive and negative examples, the entropy is between **0** and 1. The following figure shows this.



- If the target classification has c classes, then the entropy of ***S*** relative to this c-wise classification is defined as

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

- where p_i is the proportion of ***S*** belonging to class ***i.***
- Having been defined entropy, now we can define the information gain, ***Gain(S, A)*** of an attribute A, relative to a collection of samples ***S*** as

$$Gain(S, A) - Entropy(S) - Entropy_A(S)$$

Where $Entropy_A(S)$ is given as

$$\sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where ***Values(A)*** is the set of all possible values for attribute A, and ***S,*** is the subset of S for which attribute A has value v.

- As an example, let us construct the decision tree for the following data which shows training examples for the target concept ***PlayTennis***.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

First, compute ***Entropy(S)*** [S is the given data set. There are two classes yes = 9, No = 5].

$$Entropy([9+, 5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$
$$= 0.940$$

Next, compute the information gain of each attribute in the data set.
$Entropy_{Outlook}(S)$ is given as

$$\sum_{v \in Values(Outlook)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= \frac{5}{14}\left(-\frac{3}{5}log\frac{3}{5} - \frac{2}{5}log\frac{2}{5}\right) + \frac{4}{14}\left(-\frac{4}{4}log\frac{4}{4}\right) + \frac{5}{14}\left(-\frac{3}{5}log\frac{3}{5} - \frac{2}{5}log\frac{2}{5}\right)$$

$= 0.694$

Therefore

$$Gain(S, Outlook) = Entropy(S) - Entropy_{Outlook}(S)$$
$$= 0.940 - 0.694$$
$$= 0.246$$

Similarly,
$Entropy_{Temperature}(S)$ is given as

$$\sum_{v \in Values(Temperature)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= \frac{4}{14}\left(-\frac{2}{4}log\frac{2}{4}-\frac{2}{4}log\frac{2}{4}\right)+\frac{6}{14}\left(-\frac{4}{6}log\frac{4}{6}-\frac{2}{6}log\frac{2}{6}\right)+\frac{4}{14}\left(-\frac{3}{4}log\frac{3}{4}-\frac{1}{4}log\frac{1}{4}\right)$$

$= 0.911$

Therefore

$$Gain(S, Temperature) = Entropy(S) - Entropy_{Temperature}(S)$$

$$= 0.940 - 0.911$$

$$= 0.029$$

Similarly,

$Entropy_{Humidity}(S)$ is given as

$$\sum_{v \in Values(Humidity)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$-\frac{7}{14}\left(-\frac{3}{7}log\frac{3}{7}-\frac{4}{7}log\frac{4}{7}\right)+\frac{7}{14}\left(-\frac{6}{7}log\frac{6}{7}-\frac{1}{7}log\frac{1}{7}\right)$$

$= 0.789$

Therefore

$$Gain(S, Humidity) - Entropy(S) - Entropy_{Humidity}(S)$$

$$= 0.940 - 0.789$$

$$= 0.151$$

Similarly,

$Entropy_{Wind}(S)$ is given as

$$\sum_{v \in Values(Wind)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= \frac{8}{14}\left(-\frac{6}{8}log\frac{6}{8}-\frac{2}{8}log\frac{2}{8}\right)+\frac{6}{14}\left(-\frac{3}{6}log\frac{3}{6}-\frac{3}{6}log\frac{3}{6}\right)$$

$= 0.892$

Therefore

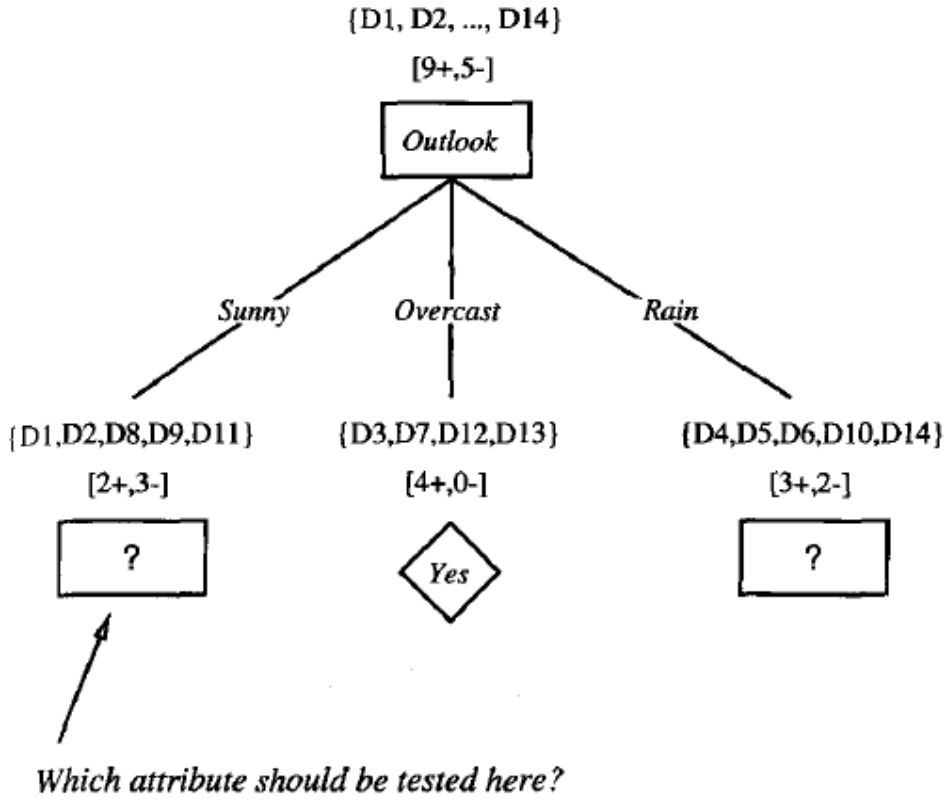
$$Gain(S, Wind) = Entropy(S) - Entropy_{Wind}(S)$$

$$= 0.940 - 0.892$$

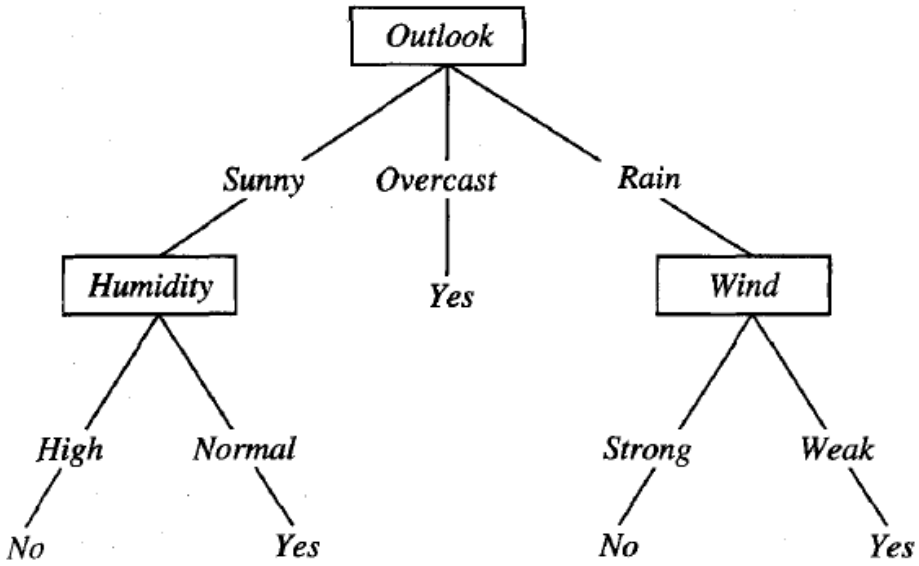
$$= 0.048$$

Since, *Gain(S, Outlook)* is high so, the attribute *Outlook* is tested first and becomes the root for the decision tree. It then classifies S into three

partitions [Let us say S1:leftmost subtree, S2:middle subtree, S3:rightmost subtree] (*because it has three different values: Sunny, Overcast, Rainy*) as shown below.



Now, we need to apply the same procedure to decide the root node for the leftmost subtree and for the rightmost subtree. However, observe that all the samples of middle tree are related to the same class (*Yes class*). So, we can create a leaf node here and label it with ***Yes.*** The final decision tree for the given data set, S, is given below.



After understanding this procedure, now we are ready to design the algorithm for the basic decision tree learning algorithm (ID3). This is given below.

ID3(*(Examples, Targetattribute, Attributes)*

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - *A* ← the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* ← *A*
  - For each possible value, v_i, of *A*,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for *A*
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {*A*}))
- End
- Return *Root*

3.5. HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- ID3 can be characterized as searching a space of hypotheses for one that fits the training examples.

- The hypothesis space searched by ID3 is the set of possible decision trees. ID3 performs a simple-to complex, hill-climbing search through this hypothesis space, beginning with the empty tree, then considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data. The evaluation function that guides this hill-climbing search is the information gain measure.
- This search is depicted in the figure.

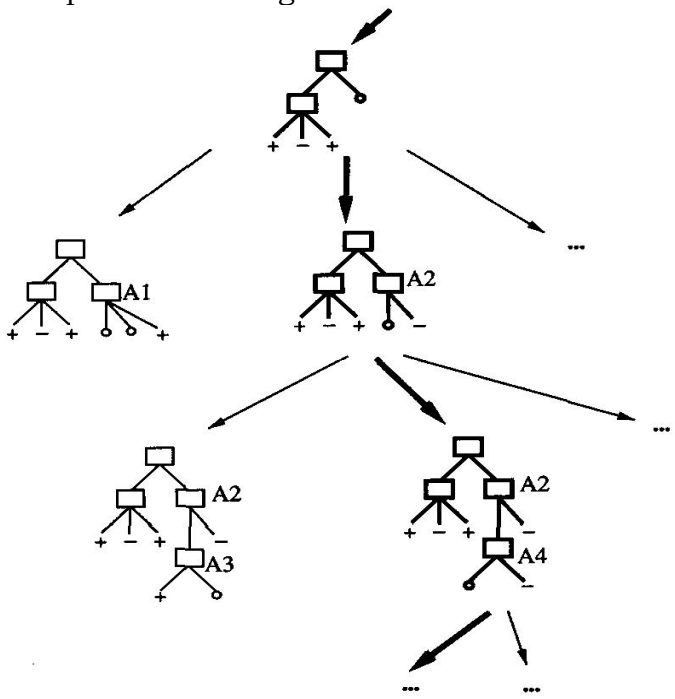


Fig: Hypothesis space search in decision tree learning

- In the above diagram, the search is denoted by dark lines.
- The search strategy has the following capabilities and limitations.
  - ID3's hypothesis space of all decision trees is a ***complete*** space relative to the available attributes.
  - ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
  - ID3 in its pure form performs no backtracking in its search. Once it, selects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice. Therefore, it is susceptible to the usual risks locally optimal solutions that are not globally optimal.
  - ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis.

3.6. INDUCTIVE BIAS IN DECISION TREE LEARNING

- Given a collection of training examples, there are typically many decision trees consistent with these examples.
- Describing the inductive bias of ID3 therefore consists of describing the basis by which it chooses one of these consistent hypotheses over the others. Which of these decision trees does ID3 choose?
- It chooses the first acceptable tree it encounters in its search through the space of possible trees.
- Roughly speaking, then, the ID3 search strategy (a) selects in favour of shorter trees over longer ones, and (b) selects trees that place the attributes with highest information gain closest to the root.
- However, we can approximately characterize its bias as a preference for short decision trees over complex trees.

Approximate inductive bias of ID3: Shorter trees are preferred over larger trees.

- In particular, it does not always find the shortest consistent tree, and it is biased to favor trees that place attributes with high information gain closest to the root.

A closer approximation to the inductive bias of ID3: Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

3.6.1 Restriction Biases and Preference Biases

Restriction Bias – is a bias where there are restrictions on the selection of hypothesis among all possible hypotheses. For example, Candidate-Elimination algorithm has restriction bias.

Preference Bias - is a bias where there are no restrictions on the selection of hypothesis among all possible hypotheses. For example, ID3 has preference bias.

The difference between the hypothesis space search in these two approaches:

ID3 searches a complete hypothesis space (i.e., one capable of expressing any finite discrete-valued function). It searches incompletely through this space, from simple to complex hypotheses, until its termination condition is met (e.g., until it finds a hypothesis consistent with the data). Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy. Its hypothesis space introduces no additional bias.

The version space CANDIDATE-ELIMINATION algorithm searches an incomplete hypothesis space (i.e., one that can express only a subset of the potentially teachable concepts). It searches this space completely, finding every hypothesis consistent with the training data. Its inductive bias is solely a consequence of the expressive power of its hypothesis representation. Its search strategy introduces no additional bias.

3.6.2 Why Prefer Short Hypotheses?

- Is ID3's inductive bias favouring shorter decision trees a sound basis for generalizing beyond the training data? Occam discussed this question and it is called as Occam's razor.

Occam's razor: Prefer the simplest hypothesis that fits the data.

- Then why prefer shorter hypothesis? Because, for the given training data, there are fewer short hypotheses whereas there are a large number of complex hypotheses that fit the training data.

There are many more 500-node decision trees than 5-node decision trees. Given a small set of 20 training examples, we might expect to be able to find many 500-node decision trees consistent with these, whereas we would be more surprised if a 5-node decision tree could perfectly fit this data. We might therefore believe the 5-node tree is less likely to be a statistical coincidence and prefer this hypothesis over the 500-node hypothesis.
A second problem with the above argument for Occam's razor is that the size of a hypothesis is determined by the particular representation used ***internally*** by the learner. Two learners using different internal representations could therefore anive at different hypotheses, both justifying their contradictory conclusions by Occam's razor!.

3.7 ISSUES IN DECISION TREE LEARNING

Practical issues in learning decision trees include determining how deeply to grow the decision tree, handling continuous attributes, choosing an appropriate attribute selection measure, handling training data with missing attribute values, handling attributes with differing costs, and improving computational efficiency. The issues are listed below:

3.7.1 Avoiding Overfitting the Data

The algorithm described in ID3 grows each branch of the tree just deeply enough to perfectly classify the training examples. While this is ometimes a reasonable strategy, in fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that ***overfit*** the training examples.

We will say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

Definition: **Given a hypothesis space H, a hypothesis *h E* H is said to overfit the training data if there exists some alternative hypothesis *h'* E H, such that *h* has smaller error than *h'* over the training examples, but *h'* has a smaller error than *h* over the entire distribution of instances.**

Figure illustrates the impact of overfitting in a typical application of decision tree learning. In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes. The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by the tree.

The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples (not included in the training set). Predictably, the accuracy of the tree over the training examples increases monotonically as the tree is grown. However, the accuracy measured over the independent test examples first increases, then decreases.

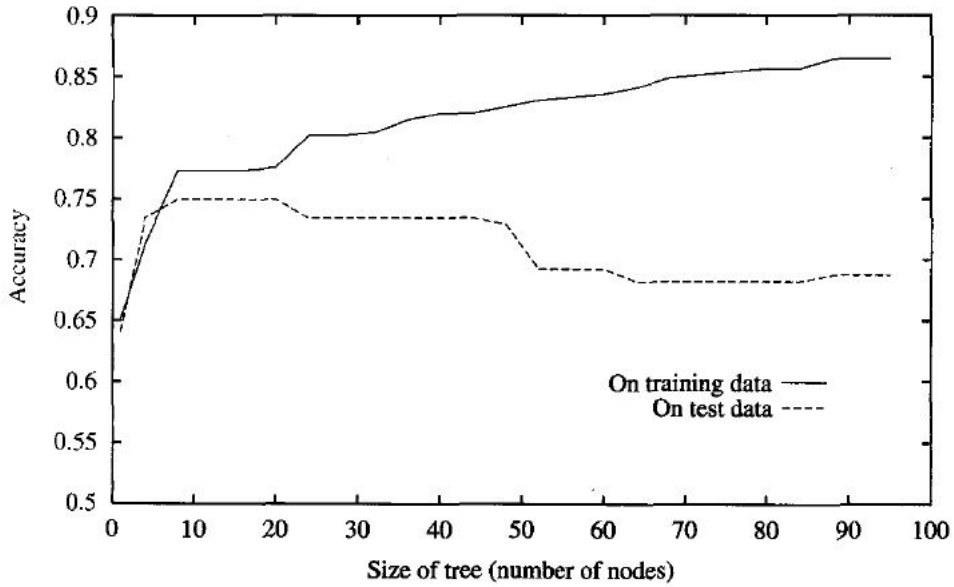


Fig: Overfitting in decision tree learning

There are several approaches to avoiding overfitting in decision tree learning.

These can be grouped into two classes:

- approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
- approaches that allow the tree to overfit the data, and then post-prune the tree.

3.7.1.1 REDUCED ERROR PRUNING

Reduced-error pruning (Quinlan 1987), considers each of the decision nodes in the tree to be candidates for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node.

Nodes are removed only if the resulting pruned tree performs no worse than-the original over the validation set. This has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set.

Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set. Pruning of nodes continues until further pruning is harmful.

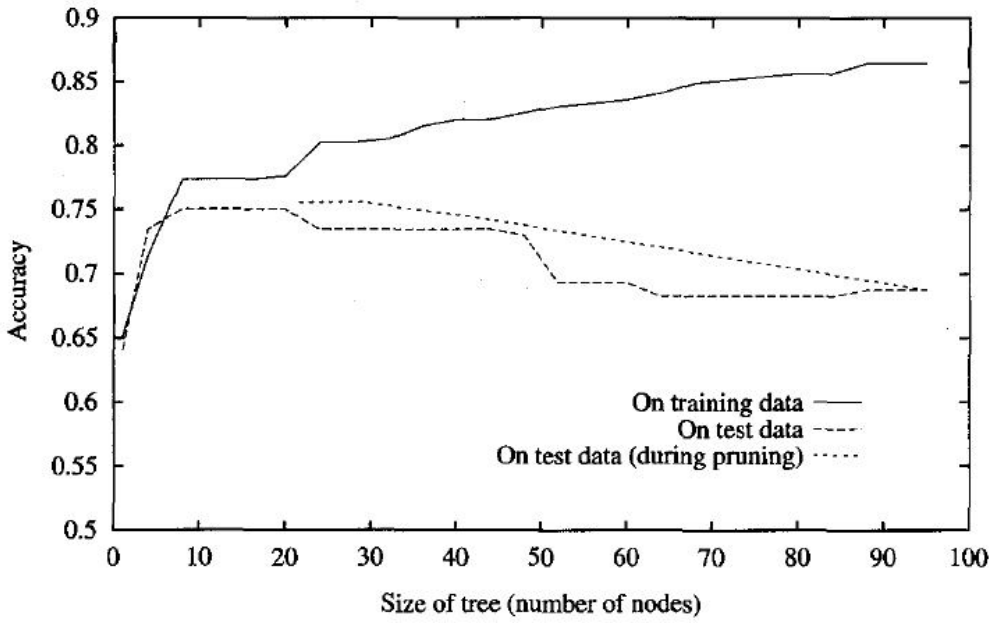


Fig: Effect of reduced-error pruning in decision tree learning

3.7.1.2RULE POST-PRUNING

Rule post-pruning involves the following steps:

a. Infer the decision tree from the training set, growing the tree until the training
data is fit as well as possible and allowing overfitting to occur.

b. Convert the learned tree into an equivalent set of rules by creating one rule
for each path from the root node to a leaf node.

c. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.

d. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Why convert the decision tree to rules before pruning? There are three main advantages.

- Converting to rules allows distinguishing among the different contexts in which a decision node is used. Because each distinct path through the decision tree node produces a distinct rule, the pruning decision regarding that attribute test can be made differently for each path. In contrast, if the tree itself were pruned, the only two choices would be to remove the decision node completely, or to retain it in its original form.
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves. Thus, we avoid messy bookkeeping issues such as how to reorganize the tree if the root node is pruned while retaining part of the subtree below this test.

- Converting to rules improves readability. Rules are often easier for to understand.

3.7.2 Incorporating Continuous-Valued Attributes

Our initial definition of ID3 is restricted to attributes that take on a discrete set of values.

First, the target attribute whose value is predicted by the learned tree must be discrete valued. Second, the attributes tested in the decision nodes of the tree must also be discrete valued. This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree. This can be accomplished by dynamically defining new discretevalued attributes that partition the continuous attribute value into a discrete set of intervals. In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new boolean attribute A, that is true if A < c and false otherwise. The only question is how to select the best value for the threshold c.

As an example, suppose we wish to include the continuous-valued attribute ***Temperature*** in describing the training example days in the learning task of play tennis.

Suppose further that the training examples associated with a particular node in the decision tree have the following values for ***Temperature*** and the target attribute ***PlayTennis.***

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

What threshold-based boolean attribute should be defined based on Temperature?

Clearly, we would like to pick a threshold, ***c,*** that produces the greatest information gain. By sorting the examples according to the continuous attribute **A,** then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of ***A.***

UNIT-III
SECTION-A

Objective Questions

1. Decision Tree is a display of an algorithm. [ ]

A. True B. False

2. Decision Trees can be used for classification tasks. [ ]

A. True B. False

3. When a decision tree is grown to full depth, it is more likely to fit the noise in the data. [ ]

A. True B. False

4. For which of the following hyper parameters, higher value is better for decision tree algorithm? [ ]

1. Number of samples used for split
2. Depth of tree
3. Samples for leaf

A.1 and 2 B. 2 and 3 C. 1 and 3 D. 1, 2 and 3

4. Below are some assumptions that we made while using decision tree. Identify the wrong statement.

[ ]

A. At the beginning, we consider the whole training set as the child node
B. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
C. On the basis of attribute values records are distributed recursively.
D. We use statistical methods for ordering attributes as root or the internal node.

6. In which of the following scenario a gain ratio is preferred over Information Gain? [ ]

A. When a categorical variable has very large number of category
B. When a categorical variable has very small number of category
C. Number of categories is the not the reason
D. None of these

7. Below are the 8 actual values of target variable in the train file:[0,0,0,1,1,1,1,1].What is the entropy of the target variable? [ ]

A. -(5/8 log(5/8) + 3/8 log(3/8)) B. 5/8 log(5/8) + 3/8 log(3/8)

C. 3/8 log(5/8) + 5/8 log(3/8) D. 5/8 log(3/8) – 3/8 log(5/8)

8. Decision Tree is [ ]

A. Flow-Chart

B. Structure in which internal node represents test on an attribute, each branch represents outcome of test and each leaf node represents class label

C. Flow-Chart & Structure in which internal node represents test on an attribute, each branch represents outcome of test and each leaf node represents class label

D. None of the mentioned

9. Suppose S is a collection of 14 examples of some Boolean concept, including 9 positive and 5 negative examples (we adopt the notation [9+, 5-] to summarize such a sample of data). Then the entropy of S relative to this boolean classification is [ ]

A.0.940 B.0.80 C.0.70 D.0.89

10. Suppose *S* is a collection of training-example days described by attributes including *Wind,* which can have the values *Weak* or *Strong.* Also consider, *S* is a collection containing *14* examples, [9+, 5-]. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have *Wind = Weak,* and the remainder have *Wind = Strong. What is the* information gain due to sorting the original *14* examples by the attribute *Wind ?* [ ]

A.0.045 B.0. C.0.038 D.0.035

11. Indentify wrong statement in the issues of decision tree learning [ ]

A. reduced error pruning

B. rule pre-pruning

C. incorporating continuous-valued attributes

D. handling training examples with missing attribute values

SECTION-B

Descriptive Questions

1. Explain basic decision tree algorithm.
2. Explain how hypothesis space search is carried in decision tree learning.
3. Discuss any three issues in decision tree learning.
4. What do you mean by Gain and Entropy? How is it used to build the Decision tree in algorithm? Illustrate using an example
5. What is the procedure of building decision tree using ID3 algorithm with gain and entropy. Illustrate with example.
6. Explain inductive bias in decision tree learning.
7. Explain various attributes selection measures for constructing a decision tree.
8. How is a decision tree pruned?
9. Consider the following set of training example:

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

   a. what is the entropy of this collection of training example with respect to target function classification.

   b. What is information gain of a2 relative to above training example.

10. What is Occam's razor?