16.1.15

UNIT-II

Chapter 3

DECISION TREE LEARNING

Introduction:

- Decision tree learning is one of the most widely used and practical methods for inductive inference.
- It is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.

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.

Ex: A decision tree for the concept PlayTennis (whether or not Saturday mornings are suitable for playing tennis)

yes — means suitable
no — " not suitable

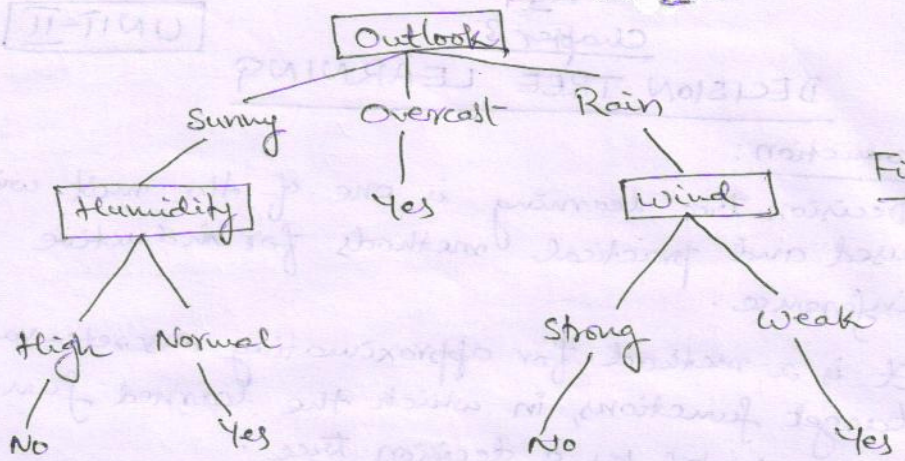


Fig. 1

In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

The decision tree in Fig. 1 corresponds to the expression:

(Outlook = Sunny ∧ Humidity = Normal)
∨ (Outlook = Overcast)
∨ (Outlook = Rain ∧ Wind = Weak)

Appropriate Problems for Decision Tree Learning

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

(i) Instances are represented by attribute-value pairs.
(Ex: Attribute 'Temperature' taking values Hot, Mild and cool.)

(ii) The target function has discrete output values.
(the above decision tree has boolean classification: yes or no)

(iii) Disjunctive descriptions may be required.
(iv) The training data may contain errors.
(v) The training data may contain missing attribute values.

Classification Problems:

Problems in which the task is to classify examples into one of a discrete set of possible categories, are often referred to as "classification problems".

The Basic Decision Tree Learning Algorithm

ID3 is a core algorithm that employs a top-down, greedy search through the space of possible decision trees. C4.5 is its successor.

ID3 learns decision trees by constructing them top-down. The best attribute is selected and used as the test at the root node of the tree. A descendant of the root node is then created for each possible value of its attribute.

The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.

This forms a greedy search in which the algorithm never backtracks to reconsider earlier choices.

A simplified version of the algorithm, specialized to learning boolean-valued functions (ie, concept learning) is described below.

ID3 Algorithm:

ID3(Examples, Target_attribute, Attributes)

Examples are the training examples.

Target_attribute is the attribute whose value is to be predicted by the tree.

Attributes is a list of other attributes that may be tested by the learned decision tree.

Returns a decision tree that correctly classifies the given Examples.

- 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.

The best attribute (as decided by the ID3 algorithm) is the one with highest information gain.

Information Gain:

It measures how well a given attribute separates the training examples according to their target classification. Based upon this, ID3 selects candidate (best) attributes at each step while growing the tree.

Entropy:

It is a measure that characterizes the (im)purity of an arbitrary collection of examples. (It measures homogeneity of examples.)

Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

$$\text{Entropy}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \quad \rightarrow (1)$$

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

$p_{\ominus}$ is the proportion of negative examples in S.

Consider S as a collection of 14 examples of some boolean concept, as shown below.

Table 1

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

S has 9 positive and 5 negative examples.
Denote this by [9+, 5-].

Then the entropy of S is:

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

⟶ (2)

17.1.15

Note:

(i) Entropy is 0 if all members belong to the same class (yes or no).

If all are positive, then $p_{\oplus} = \frac{14}{14} = 1$

and $p_{\ominus} = \frac{0}{14} = 0$.

$$\therefore \text{Entropy}(S) = -\left(\frac{14}{14}\right)\log_2\left(\frac{14}{14}\right) - \left(\frac{0}{14}\right)\log_2\left(\frac{0}{14}\right)$$

$$= -1.0 - 0.0 = \underline{0}$$

(ii) Entropy is 1 when S contains an equal no. of positive and negative examples.

ie, $$\text{Entropy}(S) = -\left(\frac{7}{14}\right)\log_2\left(\frac{7}{14}\right) - \left(\frac{7}{14}\right)\log_2\left(\frac{7}{14}\right)$$

$$= \log_2\left(\frac{7}{14}\right)\left[-\tfrac{1}{2} - \tfrac{1}{2}\right]$$

$$= \log_2 2^{-1}\,[-1] = -1.-1 = \underline{1}$$

(iii) If the collection contains unequal no. of positive and negative examples, then entropy is between 0 and 1.

Fig. 2 shows the form of the entropy function relative to a boolean classification, as $p_{\oplus}$ varies between 0 and 1.

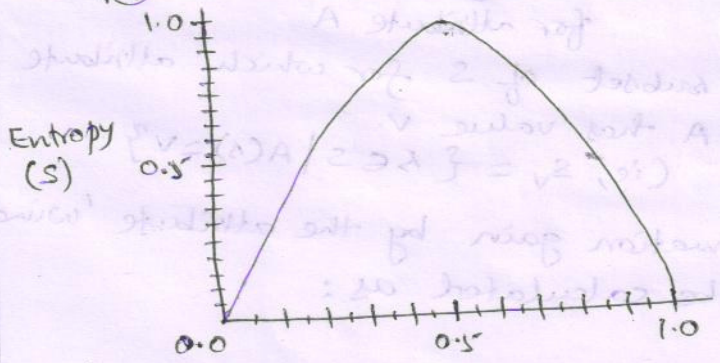


Fig. 2

Entropy is discussed above as a special case where the target classification is boolean.

ie, If the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as,

$$\text{Entropy}(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i \longrightarrow (3)$$

Information Gain:

- It is a measure of the effectiveness of an attribute in classifying the training data.
- It is the expected reduction in entropy caused by partitioning the examples according to this attribute.

The information gain, Gain(S,A) of an attribute A relative to a collection of examples S, is defined as,

$$\text{Gain}(S,A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \longrightarrow (4)$$

where

Values(A) – set of all possible values for attribute A

S_v – subset of S for which attribute A has value v.

(ie, $S_v = \{ s \in S \mid A(s) = v \}$.

Ex: Information gain by the attribute 'wind' can be calculated as:

Values (wind) = weak, strong (from Table 1)

$S = [9+, 5-]$

$S_{weak} \leftarrow [6+, 2-]$

$S_{strong} \leftarrow [3+, 3-]$

$$Gain(S, wind) = Entropy(S) - \sum_{v \in \{weak, strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

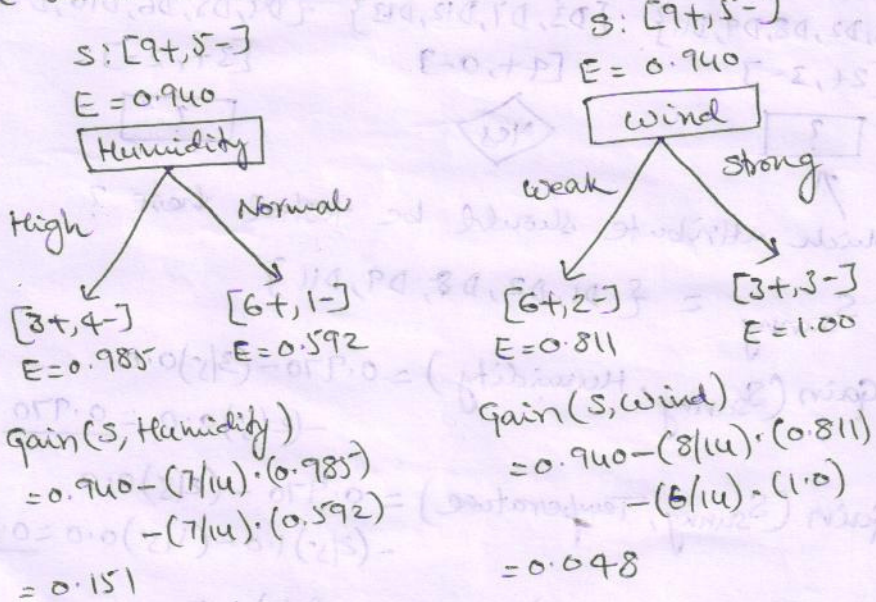
$$= Entropy(S) - (8/14)\, Entropy(S_{weak}) - (6/14)\, Entropy(S_{strong})$$

$$= 0.940 - (8/14)\,0.811 - (6/14)\,1.00$$

$$= 0.048$$

(Here,
$Entropy(S_{weak}) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8)$
$Entropy(S_{strong}) = 1$ (∵ positive and negative members are equal))

Information gain by 'Humidity' and 'Wind' are shown below:



Gain(S, Humidity)
$= 0.940 - (7/14)\cdot(0.985) - (7/14)\cdot(0.592)$
$= 0.151$

Gain(S, wind)
$= 0.940 - (8/14)\cdot(0.811) - (6/14)\cdot(1.0)$
$= 0.048$

Fig. 3

The information gain values for all four attributes are :

Gain (S, Outlook) = 0.246
Gain (S, Humidity) = 0.151
Gain (S, Wind) = 0.048
Gain (S, Temperature) = 0.029

From these, it is understood that 'Outlook' can be selected as the decision attribute for the root node.

The resulting partial decision tree is shown in Fig. 4.

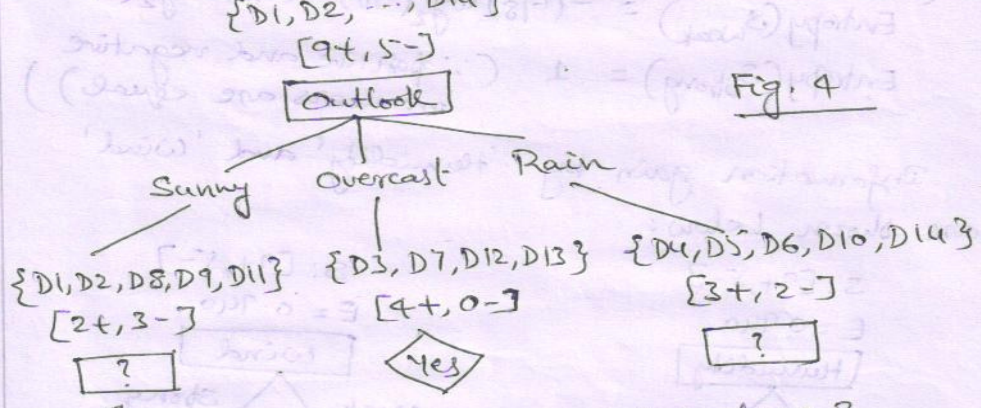


Fig. 4

Which attribute should be tested here ?

S_{sunny} = {D1, D2, D8, D9, D11}

Gain (S_{sunny}, Humidity) = 0.970 − (3/5) 0.0 − (2/5) 0.0 = 0.970 ✓

Gain (S_{sunny}, Temperature) = 0.970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = 0.570

Gain (S_{sunny}, Wind) = 0.970 − (2/5) 1.0 − (3/5) 0.918 = 0.019

In the above figure,

- Every example for which Outlook = Overcast is also a positive example of PlayTennis. Therefore this node of the tree becomes a leaf node with the classification PlayTennis = Yes.
- The descendants corresponding to Outlook = Sunny and Outlook = Rain still have nonzero entropy, and the decision tree will be further elaborated below these nodes.

The process of selecting a new attribute is now repeated for each nonterminal descendant node.

Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.

This process continues for each new leaf node until either of two conditions is met:-

(1) every attribute has already been included along this path through the tree, or

(2) the training examples associated with this leaf node all have the same target attribute value (ie, their entropy is zero).

- Fig. 4 illustrates the computations of information gain for the next step.
- The final decision tree learned by ID3 is shown in Fig. 1.

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 is the set of possible decision trees.
- ID3 performs a simple-to-complex, hill-climbing search, 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 is the information gain measure.

18.1.15

Capabilities and Limitations of ID3 algorithm:

(1) ID3's hypothesis space of all decision trees is a complete space of finite discrete-valued functions, relative to the available attributes.

(Some other methods search incomplete hypothesis spaces — considering only conjunctive hypotheses.)

(2) ID3 maintains only a single current-hypothesis as it searches through the space of decision trees. Because of this, ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses.

(Candidate-Elimination method maintains the set of all hypotheses consistent with the available training examples.)

(3) ID3 in its pure form performs no backtracking in its search.

Therefore, there is a risk of converging to locally optimal solutions that are not globally optimal. Locally optimal solution corresponds to the decision tree it selects along the single search path it explores.

(Another method 'post-pruning the decision tree' allows backtracking.)

(4) ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis.

One advantage of using measures like information gain is that the resulting search is much less sensitive to errors in individual training examples.

(FIND-S or Candidate-Elimination methods make decisions incrementally, based on individual training examples.)

Inductive Bias in Decision Tree Learning

Inductive Bias of ID3 means — the policy by which ID3 generalizes from observed training examples to classify unseen instances.

- Given a collection of training examples, there are many decision trees consistent with these examples.
- Describing the inductive bias of ID3 consists of describing the basis by which it chooses one of these consistent hypotheses over the others.
- 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.

We can approximately characterize the bias of ID3.

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

Because ID3 uses the information gain heuristic and a hill climbing strategy, it is biased to favour 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.

20.1.15

1. Restriction Biases and Preference Biases

Differences between the hypothesis space search in ID3 and Candidate-Elimination approaches:

ID3:

- It searches a complete hypothesis space (ie, 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 (ex: 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.

Candidate-Elimination Algorithm:

- It searches an incomplete hypothesis space (ie, 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.

Reference Bias:

The inductive bias of ID3 is a preference for certain hypotheses over others (ex: for shorter hypotheses, with no hard restriction on the hypotheses

that can be eventually enumerated.)

This form of bias is called a preference bias (or a search bias).

Restriction Bias:

The bias of Candidate-Elimination algorithm is in the form of a categorical restriction on the set of hypotheses considered.

This form of bias is called a restriction bias (or a language bias).

- A preference bias is more desirable than a restriction bias, because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function. (ID3 exhibits purely preference bias.)
- A restriction bias that strictly limits the set of potential hypotheses is generally less desirable, because it introduces the possibility of excluding the unknown target function altogether. (Candidate-Elimination algorithm exhibits purely restriction bias.)

21.1.15

2. Why prefer Short Hypotheses?
(Occam's Razor)

William of Occam (or Ockham) suggested the following and was given the name 'Occam's razor'.

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

(William was born in Occam — a small village in England. His period : 1287–1347.)

- Scientists sometimes appear to follow this inductive bias — they prefer simple explanations (ex: for the motions of the planets) over more complex explanations.

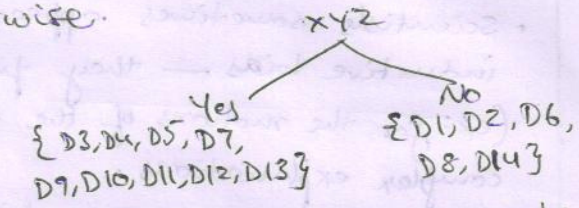
Consider decision tree hypotheses.
Suppose we have 20 training examples.
There are many 500-node decision trees consistent with these.
If a 5-node decision tree that could perfectly fit this data, then we prefer this hypothesis over the 500-node hypothesis.
(The 5-node decision tree is less likely to be a statistical coincidence.)

22.1.15

A problem with Occam's Razor:

The problem 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 arrive at different hypotheses, both justifying their contradictory conclusions by Occam's razor.
For ex. the function represented by the learned decision tree in Fig. 1 could be represented [redacted] as a tree with just one decision node, by a learner that uses the boolean attribute XYZ, where we define the attribute XYZ

to be true for instances that are classified positive by the decision tree (in Fig. 1) and false otherwise.



Thus, two learners, both applying Occam's razor, would generalize in different ways if one used the XYZ attribute to describe its examples, and the other used only the attributes Outlook, Temperature, Humidity and Wind.

Issues in Decision Tree learning

These are:

- How deeply to grow the 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.

These issues are discussed below w.r.t. the extensions to the basic ID3 algorithm.

1. Avoiding Overfitting the Data

The ID3 algorithm grows each branch of the tree just deeply enough to perfectly

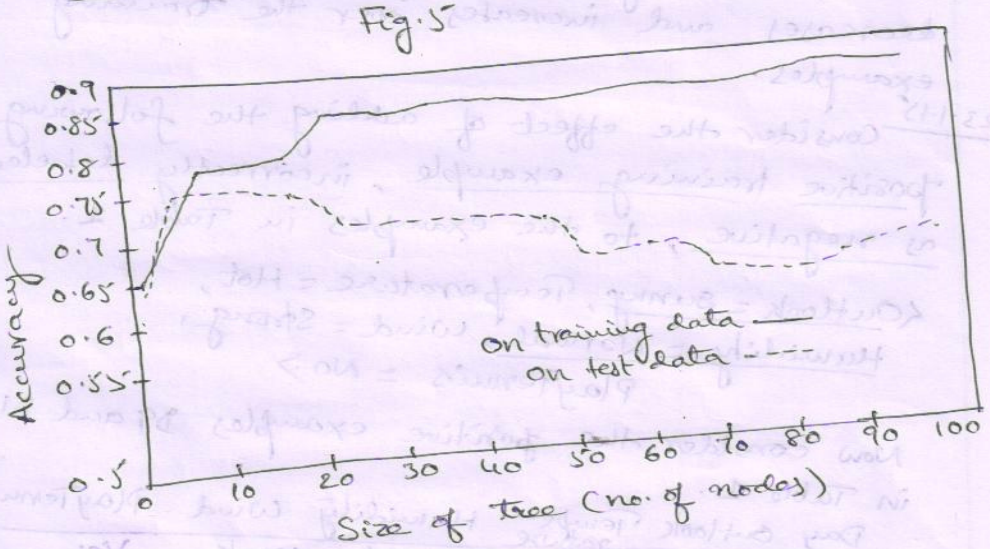
classify the training examples.

Sometimes it can lead to difficulties when there is no noise in the data, or when the no. 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.

Definition:

Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, 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.

Fig. 5



- Fig. 5 illustrates the impact of overfitting.
- Here, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes.
- X-axis indicates the total no. of nodes in the decision tree, as the tree is being constructed.
- Y-axis indicates the accuracy of predictions made by the tree.
- Solid line shows the accuracy of the tree over the training examples.
- Broken line shows accuracy over an independent set of test examples (not included in the training set).

Once the size exceeds approximately 25 nodes, accuracy over the test examples decreases and increases over the training examples.

23.1.15 Consider the effect of adding the following positive training example, incorrectly labeled as negative, to the examples in Table 1.

<Outlook = Sunny, Temperature = Hot, Humidity = Normal, Wind = Strong, PlayTennis = No>

Now consider the positive examples D9 and D11 in Table 1.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

The new example will be sorted into the second leaf node from the left in the learned tree of Fig. 1, along with the positive examples D9 and D11.

In D9 and D11, we have Outlook = Sunny
and Humidity = Normal

In the new example also, we have the same values; hence it must be a positive example. But it is labeled as a negative example — hence we can say that the set of training examples now contain random error (or noise).

- Given the original error-free data, ID3 produces the decision tree in Fig. 1.
- But the addition of this incorrect example will now cause ID3 to construct a more complex tree.
- ID3 will succeed in finding a new decision attribute to separate out this new example from D9 and D11 at this tree node.
- The result is that ID3 will output a decision tree (h) that is more complex than the original tree from Fig. 1 (h').
- h will fit the collection of training examples perfectly, whereas the simpler h' will not.
- Given that new decision node is simply a consequence of fitting the noisy training example, we expect h to outperform h' over subsequent data drawn from the same instance distribution.

The above example illustrates how random noise in the training examples can lead to overfitting.

Approaches to avoiding overfitting in decision tree learning:

These can be grouped into two classes:

(i) approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data.

(ii) approaches that allow the tree to overfit the data, and then post-prune the tree.

26.1.15 A key question is what criterion is to be used to determine the correct final tree size (regardless of whether the correct size is found by stopping early or by post-pruning).

Approaches include:

(a) Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.

(b) Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. (Ex: Using a chi-square test)

(c) Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. (This approach is based on a heuristic called the Minimum Description Length Principle.)

Approach (a) is referred to as "training and validation set approach". Here, the available data are separated into two sets of examples.

training set — used to form the learned hypothesis

validation set — used to evaluate the accuracy of this hypothesis over subsequent data, and to evaluate the impact of pruning this hypothesis.

Validation set can be expected to provide a safety check against overfitting the spurious characteristics of the training set.

(Usually we withhold one-third of the available examples for the validation set, using the other two-thirds for training.)

The two main variants of approach (a) are discussed below.

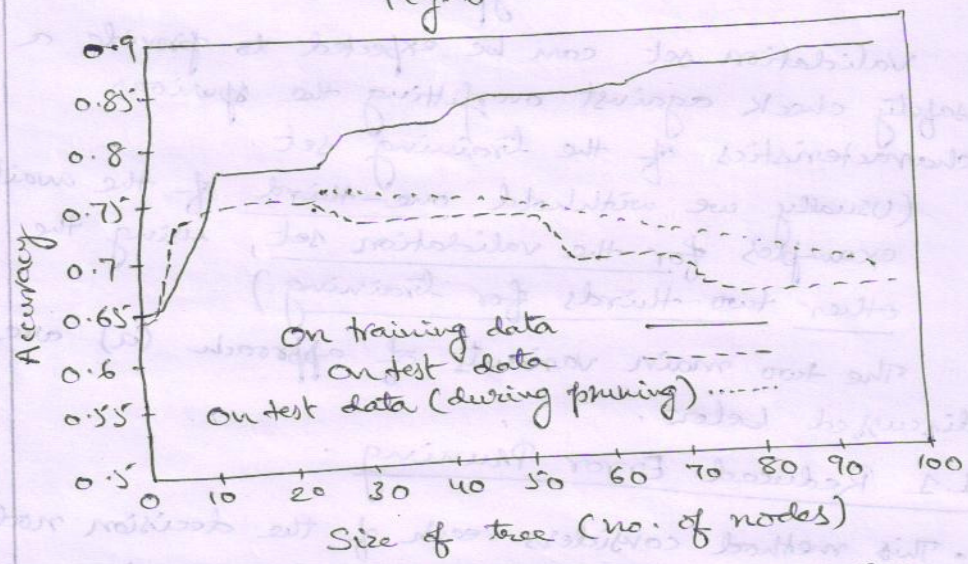
1.1 Reduced Error Pruning:

- This method 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.

- 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 (ie, decreases accuracy of the tree over the validation set).

The impact of reduced-error pruning on the accuracy of the tree is shown in Fig. 6.

Fig. 6



- The additional line (.....) shows accuracy over the test examples as the tree is pruned. When pruning begins, the tree is at its maximum size and lowest accuracy. As pruning proceeds, the no. of nodes is reduced and accuracy over the test set increases.
- Here, the available data has been split into three subsets:

- the training examples
- the validation examples used for pruning the tree, and
- the test examples used to provide an unbiased estimate of accuracy over future unseen examples.

- The plot shows accuracy over the training and test sets. Accuracy over the validation set is not shown.

Drawback:

When data is limited, withholding part of it for the validation set reduces the no. of examples available for training.

(Section 1.2 presents an alternative approach to pruning — found useful — where data is limited.)

1.2 Rule Post-Pruning:

It involves the following steps:

(i) 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.

(ii) 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.

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

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

In rule post-pruning, one rule is generated for each leaf node in the tree.

From Fig. 1,

IF (Outlook = Sunny) ∧ (Humidity = High)
THEN PlayTennis = No

(Outlook = Sunny) ∧ (Humidity = High) → rule antecedent (precondition)

PlayTennis = No → rule consequent (postcondition)

A rule is removed (pruned) by removing any antecedent, whose removal does not worsen its estimated accuracy.

For the above example, rule post-pruning will remove the preconditions (Outlook = Sunny) and (Humidity = High).

No pruning step is performed if it reduces the estimated rule accuracy.

Necessity of converting the decision tree to rules before pruning:

(i) It allows distinguishing among the different contexts in which a decision node is used.

(ii) It removes the distinction between attribute tests that occur near the root and those that occur near the leaves of the tree.

(iii) It improves readability. Rules are often easier for people to understand.

27-1-15

2. Incorporating Continuous-Valued Attributes

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

(i) target attribute (ex. PlayTennis) — must be discrete-valued

(ii) attributes tested in the decision node — must also be discrete-valued

Restriction (ii) can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree.

Consider the continuous-valued attribute Temperature as shown below:

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

If attribute A is continuous-valued, the algorithm can dynamically create a new boolean attribute A_c that is true if $A < c$ and false otherwise.

Clearly, we would like to pick a threshold c, that produces the greatest information gain.

For this, by identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of A.

For the above example, there are two candidate thresholds.

(i) $(48+60)/2 \rightarrow 54$

(ii) $(80+90)/2 \rightarrow 85$

The thresholds are: Temperature > 54 and Temperature > 85.

Information gain can be computed for each of these candidate attributes, and the best can be selected (Temperature > 54).

This dynamically created boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree.

3. Alternative Measures for Selecting Attributes

There is a natural bias in the information gain measure that favours attributes with many values over those with few values.

Consider attribute Date (ex. March 4, 1979).

It has a very large no. of possible values. If this is added to Table 1, it will perfectly predict the target attribute over the training data.

It would be selected as the root node of the tree and lead to a quite broad tree of depth one, which perfectly classifies the training data.

It will have a very high information gain but it will be a poor predictor of the target function over unseen instances.

One way to avoid this difficulty is to select decision attributes based on some measure other than information gain. That is Gain Ratio.

Gain Ratio uses Split information.

$$\text{SplitInformation}(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_1 through S_c are the c subsets of examples resulting from partitioning S by the c-valued attribute.

$$\text{GainRatio}(S,A) \equiv \frac{\text{Gain}(S,A)}{\text{SplitInformation}(S,A)}$$

(Gain(S,A) is the information gain of A over S).

- SplitInformation term discourages the selection of attributes with many uniformly distributed values.

Case 1 : Consider a collection of n examples that are completely separated by attribute A (eg. Date).

In this case, the SplitInformation $(S,A) = \log_2 n$.

Ex: Suppose $n = 4$.

Then SplitInformation (S, Date) $=$ Date: 2.11.74 | 2.4.76 | 1.10.70 | 4.8.68 (1 | 1 | 1 | 1)

$$= -\frac{|S_1|}{|S|}\log_2\frac{|S_1|}{|S|} - \frac{|S_2|}{|S|}\log_2\frac{|S_2|}{|S|} - \frac{|S_3|}{|S|}\log_2\frac{|S_3|}{|S|} - \frac{|S_4|}{|S|}\log_2\frac{|S_4|}{|S|}$$

$$= -\frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4}$$

$$= -1.\log_2\frac{1}{4}$$

$$= -1.-2 \quad (\because \log_2\tfrac{1}{4} = \log_2 0.25 = \log_{10}0.25/\log_{10}2 = -0.60206/0.30103 = -2)$$

$$= 2$$

The result 2 is equal to $\log_2 4$.

Case 2: A boolean attribute B that splits the same n examples exactly in half will have SplitInformation of 1.

Ex: Take $n = 4$.

Then SplitInformation (S, A)

$$= -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}$$

$$= -\frac{1}{2}\left[\log_2\frac{1}{2} + \log_2\frac{1}{2}\right]$$

$$= -\frac{1}{2}[-1 - 1] = \underline{1}$$

- If an attribute is having the same value for nearly all members of S, then denominator (in GainRatio) can be zero or very small.

Ex: $n = 4$ Attribute is having same value for all 4 examples.

Then SplitInformation (S, A)

$$= -\frac{4}{4}\log_2\frac{4}{4} = -1 \cdot \log_2 1$$

$$= -1 \cdot 0 = \underline{0}.$$

So to avoid this, first calculate the Gain of each attribute, then apply the GainRatio test only considering those attributes with above average Gain.

4. Handling Training Examples with Missing Attribute values

Ex: A medical domain

We wish to predict patient outcome based on various laboratory tests.

The lab test 'Blood-Test-Result' is available only for a subset of the patients.

Then it is common to estimate the missing value based on other examples for which this attribute has a known value.

To deal with missing attribute value,

(i) assign it the value that is most common among training examples at node n.

Ex: For 'wind': 8 values are 'strong'.
5 " 'weak'.
1 value is missing.

So assign 'strong' in place of missing value.

(ii) Assign it the most common value among examples at node n that have the classification c(x).

Ex: 'weak' has classification 'No' for 8 examples.
'strong' " 'Yes' for 5 " .

So, assign 'weak' in place of missing attribute value.

5. Handling attributes with Differing Costs

In some learning tasks, the instance attributes may have associated costs.

Ex: classifying medical diseases.

Attributes: Temperature, BiopsyResult, Pulse, BloodTestResult, etc.

They vary significantly in their cost (in terms of monetary cost and cost to patient comfort).

So, we would prefer decision trees that use low-cost attributes where possible, relying on

high-cost attributes only when needed to produce reliable classification.

ID3 can be modified to take into account attribute costs by introducing a cost term into the attribute selection measure.

Ex: Divide the Gain by the cost of the attribute, so that lower-cost attributes would be preferred.

K. Srinivasa Rao
Associate Professor
Dept. of CSE, GIT,
GITAM University.

27-1-2015.

28.1.15

Chapter 6
BAYESIAN LEARNING

1. Introduction

Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems.

Bayesian methods are important for the study of machine learning because they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

Features of Bayesian Learning methods:

(1) Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. (This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.)

(2) Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.

(Prior knowledge is provided by asserting (i) a prior probability for each candidate hypothesis, and (ii) a probability distribution

over observed data for each possible hypothesis.)

(3) Bayesian methods can accommodate hypotheses that make probabilistic predictions. (ex: hypotheses such as "this pneumonia patient has a 93% chance of complete recovery".)

(4) New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

(5) Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

Practical difficulties:

(1) Bayesian methods require initial knowledge of many probabilities. When these are not known in advance, they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.

(2) Significant computational cost is required to determine the optimal hypothesis in the general case.

2. Bayes Theorem

'Best hypothesis' means we demand the most probable hypothesis, given the training data D plus any initial knowledge about the prior probabilities of various hypotheses in H.

Bayes theorem provides a direct method for calculating such probabilities.

It provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

Some Notations:

- $P(h)$ — initial probability that hypothesis h holds, before we have observed the training data (called 'Prior probability of h')
- $P(D)$ — prior probability that training data D will be observed (ie, the probability of D given no knowledge about which hypothesis holds.)
- $P(D|h)$ — probability of observing data D given hypothesis h holds.
  ($P(x|y)$ denotes probability of x given y)
- $P(h|D)$ — probability of h given the observed training data D (called posterior probability of h) (reflects our confidence that h holds after we have seen the training data D)

Note: $P(h)$ is independent of D and $P(h|D)$ reflects the influence of the training data D.

Bayes theorem calculates the posterior probability as follows:

Bayes Theorem:

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)} \longrightarrow (1)$$

- $P(h|D)$ increases with $P(h)$ and with $P(D|h)$
- $P(h|D)$ decreases as $P(D)$ increases because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

MAP hypothesis:

The learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis $h \in H$ given the observed data D.

Any such maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis.

We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.

h_{MAP} is a MAP hypothesis provided

$$h_{MAP} \equiv \underset{h \in H}{\operatorname{argmax}}\ P(h|D)$$

$$= \underset{h \in H}{\operatorname{argmax}}\ \frac{P(D|h)\,P(h)}{P(D)}$$

$$= \underset{h \in H}{\operatorname{argmax}}\ P(D|h)\,P(h) \longrightarrow (2)$$

In the final step, the term $P(D)$ is dropped because it is a constant independent of h.

In some cases, we will assume that every hypothesis in H is equally probable a priori ($P(h_i) = P(h_j)$ for all h_i and h_j in H).

Hence in eqn. (2), we only consider the term $P(D|h)$ to find the most probable hypothesis.

$P(D|h)$ — is called the likelihood of the data D given h

h_{ML} — any hypothesis that maximizes $P(D|h)$ is called a maximum likelihood (ML) hypothesis, h_{ML}.

$$h_{ML} \equiv \underset{h \in H}{\operatorname{argmax}}\ P(D|h) \longrightarrow (3)$$

Example:

Consider a medical diagnosis problem in which there are two alternative hypotheses that:

(1) the patient has a particular form of cancer, and

(2) the patient does not.

- The available data is from a particular lab test with two possible outcomes: ⊕ (positive) and ⊖ negative.
- Prior knowledge: Over the entire population of people, only 0.008 have this disease.
- Lab test is only an imperfect indicator of the disease.
- The test returns:
  a correct positive result in only 98% of the cases in which the disease is actually present, and
  a correct negative result in only 97% of the cases in which the disease is not present.
  - In other cases, the test returns the opposite result.

The above situation can be summarized by the following probabilities:

$P(\text{cancer}) = 0.008$ $\quad P(\neg\text{cancer}) = 0.992$

$P(\oplus|\text{cancer}) = 0.98$ $\quad P(\ominus|\text{cancer}) = 0.02$

$P(\oplus|\neg\text{cancer}) = 0.03$ $\quad P(\ominus|\neg\text{cancer}) = 0.97$

Suppose now we observe a new patient for whom the lab test returns a positive result.

The MAP hypothesis can be found using eqn. (2).

$P(\oplus|\text{cancer})\,P(\text{cancer}) = (0.98)(0.008) = 0.0078$

$P(\oplus|\neg\text{cancer})\,P(\neg\text{cancer}) = (0.03)(0.992) = 0.0298$ ✓

Thus, $h_{MAP} = \neg\text{cancer}$.

The above two probability values are obtained from the two hypotheses:
h is 'cancer' and h is '¬cancer'.
In the above probability values, the maximum is 0.0298, which belongs to the hypothesis '¬cancer'.
Hence we can say that 'the patient does not have cancer'.

3. Bayes Theorem and Concept Learning

Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data.
So, we can use it as the basis for a straight-forward learning algorithm that can output the most probable hypothesis.
Here we consider a Brute-force Bayesian concept learning algorithm.

3.1 Brute-Force Bayes Concept Learning

Consider the concept learning problem 'EnjoySport' (in chapter 2).
we assume that the learner is given some sequence of training examples $\langle\langle x_1, d_1\rangle, \ldots, \langle x_m, d_m\rangle\rangle$ where x_i is some instance from X and d_i is the target value of x_i (ie, $d_i = c(x_i)$).

To simplify we assume the sequence of instances $\langle x_1, \ldots, x_m\rangle$ is held fixed, so that the training data D

can be written simply as the sequence of the target values $D = \langle d_1, \ldots d_m \rangle$.

- We can design a straightforward concept-learning algorithm to output the MAP hypothesis, based on Bayes theorem, as follows:

30.1.15

Brute-Force MAP Learning Algorithm

1. For each hypothesis h in H, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)}$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}}\ P(h|D).$$

Working of the algorithm:

Let us choose $P(h)$ and $P(D|h)$ to be consistent with the following assumptions:

(1) The training data D is noise free (ie, $d_i = c(x_i)$).

(2) The target concept is contained in the hypothesis space H.

(3) We have no a priori reason to believe that any hypothesis is more probable than any other.

Values for $P(h)$:

From (3), it is reasonable to assign the same prior probability to every hypothesis h in H.

And from (2), we should require that these probabilities sum to 1.

These constraints imply that

$$P(h) = \frac{1}{|H|} \text{ for all } h \text{ in } H$$ ($|H|$ is no. of hypotheses)

Values for $P(D|h)$:

$P(D|h)$ is the probability of observing the target values $D = \langle d_1 \cdots d_m \rangle$ for the fixed set of instances $\langle x_1 \cdots x_m \rangle$, given a world in which hypothesis h holds (ie, given a world in which h is the correct description of the target concept c).

From the assumption (1),

the probability of observing classification d_i given h is just 1 if $d_i = h(x_i)$ and

0 if $d_i \neq h(x_i)$.

$$\therefore P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases} \longrightarrow (4)$$

(ie, $P(D|h)$ is 1 if D is consistent with h, and 0 otherwise)

From Bayes theorem, we have

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)}$$

To calculate $P(h|D)$, consider the following two cases:

Case 1 : h is inconsistent with D.

$$\therefore P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0 \quad (\because P(D|h) = 0 \text{ from } (4))$$

Case 2: h is consistent with D.

$$\therefore P(h|D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} \quad (\because P(D|h) = 1 \text{ from } ④, \text{ and } P(h) = \frac{1}{|H|} \text{ (all hypotheses have same prior probabilities)})$$

$$= \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}}$$

$$= \frac{1}{|VS_{H,D}|}$$

where $VS_{H,D}$ is the subset of hypotheses from H that are consistent with D.

In the above, we have written $P(D) = \frac{|VS_{H,D}|}{|H|}$

Verification of the value of $P(D)$:

Consider the theorem of total probability:

"If events $A_1, \cdots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

$$P(B) = \sum_{i=1}^{n} P(B|A_i) P(A_i)",$$

and the fact that the hypotheses are mutually exclusive (ie, $(\forall i \neq j)(P(h_i \wedge h_j) = 0)$)

From these, we can derive $P(D)$ as:

$$P(D) = \sum_{h_i \in H} P(D|h_i) P(h_i)$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} = \frac{|VS_{H,D}|}{|H|}$$

(Hence it is verified)

So, Bayes theorem computes the posterior probability $P(h|D)$ as:

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases} \longrightarrow (5)$$

∴ Every consistent hypothesis is a MAP hypothesis.

31.1.15

3.2 MAP Hypotheses and Consistent Learners

Every hypothesis consistent with D is a MAP hypothesis (from the above analysis). From this, we can describe a general class of learners called "Consistent Learners".

A learning algorithm is a consistent learner provided it outputs a hypothesis that commits zero errors over the training examples.

✓ Every consistent learner outputs a MAP hypothesis, if we assume a uniform prior probability distribution over H (ie, $P(h_i) = P(h_j)$ for all i, j), and if we assume deterministic, noise-free training data (ie, $P(D|h) = 1$ if D and h are consistent, and 0 otherwise).

Ex: FIND-S

FIND-S outputs a consistent hypothesis — that is, it will output a MAP hypothesis under the probability distributions $P(h)$ and $P(D|h)$.

FIND-S does not explicitly manipulate probabilities at all — it simply outputs a maximally specific member of the version space.

By identifying distributions for P(h) and P(D|h), we have a useful way of characterizing the behaviour of FIND-S.

Inductive Bias:

It was defined (in chap. 2) as the set of assumptions B sufficient to deductively justify the inductive inference performed by the learner.

Here, instead of modeling the inductive inference method by an equivalent deductive system, we model it by an equivalent probabilistic reasoning system based on Bayes theorem.

The assumptions of the learner are of the form:

- the prior probabilities over H are given by the distribution P(h), and
- the strength of data in rejecting or accepting a hypothesis is given by P(D|h).

K. SRINIVASA RAO
Associate Professor
Dept. of CSE, GIT
GITAM University

31-1-2015