|| Jai Sri Gurudev |

Sri AdichunchanagiriShikshana Trust (R)

SJB INSTITUTE OF TECHNOLOGY

Study Material

Subject Name: Machine Learning

Subject Code: BCS602

By

Faculty Name: Mrs. Pavithra

Designation: Assistant Professor

Semester: VI

Department of Information Science & Engineering

Aca. Year: Even Sem /2024-25

MODULE 3

k- NEAREST NEIGHBORLEARNING

- The most basic instance-based method is the K- Nearest Neighbor Learning. This algorithm assumes all instances correspond to points in the n-dimensional spaceR^n.
- The nearest neighbors of an instance are defined in terms of the standard Euclidean distance.
- Let an arbitrary instance x be described by the featurevector

  $$(a_1(x), a_2(x), \ldots\ldots\ldots, a_n(x))$$

  Where, $a_r(x)$ denotes the value of the rth attribute of instance x.

- Then the distance between two instances x_i and x_j is defined to be $d(x_i , x_j)$
  Where,

  $$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2}$$

- In nearest-neighbor learning the target function may be either discrete-valued or real-valued.

Let us first consider learning ***discrete-valued target functions*** of the form

$$f : \Re^n \rightarrow V.$$

Where, V is the finite set $\{v_1, \ldots v_s\}$

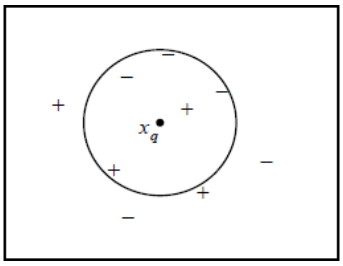
The k- Nearest Neighbor algorithm for approximation a **discrete-valued target function** is given below:

```
Training algorithm:
    • For each training example ⟨x, f(x)⟩, add the example to the list training_examples

Classification algorithm:
    • Given a query instance $x_q$ to be classified,
        • Let $x_1 \ldots x_k$ denote the k instances from training_examples that are nearest to $x_q$
        • Return
              $\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
          where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.
```

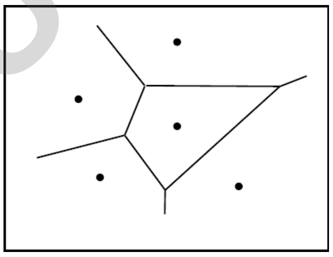
- The value $\hat{f}(x_q)$ returned by this algorithm as its estimate of $f(x_q)$ is just the most common value of f among the k training examples nearest tox_q.
- If k = 1, then the 1- Nearest Neighbor algorithm assigns to $\hat{f}(x_q)$ the value f(xi). Where x_i is the training instance nearest tox_q.
- Forlargervaluesofk,thealgorithmassignsthemostcommonvalueamongtheknearest trainingexamples.

- Belowfigureillustratestheoperationofthek-NearestNeighboralgorithmforthecasewhere theinstancesarepointsinatwo-dimensionalspaceandwherethetargetfunctionisBoolean valued.



- Thepositiveandnegativetrainingexamplesareshownby"+"and"-"respectively.A query point x_q is shown as well.
- The 1-Nearest Neighbor algorithm classifies x_q as a positive example in this figure, whereas the 5-Nearest Neighbor algorithm classifies it as a negativeexample.

- Belowfigureshowstheshapeofthis**decisionsurface**inducedby1-NearestNeighborover the entire instance space. The decision surface is a combination of convex polyhedra surrounding each of the trainingexamples.



- For every training example, the polyhedron indicates the set of query points whose classification will be completely determined by that training example. Query points outside the polyhedron are closer to some other training example. This kind ofdiagram is often called the ***Voronoi diagram*** of the set of trainingexample

The K- Nearest Neighbor algorithm for approximation a **real-valued target function** is givenbelow $f : \Re^n \rightarrow \Re$

```
Training algorithm:
    • For each training example ⟨x, f(x)⟩, add the example to the list training_examples
Classification algorithm:
    • Given a query instance $x_q$ to be classified,
        • Let $x_1 \ldots x_k$ denote the k instances from training_examples that are nearest to $x_q$
        • Return
              $\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$
```

Distance-Weighted Nearest Neighbor Algorithm

- The refinement to the k-NEAREST NEIGHBOR Algorithm is to weight the contributionofeachofthekneighborsaccordingtotheirdistancetothequerypointx_q, giving greater weight to closerneighbors.
- Forexample,inthek-NearestNeighboralgorithm,whichapproximatesdiscrete-valued target functions, we might weight the vote of each neighbor according to the inverse square of its distance fromx_q

Distance-Weighted Nearest Neighbor Algorithm for approximation a discrete-valued targetfunctions

```
Training algorithm:
    • For each training example ⟨x, f(x)⟩, add the example to the list training_examples
Classification algorithm:
    • Given a query instance $x_q$ to be classified,
        • Let $x_1 \ldots x_k$ denote the k instances from training_examples that are nearest to $x_q$
        • Return
              $\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$
      where
              $w_i \equiv \frac{1}{d(x_q, x_i)^2}$
```

Distance-Weighted Nearest Neighbor Algorithm for approximation a Real-valued targetfunctions

```
Training algorithm:
    • For each training example ⟨x, f(x)⟩, add the example to the list training_examples

Classification algorithm:
    • Given a query instance $x_q$ to be classified,
        • Let $x_1 \ldots x_k$ denote the k instances from training_examples that are nearest to $x_q$
        • Return
            $\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$
          where
            $w_i \equiv \frac{1}{d(x_q, x_i)^2}$
```

Terminology

- **Regression** means approximating a real-valued target function.
- **Residual** is the error $\hat{f}(x) - f(x)$ in approximating the targetfunction.
- **Kernel function** is the function of distance that is used to determine the weight ofeach training example. In other words, the kernel function is the function K suchthat $w_i = K(d(x_i, x_q))$

LOCALLY WEIGHTED REGRESSION

- The phrase "**locally weighted regression**" is called **local** because the functionis approximated based only on data near the query point, **weighted** because the contribution of each training example is weighted by its distance from the query point, and**regression**becausethisisthetermusedwidelyinthestatisticallearningcommunity for the problem of approximating real-valuedfunctions.

- Given a new query instance x_q, the general approach in locally weighted regression is to construct an approximation $\hat{f}$ that fits the training examples in the neighborhood surrounding x_q. This approximation is then used to calculate the value $\hat{f}(x_q)$, which is output as the estimated target value for the queryinstance.

Locally Weighted Linear Regression

- Considerlocallyweightedregressioninwhichthetargetfunction***f***isapproximatednearx_q using a linear function of theform

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \cdots + w_n a_n(x)$$

  Where, $a_i(x)$ denotes the value of the ith attribute of the instance x

- Derived methods are used to choose weights that minimize the squared error summed over the set D of training examples using gradient descent

$$E \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$

  Which led us to the gradient descent training rule

$$\Delta w_j = \eta \sum_{x \in D} (f(x) - \hat{f}(x)) a_j(x)$$

  Where, η is a constant learning rate

- Need to modify this procedure to derive a local approximation rather than a globalone. Thesimplewayistoredefinetheerrorcriterion Etoemphasizefittingthelocaltraining examples. Three possible criteria are givenbelow.

1. Minimize the squared error over just the k nearestneighbors:

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in \, k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 \qquad \textbf{equ(1)}$$

2. Minimize the squared error over the entire set D of training examples, while weighting the error of each training example by some decreasing function K of its distance from x_q:

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \; K(d(x_q, x)) \qquad \textbf{equ(2)}$$

3. Combine 1 and2:

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in \, k \text{ nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2 \; K(d(x_q, x)) \qquad \textbf{equ(3)}$$

If we choose criterion three and re-derive the gradient descent rule, we obtain the following training rule

$$\Delta w_j = \eta \sum_{x \in \text{ k nearest nbrs of } x_q} K(d(x_q, x))\ (f(x) - \hat{f}(x))\ a_j(x)$$

The differences between this new rule and the rule given by Equation (3) are that the contribution of instance x to the weight update is now multiplied by the distance penalty **$K(d(x_q, x))$,** and that the error is summed over only the k nearest training examples.

DECISION TREE LEARNING

Decisiontreelearningisamethodforapproximatingdiscrete-valuedtargetfunctions,inwhich the learned function is represented by a decisiontree.

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 theinstance.
- 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 thisattribute.
- An instance is classified by starting at the root node of the tree, testing the attribute specifiedbythisnode,thenmovingdownthetreebranchcorrespondingtothevalueof the attribute in the given example. This process is then repeated for the subtree rooted at the newnode.

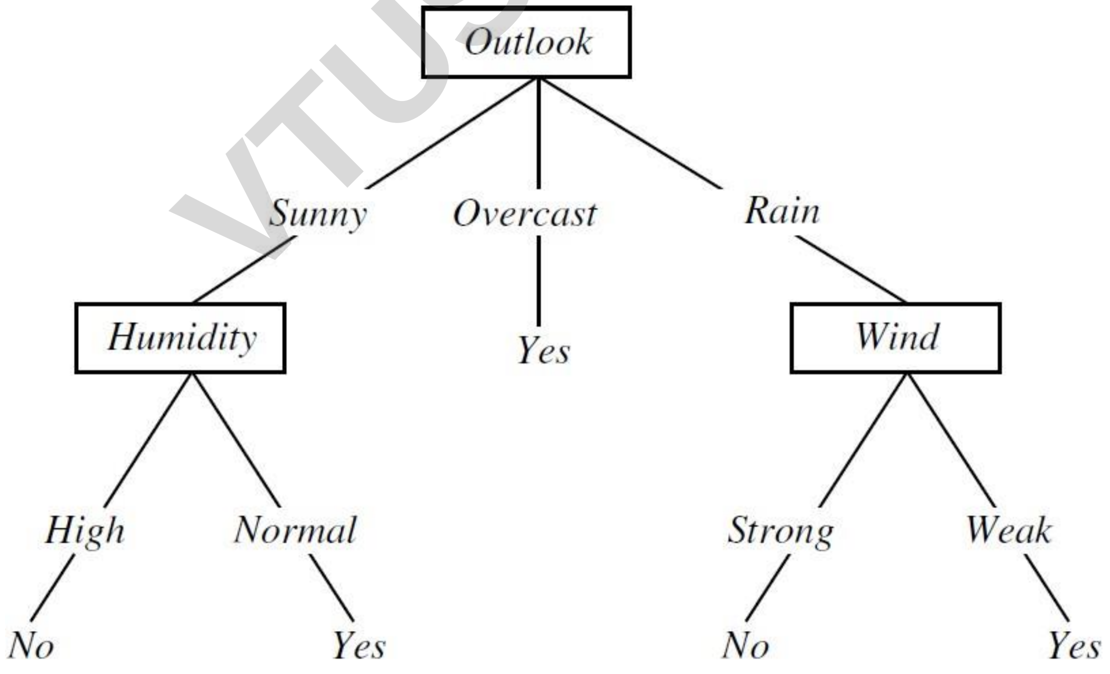


FIGURE: A decision tree for the concept ***PlayTennis***. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf

- 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 theseconjunctions

For example, the decision tree shown in above figure corresponds to the expression

$$\begin{aligned} &(Outlook = Sunny \wedge Humidity = Normal) \\ \vee \quad &(Outlook = Overcast) \\ \vee \quad &(Outlook = Rain \wedge Wind = Weak) \end{aligned}$$

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

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

1. ***Instances are represented by attribute-value pairs*** – Instances are described by a fixed set of attributes and theirvalues

2. ***The target function has discrete output values*** – The decision tree assigns a Boolean classification (e.g., yes or no) to each example. Decision tree methods easily extendto learning functions with more than two possible outputvalues.

3. ***Disjunctive descriptions may berequired***

4. ***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 theseexamples.

5. ***The training data may contain missing attribute values –*** Decision tree methods can be used even when some training examples have unknownvalues

THE BASIC DECISION TREE LEARNING ALGORITHM

The basic algorithm is ID3 which learns decision trees by constructing them top-down

```
ID3(Examples, Target_attribute, Attributes)

    Examples are the training examples. Target_attribute is the attribute whose value is tobe
    predicted by the tree. Attributes is a list of other attributes that may be tested by the
    learneddecisiontree.ReturnsadecisiontreethatcorrectlyclassifiesthegivenExamples.

• Create a Root node for thetree
• 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 inExamples

• Otherwise Begin
    • A ← the attribute from Attributes that best* classifiesExamples
    • The decision attribute for Root ←A
    • For each possible value, $v_i$, ofA,
        • 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 $v_i$forA
        • If Examples $_{v_i}$ , isempty
            • Then below this new branch add a leaf node with label = most common
              value of Target_attribute inExamples
            • Else below this new branch add the subtree
                  ID3(Examples $_{v_i}$, Targe_tattribute, Attributes –{A}))
• End
• ReturnRoot
```

* The best attribute is the one with highest informationgain

TABLE:SummaryoftheID3algorithmspecializedtolearningBoolean-valuedfunctions.ID3 isagreedyalgorithmthatgrowsthetreetop-down,ateachnodeselectingtheattributethatbest classifies the local training examples. This process continues until the tree perfectly classifies the training examples, or until all attributes have beenused

Which Attribute Is the Best Classifier?

- The central choice in the ID3 algorithm is selecting which attribute to test at each node in thetree.
- A statistical property called ***information gain*** that measures how well a given attribute separates the training examples according to their targetclassification.
- ID3 uses ***information gain*** measure to select among the candidate attributes at each step while growing thetree.

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

To define information gain, we begin by defining a measure called entropy. *Entropy measures the impurity of a collection 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

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

Where,

p_+ is the proportion of positive examples in S
$p-$ is the proportion of negative examples inS.

Example:
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

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

- The entropy is 0 if all members of S belong to the sameclass
- The entropy is 1 when the collection contains an equal number of positive andnegative examples
- If the collection contains unequal numbers of positive and negative examples, the entropy is between 0 and1

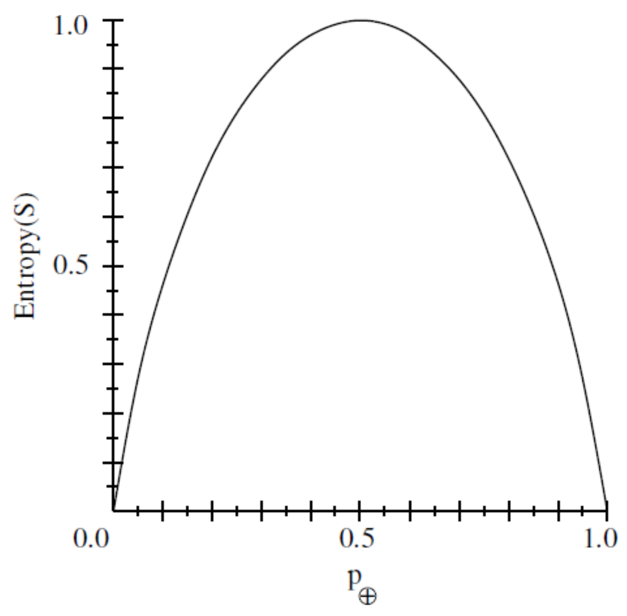


FIGURE The entropy function relative to a boolean classification, as the proportion, $p_\oplus$, of positive examples varies between 0 and 1.

INFORMATION GAIN MEASURES THE EXPECTED REDUCTION IN ENTROPY

- ***Information gain,*** is the expected reduction in entropy caused by partitioningthe examples according to thisattribute.
- The information gain, Gain(S, A) of an attribute A, relative to a collection ofexamples S, is definedas

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

<u>**Example:**</u>Information gain

Let, $Values(Wind) = \{Weak, Strong\}$

$S = [9+,5-]$

$S_{Weak} = [6+,2-]$

$S_{Strong} = [3+,3-]$

Information gain of attribute *Wind*:

$Gain(S, Wind) = Entropy(S) - 8/14\ Entropy\ (S_{Weak}) - 6/14\ Entropy(S_{Strong})$

$= 0.94 – (8/14)* 0.811 – (6/14) *1.00$

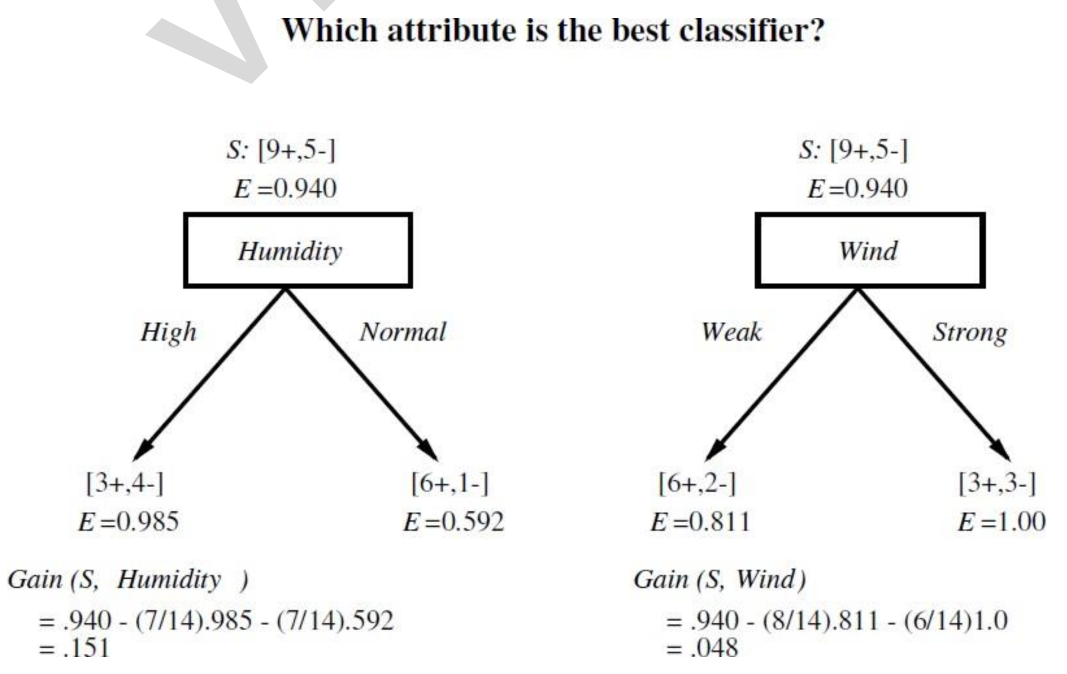
$= 0.048$

An Illustrative Example

- To illustrate the operation of ID3, consider the learning task represented by the training examples of belowtable.
- Herethetargetattribute***PlayTennis***,whichcanhavevalues***yes***or***no***fordifferentdays.
- Considerthefirststepthroughthealgorithm,inwhichthetopmostnodeofthedecision tree iscreated.

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

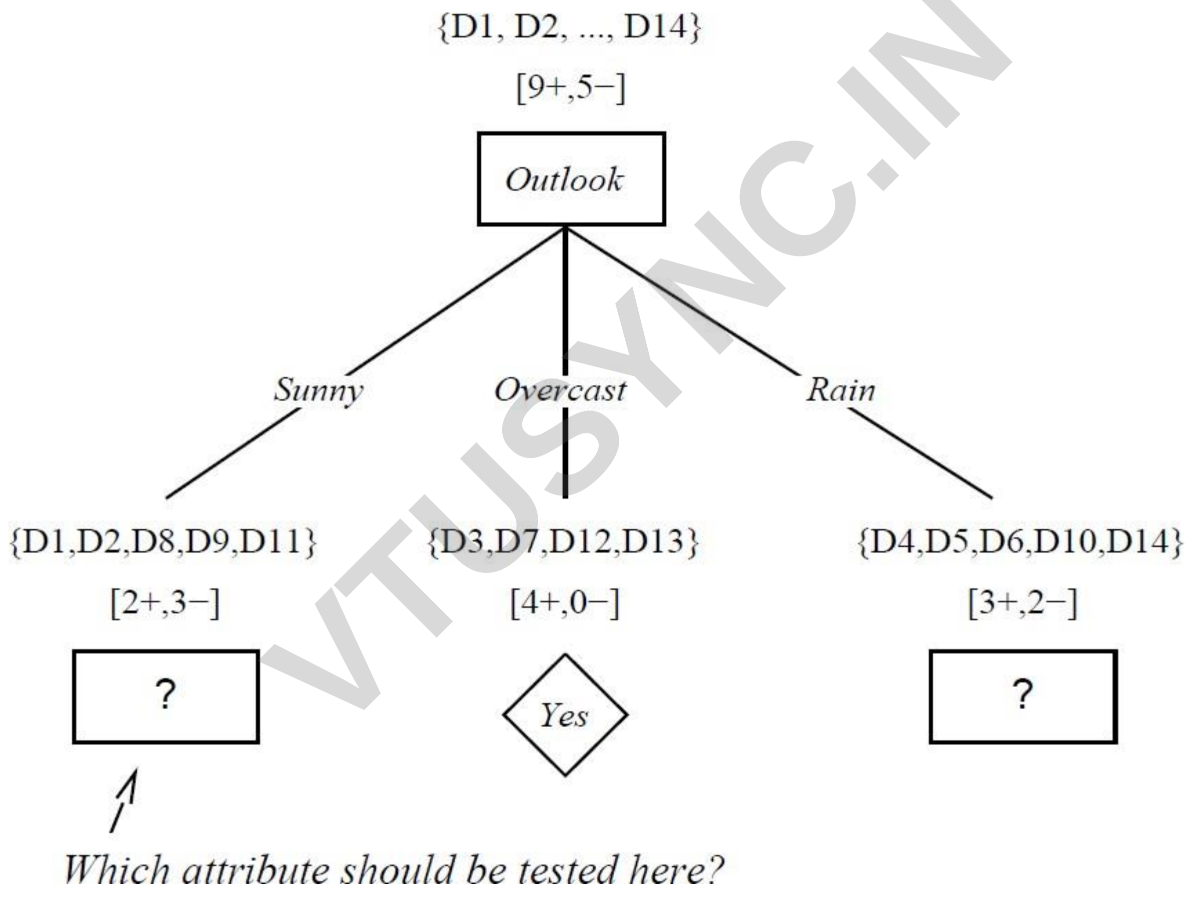
- ID3 determines the information gain for each candidate attribute (i.e., Outlook, Temperature,Humidity,andWind),thenselectstheonewithhighestinformationgain.



- The information gain values for all four attributesare

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

- According to the information gain measure, the ***Outlook*** attribute provides the best prediction of the target attribute, ***PlayTennis***, over the training examples. Therefore, ***Outlook*** is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values i.e., Sunny, Overcast, andRain.



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

$$Gain(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$$

$$Gain(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$$

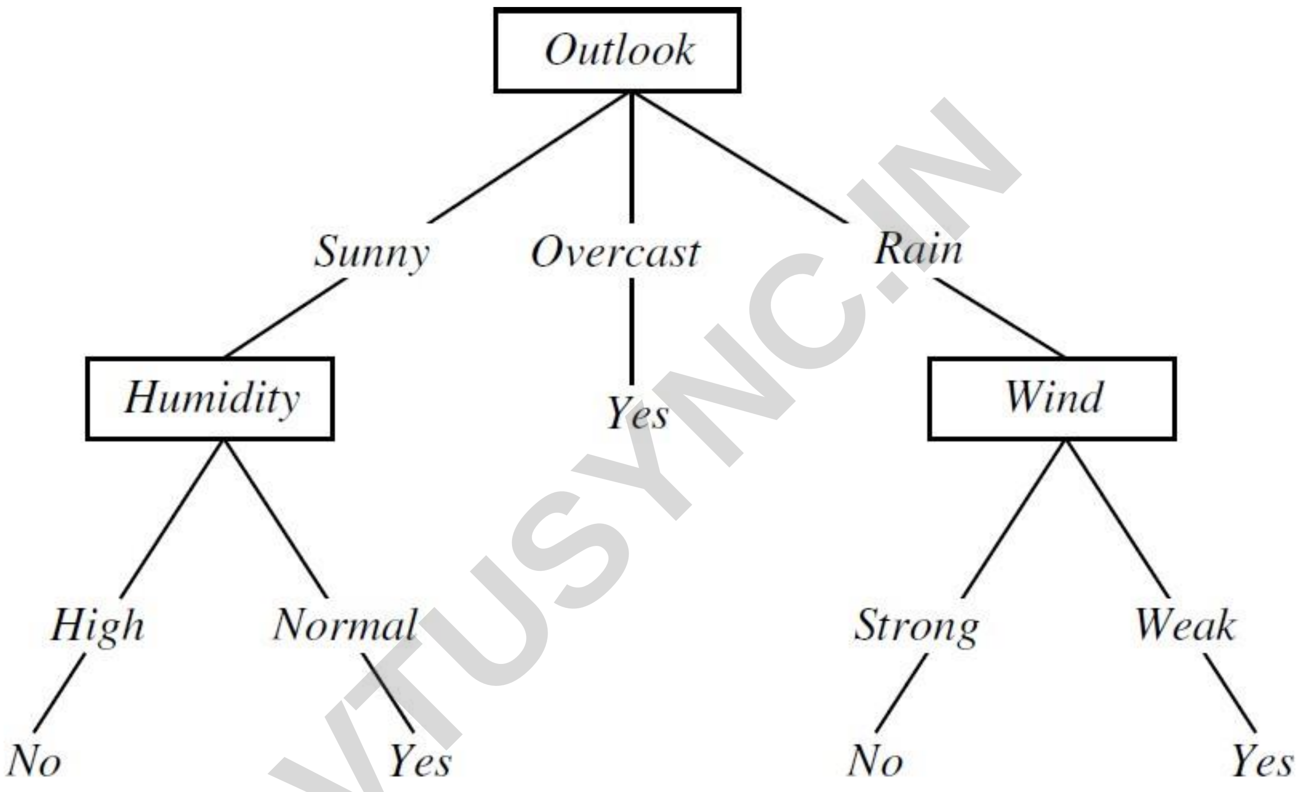
$$Gain(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$$

S_{Rain} = { D4, D5, D6, D10, D14}

Gain (S_{Rain} , *Humidity*) = 0.970 – (2/5)1.0 – (3/5)0.917 = 0.019

Gain (S_{Rain} , *Temperature*) =0.970 – (0/5)0.0 – (3/5)0.918 – (2/5)1.0 = 0.019

Gain (S_{Rain} , *Wind*) =0.970 – (3/5)0.0 – (2/5)0.0 = 0.970



HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- ID3canbecharacterizedassearchingaspaceofhypothesesforonethatfitsthetraining examples.
- The hypothesis space searched by ID3 is the set of possible decisiontrees.
- 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 trainingdata

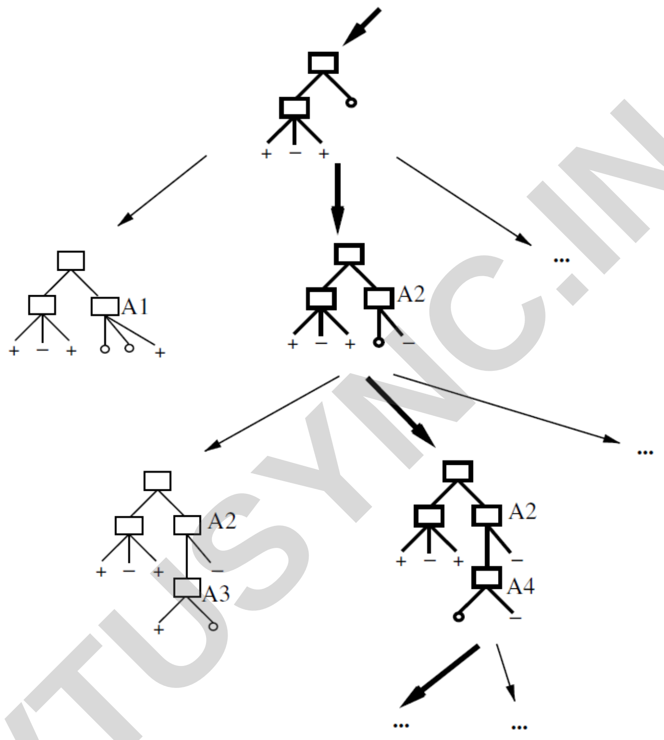


Figure: Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic.

<u>By viewing ID3 in terms of its search space and search strategy, there are some insight into itscapabilities and limitations</u>

1. *ID3'shypothesisspaceofalldecisiontreesisacompletespaceoffinitediscrete-valued functions, relative to the available attributes. Because every finite discrete-valued function can be represented by some decisiontree* ID3avoidsoneofthemajorrisksofmethodsthatsearchincompletehypothesisspaces: that the hypothesis space might not contain the targetfunction.

2. *ID3 maintains only a single current hypothesis as it searches through the space of decisiontrees.*
   <u>For example,</u>with the earlier version space candidate elimination method, which maintains the set of all hypotheses consistent with the available training examples.

   By determining only a single hypothesis, ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses.
   <u>For example</u>, it does not have the ability to determine how many alternative decision treesareconsistentwiththeavailabletrainingdata,ortoposenewinstancequeriesthat optimally resolve among these competinghypotheses

3. *ID3initspureformperformsnobacktrackinginitssearch.Onceitselectsanattribute to test at a particular level in the tree, it never backtracks to reconsider thischoice*.
   In the case of ID3, a locally optimal solution corresponds to the decision tree it selects along the single search path it explores. However, this locally optimal solution may be less desirable than trees that would have been encountered along a different branch of the search.

4. *ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its currenthypothesis*.
   One advantage of using statistical properties of all the examples is that the resulting search is much less sensitive to errors in individual training examples.
   ID3 can be easily extended to handle noisy training data by modifying its termination criterion to accept hypotheses that imperfectly fit the training data.

INDUCTIVE BIAS IN DECISION TREE LEARNING

Inductive bias is the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances

Givenacollectionoftrainingexamples,therearetypicallymanydecisiontreesconsistentwith these examples. Which of these decision trees does ID3choose?

<u>ID3 search strategy</u>

- Selects in favour of shorter trees over longerones
- Selects trees that place the attributes with highest information gain closest to theroot.

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

- Consideranalgorithmthatbeginswiththeemptytreeandsearchesbreadthfirstthrough progressively more complextrees.
- First considering all trees of depth 1, then all trees of depth 2,etc.
- Once it finds a decision tree consistent with the training data, it returns the smallest consistent tree at that search depth (e.g., the tree with the fewestnodes).
- Let us call this breadth-first search algorithmBFS-ID3.
- BFS-ID3 finds a shortest decision tree and thus exhibits the bias "shorter trees are preferred over longertrees.

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

- ID3 can be viewed as an efficient approximation to BFS-ID3, using a greedy heuristic search to attempt to find the shortest tree without conducting the entire breadth-first search through the hypothesisspace.
- Because ID3 uses the information gain heuristic and a hill climbing strategy, itexhibits a more complex bias thanBFS-ID3.
- Inparticular,itdoesnotalwaysfindtheshortestconsistenttree,anditisbiasedtofavour trees that place attributes with high information gain closest to theroot.

Restriction Biases and Preference Biases

Difference between the types of inductive bias exhibited by ID3 and by the CANDIDATE-ELIMINATION Algorithm.

ID3:

- ID3 searches a complete hypothesisspace
- It searches incompletely through this space, from simple to complex hypotheses, until its termination condition ismet
- Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy. Its hypothesis space introduces no additionalbias

CANDIDATE-ELIMINATION Algorithm:

- The version space CANDIDATE-ELIMINATION Algorithm searches an incomplete hypothesis space
- 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 additionalbias

***Preference bias* –** The inductive bias of ID3 is a preference for certain hypotheses over others (e.g.,preferenceforshorterhypothesesoverlargerhypotheses),withnohardrestrictiononthe hypotheses that can be eventually enumerated. This form of bias is called a preference bias or a searchbias.

Restrictionbias–ThebiasoftheCANDIDATEELIMINATIONalgorithmisintheformofa categoricalrestrictiononthesetofhypothesesconsidered.Thisformofbiasistypicallycalled a restriction bias or a languagebias.

Which type of inductive bias is preferred in order to generalize beyond the training data, apreference bias or restriction bias?

- A preference bias is more desirable than a restriction bias, because it allows the learner toworkwithinacompletehypothesisspacethatisassuredtocontaintheunknowntarget function.
- In contrast, 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 functionaltogether.

Why Prefer Short

Hypotheses? Occam's razor

- Occam's razor: is the problem-solving principle that the simplest solution tends to be the right one. When presented with competing hypotheses to solve a problem, one should select the solution with the fewestassumptions.

- Occam's razor: “Prefer the simplest hypothesis that fits thedata”.

Argument in favour of Occam’s razor:

- Fewer short hypotheses than longones:
  - Short hypotheses fits the training data which are less likely to becoincident
  - Longer hypotheses fits the training data might becoincident.

- Many complex hypotheses that fit the current training data but fail to generalize correctly to subsequentdata.

Argument opposed:

- There are few small trees, and our priori chance of finding one consistent with an arbitrary set of data is therefore small. The difficulty here is that there are very many small sets of hypotheses that one can define but understood by fewerlearner.
- The size of a hypothesis is determined by the representation used internally by the learner. Occam's razor will produce two different hypotheses from the same training examples when it is applied by two learners, both justifying their contradictory conclusions by Occam's razor. On this basis we might be tempted to reject Occam's razoraltogether.

ISSUES IN DECISION TREE LEARNING

Issues in learning decision trees include

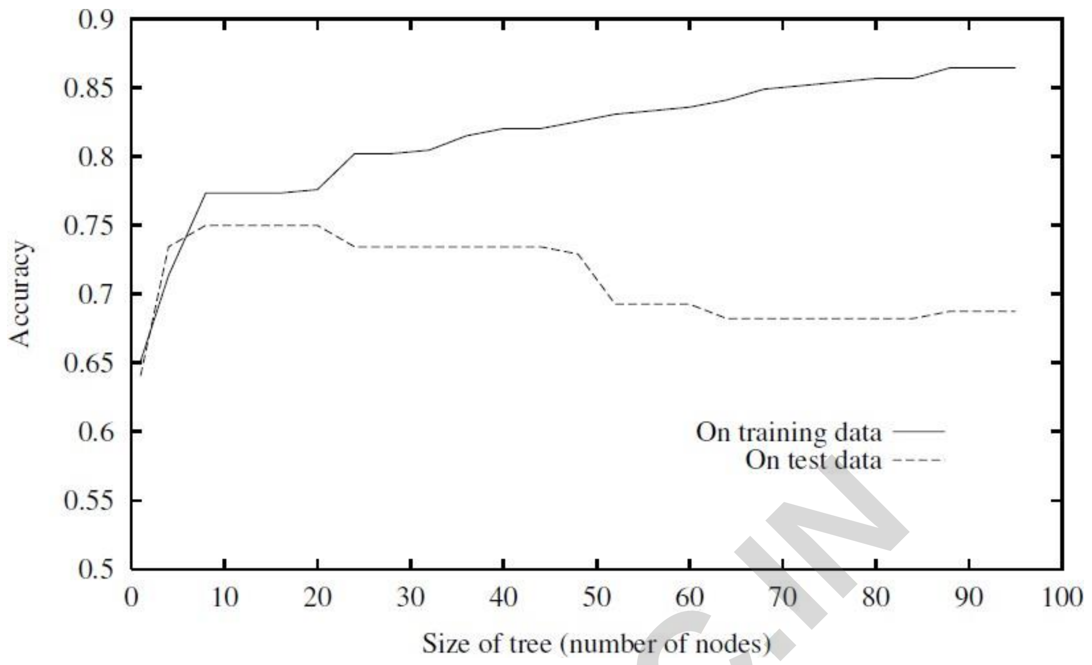
1. Avoiding Overfitting theData
   - Reduced error pruning
   - Rule post-pruning
2. Incorporating Continuous-Valued Attributes
3. Alternative Measures for SelectingAttributes
4. Handling Training Examples with Missing AttributeValues
5. Handling Attributes with DifferingCosts

1. Avoiding Overfitting theData

- TheID3algorithmgrowseachbranchofthetreejustdeeplyenoughtoperfectlyclassify the training examples but 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. This algorithm can produce trees that overfit the training examples.

- ***Definition - Overfit:***Given a hypothesis space H, a hypothesis h ∈ H is said to overfit the training data if there exists some alternative hypothesis h' ∈ 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 ofinstances.

The below figure illustrates the impact of overfitting in a typical application of decision tree learning.



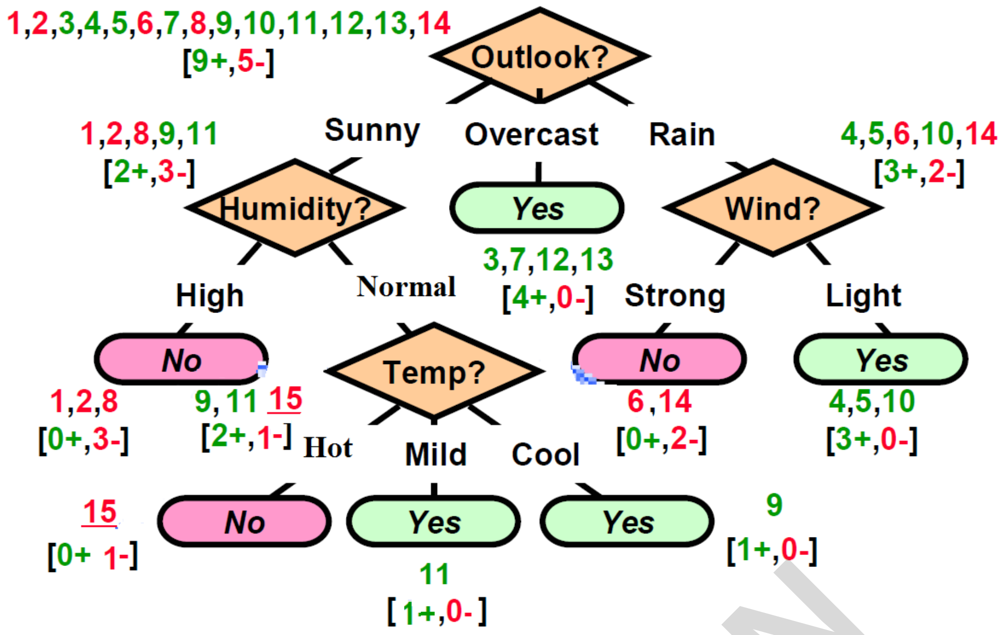
- *The horizontal axis* of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by thetree.
- *The solid line* shows the accuracy of the decision tree over the training examples. The broken line shows accuracy measured over an independent set of testexample
- The accuracy of the tree over the training examples increases monotonically as the tree is grown. The accuracy measured over the independent test examples first increases, thendecreases.

Howcanitbepossiblefortreehtofitthetrainingexamplesbetterthanh',butforittoperform more poorly over subsequentexamples?

1. Overfitting can occur when the training examples contain random errors ornoise
2. When small numbers of examples are associated with leafnodes.

<u>Noisy Training Example</u>

- Example 15: <Sunny, Hot, Normal, Strong,->
- Example is noisy because the correct label is+
- Previously constructed tree misclassifiesit



Approaches to avoiding overfitting in decision tree learning

- Pre-pruning(avoidance):Stopgrowingthetreeearlier,beforeitreachesthepointwhere it perfectly classifies the trainingdata
- Post-pruning (recovery): Allow the tree to overfit the data, and then post-prune thetree

Criterion used to determine the correct final tree size

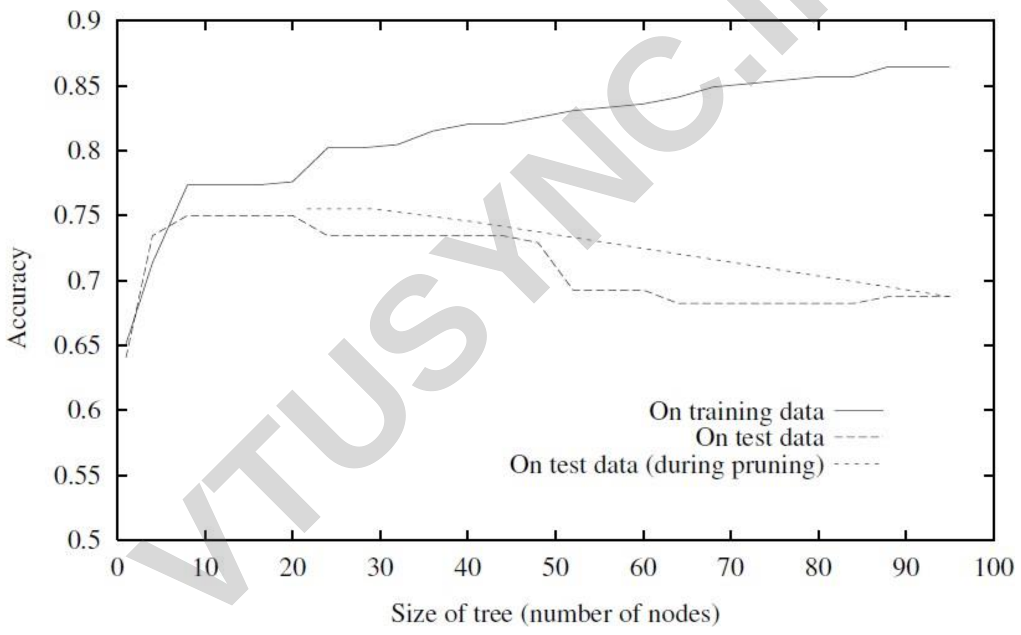
- Useaseparatesetofexamples,distinctfromthetrainingexamples,toevaluatetheutility of post-pruning nodes from thetree
- Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set
- Usemeasureofthecomplexityforencodingthetrainingexamplesandthedecisiontree, haltinggrowthofthetreewhenthisencodingsizeisminimized.Thisapproachiscalled the Minimum DescriptionLength

MDL – Minimize : size(tree) + size (misclassifications(tree))

Reduced-Error Pruning

- Reduced-error pruning, is to consider each of the decision nodes in the tree to be candidates forpruning
- ***Pruning*** a decision node consists of removing the subtree rooted at that node, making italeafnode,andassigningitthemostcommonclassificationofthetrainingexamples affiliated with thatnode
- Nodesareremovedonlyiftheresultingprunedtreeperformsnoworsethan-theoriginal over the validationset.
- Reduced error pruning has the effect that any leaf node added due to coincidental regularitiesinthetrainingsetislikelytobeprunedbecausethesesamecoincidencesare unlikely to occur in the validationset

Theimpactofreduced-errorpruningontheaccuracyofthedecisiontreeisillustratedinbelow figure



- Theadditionallineinfigureshowsaccuracyoverthetestexamplesasthetreeispruned. When pruning begins, the tree is at its maximum size and lowest accuracy over the test set.Aspruningproceeds,thenumberofnodesisreducedandaccuracyoverthetestset increases.
- Theavailabledatahasbeensplitintothreesubsets:thetrainingexamples,thevalidation examples used for pruning the tree, and a set of test examples used to provide an unbiased estimate of accuracy over future unseen examples. The plot shows accuracy over the training and testsets.

Pros and Cons

Pro: Produces smallest version of most accurate *T* (subtree of *T*)

Con: Uses less data to construct *T*

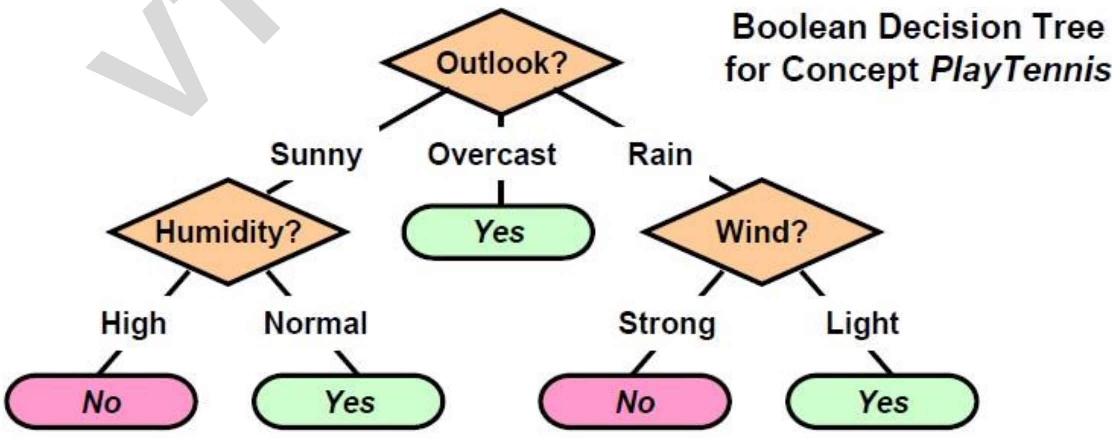
Can afford to hold out $D_{validation}$?. If not (data is too limited), may make error worse (insufficient D_{train})

Rule Post-Pruning

Rule post-pruning is successful method for finding high accuracy hypotheses

- Rule post-pruning involves the followingsteps:
- Inferthedecisiontreefromthetrainingset,growingthetreeuntilthetrainingdataisfit as well as possible and allowing overfitting to occur.
- Convertthelearnedtreeintoanequivalentsetofrulesbycreatingoneruleforeachpath from the root node to a leafnode.
- Prune (generalize) each rule by removing any preconditions that result in improving its estimatedaccuracy.
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequentinstances.

Converting a Decision Tree into Rules



Example

- IF (*Outlook* = *Sunny*) $\wedge$ (*Humidity* = *High*) THEN *PlayTennis* = *No*
- IF (*Outlook* = *Sunny*) $\wedge$ (*Humidity* = *Normal*) THEN *PlayTennis* = *Yes*
- …

For example, consider the decision tree. The leftmost path of the tree in below figure is translated into the rule.

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

Given the above rule, rule post-pruning would consider removing the preconditions
(Outlook = Sunny) and (Humidity = High)

- It would select whichever of these pruning steps produced the greatest improvement in estimated rule accuracy, then consider pruning the second precondition as a further pruningstep.
- No pruning step is performed if it reduces the estimated ruleaccuracy.

There are three main advantages by converting the decision tree to rules before pruning

1. 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 eachpath.
2. 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, it 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.
3. Converting to rules improves readability. Rules are often easier for tounderstand.

2. Incorporating Continuous-Valued Attributes

Continuous-valued decision attributes can be incorporated into the learned tree.

There are two methods for Handling Continuous Attributes
1. Define new discrete valued attributes that partition the continuous attribute value into a discrete set ofintervals.
E.g., {high ≡ Temp > 35º C, med ≡ 10º C < Temp ≤ 35º C, low ≡ Temp ≤ 10º C}

2. Using thresholds for splittingnodes
e.g., A ≤ a produces subsets A ≤ a and A > a

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

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

- Pick a threshold, c, that produces the greatest informationgain
- In the current example, there are two candidate thresholds, corresponding to the values ofTemperatureatwhichthevalueofPlayTennischanges:(48+60)/2,and(80+90)/2.
- The information gain can then be computed for each of the candidateattributes, Temperature $_{>54}$, and Temperature $_{>85}$ and the best can be selected (Temperature$_{>54}$)

3. Alternative Measures for SelectingAttributes

- The problem is if attributes with many values, Gain will select it?
- Example:considertheattributeDate,whichhasaverylargenumberofpossiblevalues. (e.g., March 4,1979).
- If this attribute is added to the PlayTennis data, it would have the highest information gain of any of the attributes. This is because Date alone perfectly predicts the target attribute over the training data. Thus, it would be selected as the decision attribute for the root node of the tree and lead to a tree of depth one, which perfectly classifies the trainingdata.
- This decision tree with root node Date is not a useful predictor because it perfectly separates the training data, but poorly predict on subsequentexamples.

<u>One Approach: Use GainRatio instead of Gain</u>

Thegainratiomeasurepenalizesattributesbyincorporatingasplitinformation,thatissensitive to how broadly and uniformly the attribute splits thedata

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

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

Where, Si is subset of S, for which attribute A has value vi

4. Handling Training Examples with Missing AttributeValues

The data which is available may contain missing values for some
attributes Example: Medical diagnosis

- <Fever = true, Blood-Pressure = normal, …, Blood-Test = ?,…>
- Sometimes values truly unknown, sometimes low priority (or cost toohigh)

Strategies for dealing with the missing attribute value

- IfnodentestA,assignmostcommonvalueofAamongothertrainingexamplessorted to noden
- AssignmostcommonvalueofAamongothertrainingexampleswithsametargetvalue
- AssignaprobabilitypitoeachofthepossiblevaluesviofAratherthansimplyassigning the most common value toA(x)

5. Handling Attributes with DifferingCosts

- In some learning tasks the instance attributes may have associatedcosts.
- For example: In learning to classify medical diseases, the patients described in termsof attributes such as Temperature, BiopsyResult, Pulse, BloodTestResults,etc.
- Theseattributesvarysignificantlyintheircosts,bothintermsofmonetarycostandcost to patientcomfort
- Decision trees use low-cost attributes where possible, depends only on high-cost attributes only when needed to produce reliableclassifications

How to Learn A Consistent Tree with Low Expected Cost?

One approach is replace Gain by Cost-Normalized-Gain

Examples of normalization functions

- Tan and Schlimmer

$$\frac{Gain^2(S,A)}{Cost(A)}.$$

- Nunez

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A)+1)^w}$$

where $w \in [0,1]$ determines importance of cost