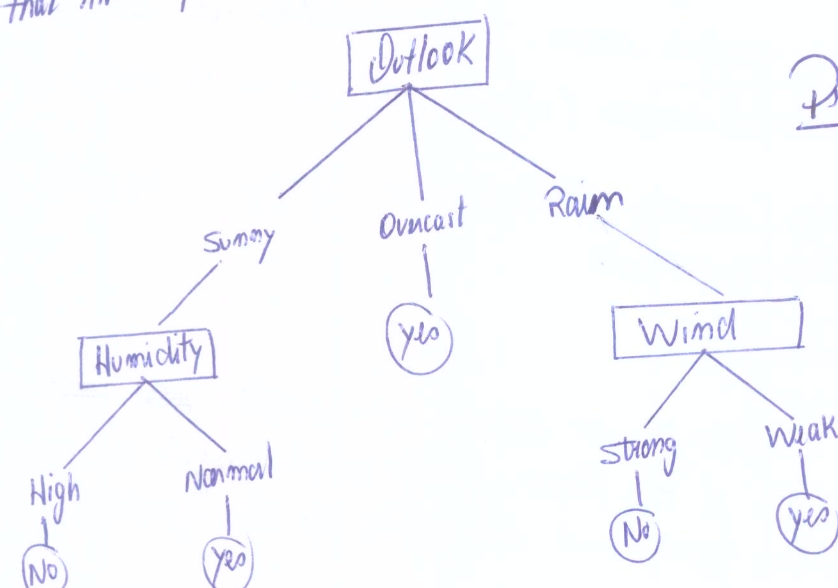
Decision tree learning

mitchell, chap. 3.

1. most widely and practical method of inductive inference
   the process of reaching a general conclusion from specific examples.
2. It is a method to approximate discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions.
3. we will see two tree algorithms in special: ID3 and C4.5
4. Learned trees can be represented as if-then-elses. to improve human readability.
5. wide success specially in medical diagnosis and credit analysis.

Representation

1. Decision trees classify instances by sorting them down the tree from the root to a leaf which provides the classification of the instance.
2. Each node in the tree specify a test of some attribute of the instance and each branch descending from that node represent the possible values of that attribute.



Problem: play tennis outside today?

Consider the instance: <outlook = Sunny, temp = Hot, Humidity = high, wind = strong>

would have $\boxed{\text{play tennis} = \text{No}}$

③ In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from root to leaf is a conjunction of attribute tests and the tree itself is a disjunction of these conjunctions.

④ The tree we just saw is equivalent to

(Outlook = Sunny AND Humidity = Normal)

or

(Outlook = Overcast)

or

(Outlook = Rain AND Wind = Weak).

Appropriate problems for decision trees

DTs are best suited for problems with the following characteristics:

① Instances are represented by discrete-values (fixed set of values per attribute)
↳ there are exceptions with rules for dealing with continuous values).

② The target function is discrete (fixed set of values).
↳ There are extensions for real-valued outputs but they are less common.

③ Disjunctive descriptions may be required (no sharing of attribute values in an instance).

④ The training data may contain errors. DTs algorithms are reasonably robust to errors in the annotation of examples (class outcome) or in the values of attributes.

⑤ Training data may contain missing attribute values. Example: humidity of the day is known for only some training examples.

The basic DT Learning algorithm

① most algorithms in the literature for DT learning are variations ~~of a~~ ~~single~~ on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. This approach is exemplified by the algorithm ID3 and its successor C4.5.

② ID3 learns decision trees by constructing them top-down beginning with the question: which attribute should be the root of the tree?

For answering this question, we evaluate each attribute with a statistical test to determine how well it classifies the data alone. The best attribute is selected and used as the test at the root.

③ A descendant of the root node is created for all values of this attribute. The entire process is repeated using the training examples associated with each descendant node to select the best attribute to test at that point of the tree.

④ This is a greedy algorithm with no backtrack whatsoever.

which attribute is the best classifier

① The central question in ID3 is how to pick the best attribute.

② For that we will define a statistical property called information gain that measures how well a given attribute separates the training examples according to their target classification.

$\boxed{\text{Entropy} \Rightarrow \text{measures homogeneity of examples}}$

To define IG, let's start with the measure Entropy (H) that characterizes the (im)purity of an arbitrary collection of examples.

Example : Given a collection S with $\oplus$ and $\ominus$ examples of a problem, the entropy of S is

$$H(S) \equiv -P_{\oplus} \cdot \log_2 P_{\oplus} - P_{\ominus} \cdot \log_2 P_{\ominus}$$

$P_{\oplus}$ proportion of $\oplus$ in training
$P_{\ominus}$ prop. of $\ominus$ in training.

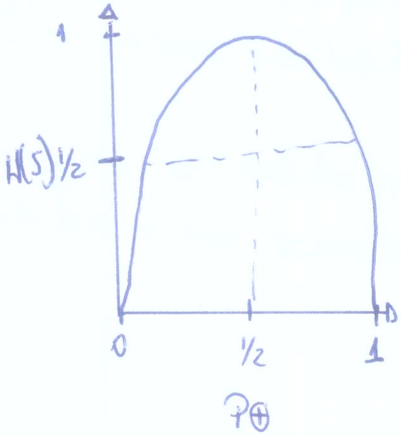
$* \; 0 \cdot \log 0 \Rightarrow 0$.

Suppose S has 14 examples, including 9 $\oplus$ and 5 $\ominus$. We adopt the notation $[9+, 5-]$ to summarize such sample of data.

$$H(S) = -9/14 \cdot \log 9/14 - 5/14 \log 5/14$$

$$= 0.940$$

Notes :- If all examples are $\oplus$ or $\ominus$, $H(S) = 0$.
- $H(S) = 1$ when $\underbrace{P_{\oplus} = P_{\ominus} = 1/2}$.



$H(S)$ as function of $p_\oplus$.

Entropy function relative to a boolean classification, as $p_\oplus$ varies between 0 and 1.

* From information theory, $H(S)$ measures the <u>min</u> number of bits of <u>info</u> needed to encode the classification of an arbitrary member of S. For example, if $p_\oplus = 1$, ~~the~~ one knows the drawn example will be positive, so no message needs to be sent, and the entropy is <u>zero</u>. On the other hand, if $p_\ominus = 1/2$, one bit is necessary to indicate whether the drawn example is positive or negative.

more generally, if a target attribute can take on c different values, $H(S)$ can be defined as

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

where p_i is the proportion of $S \in$ class i.

↳ Note $\log_2$ because entropy is a measure of the expected encoding length measured in <u>bits</u>

↳ Also as the target attribute can take on c possible values, $H(S)$ can be as large as $\boxed{\log_2 c}$ $c=2$, $\log_2 2 = 1$. $c=4$ $\log_2 4 = 2$ <u>etc</u>.

$\boxed{\text{Information Gain} \Rightarrow \text{measures the expected reduction in Entropy}}$

Given Entropy as a measure of impurity in a collection S, we can now measure the effectiveness of an attribute in classifying data. For that we define the measure called <u>information gain</u> IG.

IG is simply the <u>expected reduction in Entropy</u> caused by partitioning the examples according to this attribute.

more precisely, $G(S, A)$ of an attribute A relative to a collection S is defined as

$$G(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times H(S_v)$$

- $values(A)$ set of all possible values of A
- S_v is a subset of S for which A has value v (e.g., $S_v = \{s \in S \mid A(s) = v\}$.
- $H(S_v)$ is the sum of entropies of each subset S_v weighted by the fraction of examples $\frac{|S_v|}{|S|}$ that belong to S_v.

Example: Suppose S training examples days described by attributes wind which can have values weak or strong.

$S = [9+, 5-]$.

Suppose, 6 of the positive and 2 of the negative have $\boxed{wind = weak}$. The other 6 are $\boxed{wind = strong}$

$IG(S, wind) = ?$

$values(wind) = weak, strong$.

$S = [9+, 5-]$

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

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

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

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

$$= 0.940 - (8/14) \cdot 0.811 - (6/14) \cdot 1$$

$$= \boxed{0.048}$$

Example on a tree:

135

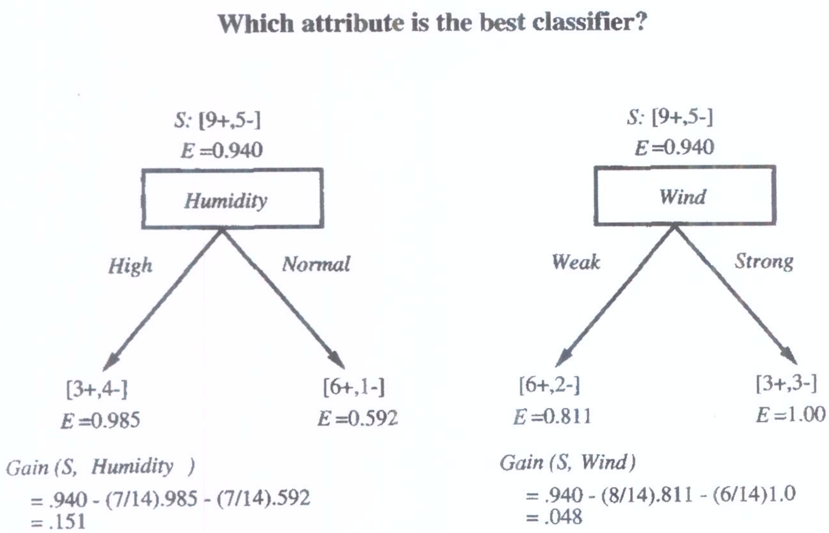


FIGURE 3.3
Humidity provides greater information gain than *Wind*, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, $[9+, 5-]$, sorting these by their *Humidity* produces collections of $[3+, 4-]$ $(Humidity = High)$ and $[6+, 1-]$ $(Humidity = Normal)$. The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute *Wind*.

3.4.2 An Illustrative Example

To illustrate the operation of ID3, consider the learning task represented by the training examples of Table 3.2. Here the target attribute *PlayTennis*, which can have values *yes* or *no* for different Saturday mornings, is to be predicted based on other attributes of the morning in question. Consider the first step through

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

G(S, outlook) = 0.246
G(S, Humidity) = 0.151
G(S, Wind) = 0.048
G(S, Temp) = 0.029

TABLE 3.2
Training examples for the target concept *PlayTennis*.

136

the algorithm, in which the topmost node of the decision tree is created. Which attribute should be tested first in the tree? ID3 determines the information gain for each candidate attribute (i.e., *Outlook, Temperature, Humidity*, and *Wind*), then selects the one with highest information gain. The computation of information gain for two of these attributes is shown in Figure 3.3. The information gain values for all four attributes are

$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$

where S denotes the collection of training examples from Table 3.2.

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*, and *Rain*). The resulting partial decision tree is shown in Figure 3.4, along with the training examples sorted to each new descendant node. Note that 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*. In contrast, the descendants corresponding to *Outlook* = *Sunny* and *Outlook* = *Rain* still have nonzero entropy, and the decision tree will be further elaborated below these nodes.

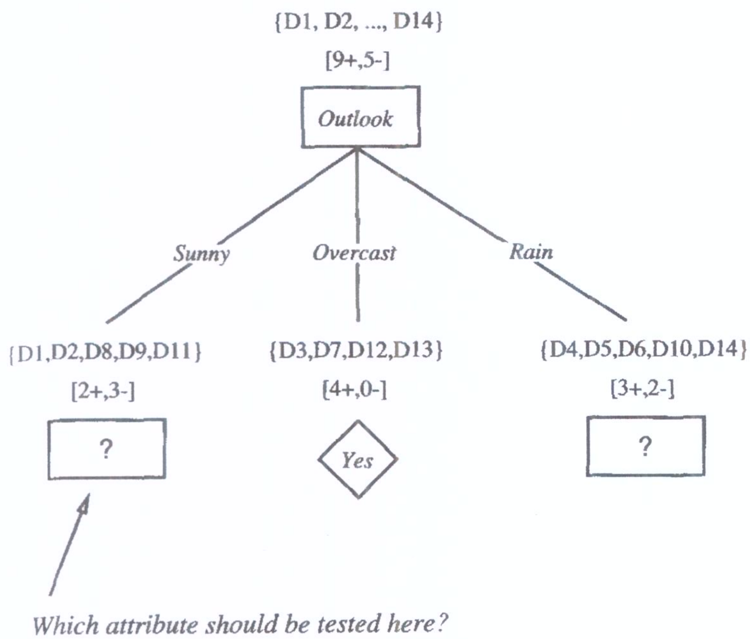
Attention here!

The process of selecting a new attribute and partitioning the training examples is now repeated for each nonterminal descendant node, this time using only the training examples associated with that 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 (i.e., their entropy is zero). Figure 3.4 illustrates the computations of information gain for the next step in growing the decision tree. The final decision tree learned by ID3 from the 14 training examples of Table 3.2 is shown in Figure 3.1.

3.5 HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

As with other inductive learning methods, 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

37



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

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

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

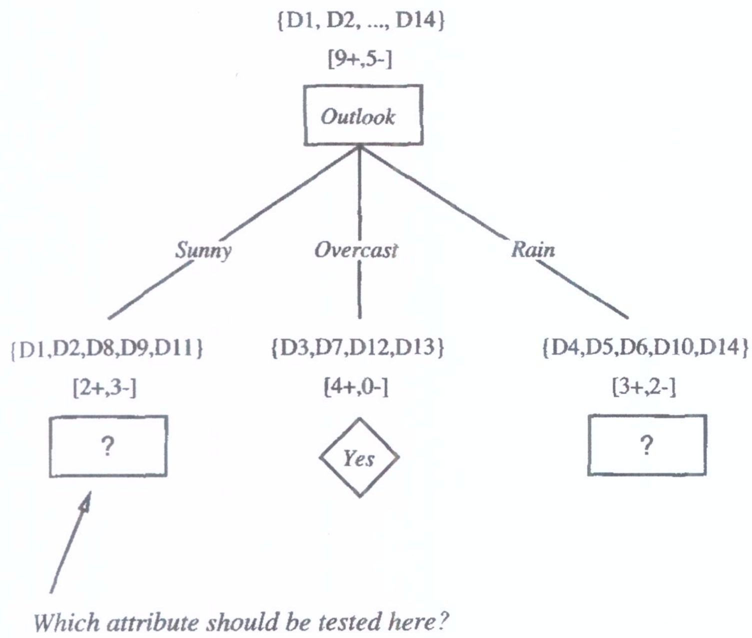
$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

FIGURE 3.4
The partially learned decision tree resulting from the first step of ID3. The training examples are sorted to the corresponding descendant nodes. The *Overcast* descendant has only positive examples and therefore becomes a leaf node with classification *Yes*. The other two nodes will be further expanded, by selecting the attribute with highest information gain relative to the new subsets of examples.

that guides this hill-climbing search is the information gain measure. This search is depicted in Figure 3.5.

By viewing ID3 in terms of its search space and search strategy, we can get some insight into its capabilities and limitations.

- ID3's hypothesis space of all decision trees is a *complete* space of finite discrete-valued functions, relative to the available attributes. Because every finite discrete-valued function can be represented by some decision tree, ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces (such as methods that consider only conjunctive hypotheses): that the hypothesis space might not contain the target function.
- ID3 maintains only a single current hypothesis as it searches through the space of decision trees. This contrasts, for example, 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



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

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

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

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

FIGURE 3.4
The partially learned decision tree resulting from the first step of ID3. The training examples are sorted to the corresponding descendant nodes. The *Overcast* descendant has only positive examples and therefore becomes a leaf node with classification *Yes*. The other two nodes will be further expanded, by selecting the attribute with highest information gain relative to the new subsets of examples.

that guides this hill-climbing search is the information gain measure. This search is depicted in Figure 3.5.

By viewing ID3 in terms of its search space and search strategy, we can get some insight into its capabilities and limitations.

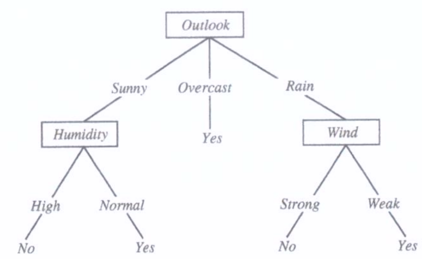
- ID3's hypothesis space of all decision trees is a *complete* space of finite discrete-valued functions, relative to the available attributes. Because every finite discrete-valued function can be represented by some decision tree, ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces (such as methods that consider only conjunctive hypotheses): that the hypothesis space might not contain the target function.
- ID3 maintains only a single current hypothesis as it searches through the space of decision trees. This contrasts, for example, 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

Decision Tree Learning

[read Chapter 3]
[recommended exercises 3.1, 3.4]

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

Decision Tree for *PlayTennis*



A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .0
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```


Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

When to Consider Decision Trees

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

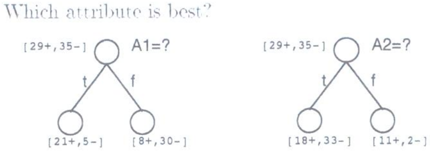
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Top-Down Induction of Decision Trees

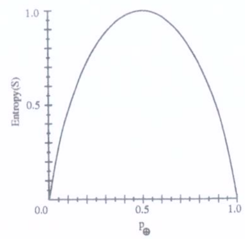
Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?



Entropy



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

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

Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of S:

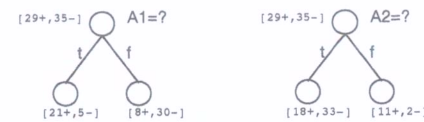
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

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

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on A

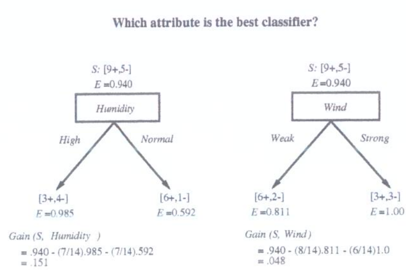
$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

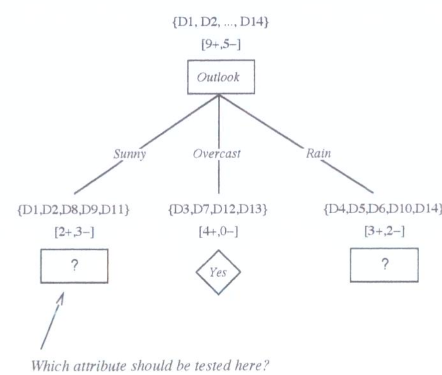


Training Examples

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

Selecting the Next Attribute





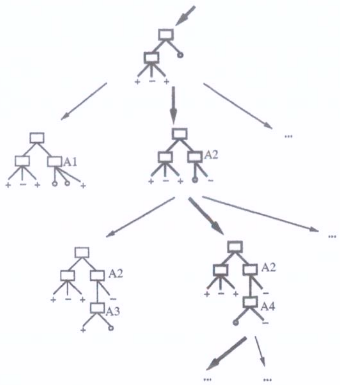
S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statisically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

Inductive Bias in ID3

Note H is the power set of instances X

→Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space H
- Occam's razor: prefer the shortest hypothesis that fits the data

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hyps. than long hyps.

→ a short hyp that fits data unlikely to be coincidence

→ a long hyp that fits data might be coincidence

Argument opposed:

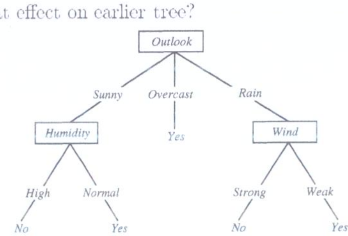
- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on *size* of hypothesis??

Overfitting in Decision Trees

Consider adding noisy training example #15:

$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$

What effect on earlier tree?



Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

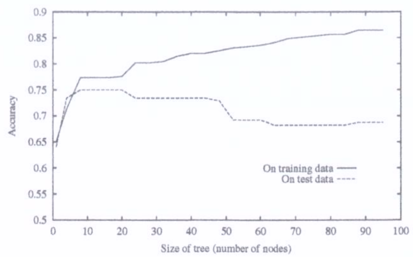
$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

140

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize $size(tree) + size(misclassifications(tree))$

Reduced-Error Pruning

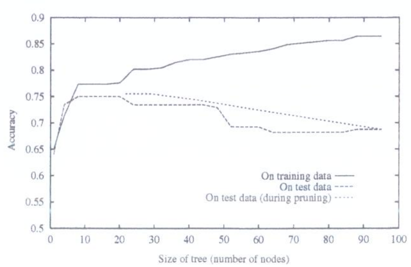
Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

Effect of Reduced-Error Pruning

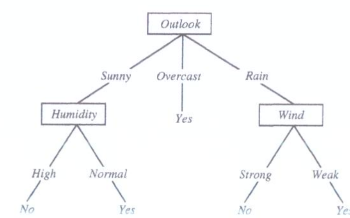


Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

Converting A Tree to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

...

Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

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

Attributes with Many Values

Problem:

- If attribute has many values, *Gain* will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use *GainRatio* instead

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

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

where S_i is subset of S for which A has value v_i

Attributes with Costs

Consider

- medical diagnosis, *BloodTest* has cost \$150
- robotics, $Width\_from\_1ft$ has cost 23 sec.

How to learn a consistent tree with low expected cost?
One approach: replace gain by

- Tan and Schlimmer (1990)

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

- Nunez (1988)

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

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

Unknown Attribute Values

What if some examples missing values of A?
Use training example anyway, sort through tree

- If node n tests A, assign most common value of A among other examples sorted to node n
- assign most common value of A among other examples with same target value
- assign probability p_i to each possible value v_i of A
  - assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion