ECOE554 Machine Learning	**Fall Term, 2003**
Lecture 10: November 3, 2003	
Lecturer: Deniz Yuret	*Scribe: Basak Mutlum*

1 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.

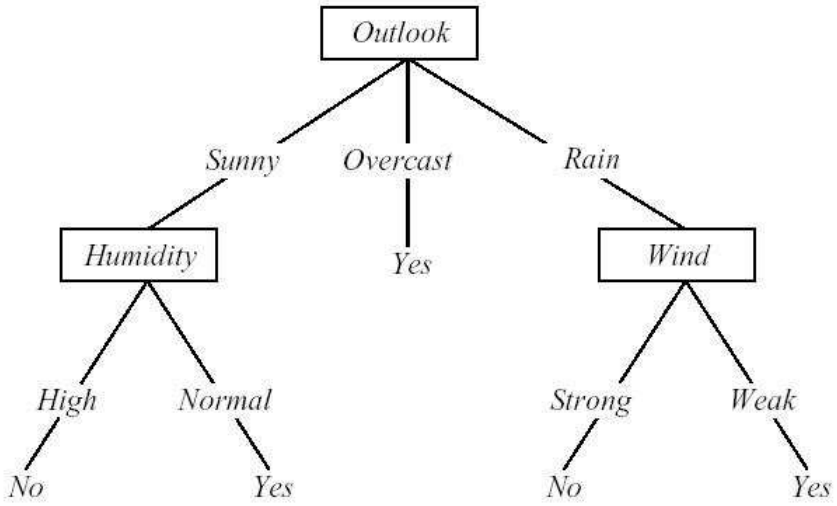


Figure 1: A decision tree for the concept *PlayTennis*.

2 Decision Tree Representation

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.In the 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

Decision trees represent a disjunction of conjunctions.

- Each path from root to a leaf is a conjunction of attribute tests.
- The tree itself is a disjunction of these conjunctions.

Figure1 illustrates a typical learned decision tree.It corresponds to the expression

$$\begin{gathered}(\text{ Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal }) \\ \vee \\ (\text{ Outlook} = \text{Overcast }) \\ \vee \\ (\text{ Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak })\end{gathered}$$

3 Appropriate Problems for Decision Tree Learning

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

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

Some examples of problems that fit to these characteristics are:

- Medical or equipment diagnosis
- Credit risk analysis
- Modelling calendar scheduling preferences

4 The Basic Decision Tree Learning Algorithm

The basic decision tree learning algorithm, ID3, employs a top-down, greedy search through the space of possible decision trees, beginning with the question "*which attribute should be tested at the root of the tree?*".The algorithm is given below:

ID3(Examples, Target-attribute, Attributes)
/* Examples: The training examples; */
/* Target-attribute:The attribute whose value is to be predicted by the tree; */
/* Attributes: A list of other attributes that may be tested by the learned decision tree. */
/* Return a decision tree that correctly classifies the given Examples */
Step 1: Create a Root node for the tree
Step 2: If all *Examples* are positive, Return the single-node tree *Root*,with label = +
Step 3: If all *Examples* are negative, Return the single-node tree *Root*,with label = -
Step 4: If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target-attribute* in *Examples*
Step 5: Otherwise Begin

- A ← the attribute from *Attributes* that best (i.e., highest information gain) 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*

We want to select the attribute that is most useful for classifying examples.In order to measure the worth of an attribute a statistical property is defined, ***information gain***, which measures how well a given attribute separates the training examples according to their target classification.

4.1 Entropy

In order to define information gain precisely,we begin by defining a measure called ***entropy***, that characterizes the (im)purity of an arbitrary collection of examples. That is, it measures the homogeneity of examples.

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

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

4.2 Information Gain

Information gain is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

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

Values(A) is the set of all possible values for attribute A
S_v is the subset os S for which attribute A has value v

4.3 An Illustrative Example

To illustrate the operation of ID3, let's consider the learning task represented by the below 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**

In the first step of the algorithm ,the topmost node of the decision tree is created.In order to determine the attribute that should be tested first in the tree,the information gain for attributes (*Outlook , Temperature ,Humidity* and *Wind*) are determined. The computation of information gain for *Humidity* and *Wind* is shown below.

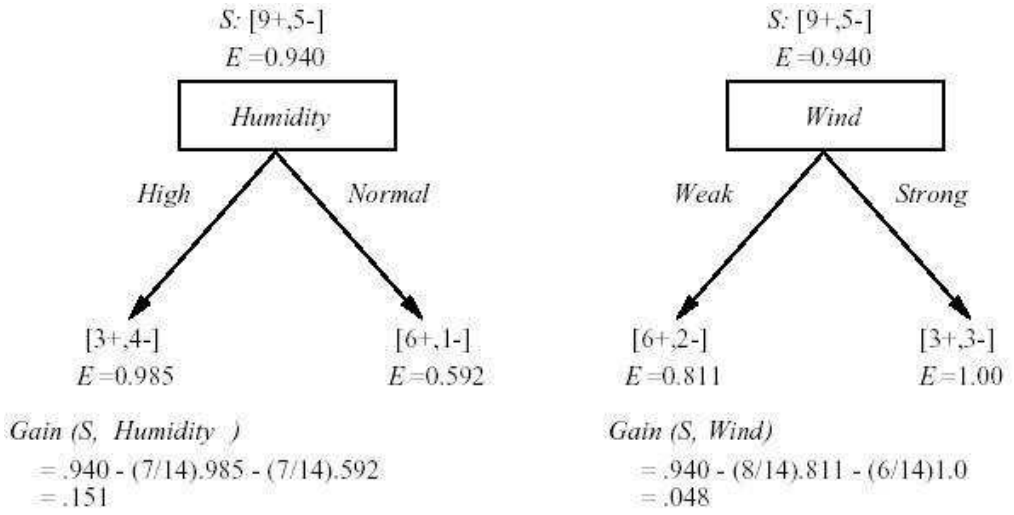


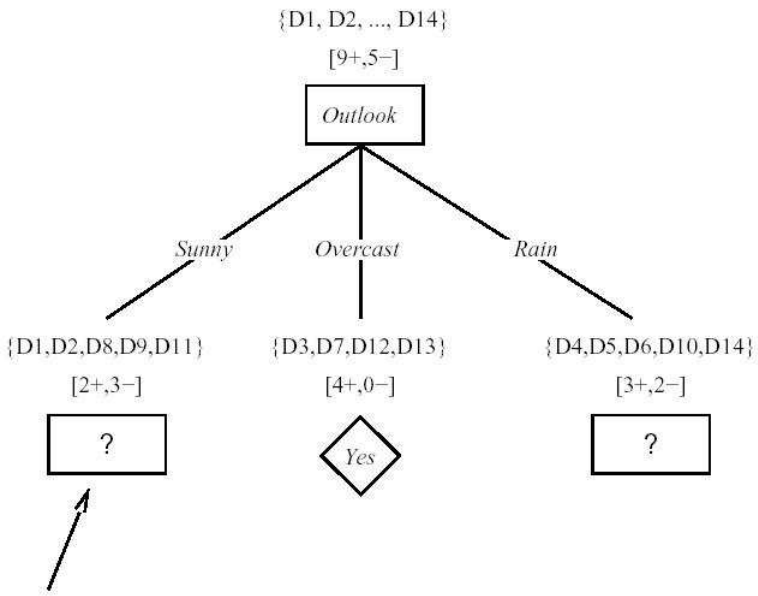
Figure 2: Information gain for *Humidity* and *Wind*.

The information gain for all four attributes are:

$$\text{Gain(S,Outlook)} = 0.246$$
$$\text{Gain(S,Humidity)} = 0.151$$
$$\text{Gain(S,Wind)} = 0.048$$
$$\text{Gain(S,Temperature)} = 0.029$$

Since *Outlook* attribute provides the best prediction, it is selected as the decision attribute for the root node.

In Figure-3,the partially learned decision tree resulting from the first step of ID3 is shown. The final decision tree learned by ID3 from the 14 training examples is shown in Figure-1.



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: The partially learned decision tree resulting from the first step of ID3.

5 Hypothesis Space Search in Decision Tree Learning

The hypothesis space searched by ID3 is the set of possible decision trees. ID3 performs a hill-climbing search through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic. This search is depicted in Figure-4.

Capabilities and limitations of ID3:

- Hypothesis space is complete!
  Target function is surely in there...
- Outputs a single hypothesis (which one?)
  Can't play 20 questions...

- No back tracking
  Local minima...
- Statistically-based search choices
  Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

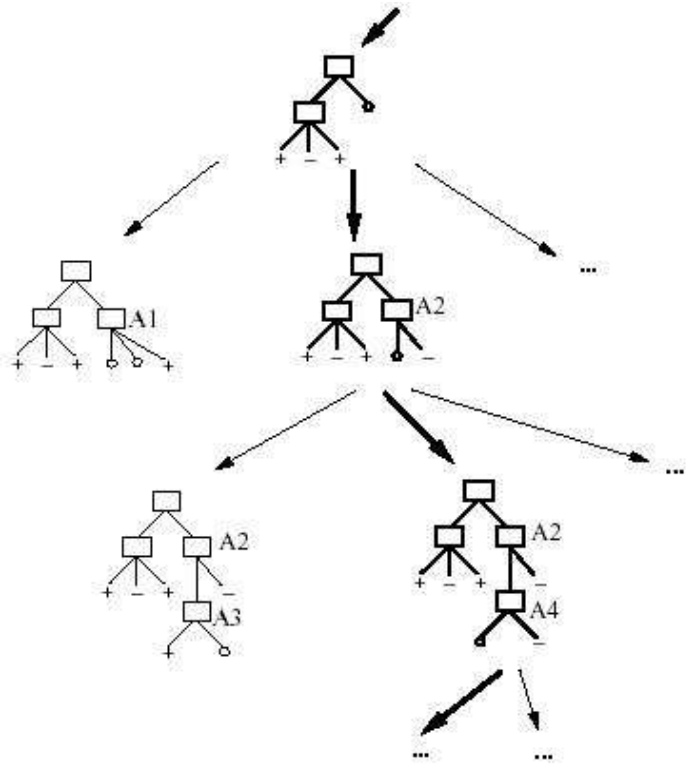


Figure 4: Hypothesis space search by ID3.

6 Inductive Bias in Decision Tree Learning

Note H is the power set of instances X.
→ Unbiased?
Not really...

- Preference for short trees over larger 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

6.1 Why Prefer Short Hypothesis?

Is ID3's inductive bias favoring shorter decision trees a sound basis for generalizing beyond the training data?

> **Occam's razor :** Prefer the simplest hypothesis that fits the data.
> Why prefer short hypotheses?

Arguments in favor:

- Fewer short hypotheses than long hypotheses
  - a short hypothesis that fits data is unlikely to be a coincidence
  - a long hypothesis that fits data might be a coincidence

Arguments opposed:

- There are many ways to define small sets of hypotheses
- 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?

7 Issues in Decision Tree Learning

Practical issues in learning decision trees include:

- Determining how deeply to grow the decision tree.
- Handling continues attributes.
- Choosing an appropriate attribute selection measure.
- Handling training data with missing attribute values.
- Handling attributes with differing costs.
- Improving computational efficiency.

7.1 Avoiding Overfitting the Data

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, *overfit* the training examples!!

Consider error of hypothesis h over

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

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

$$error_{train}(h) < error_{train}(h')$$
$$\text{and}$$
$$error_D(h) > error_D(h')$$

Consider adding a noisy training example #15 to the training examples we considered before:

Sunny, Hot, Normal, Strong, PlayTennis = No

What effect does it have on the earlier tree?

ID3 outputs a decision tree (h) that is more complex than the original tree from Figure-1(h'). h fits the training examples perfectly,whereas the simpler h' will not.

Figure-5 illustrates the impact of overfitting in a typical application of decision tree learning.

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))*

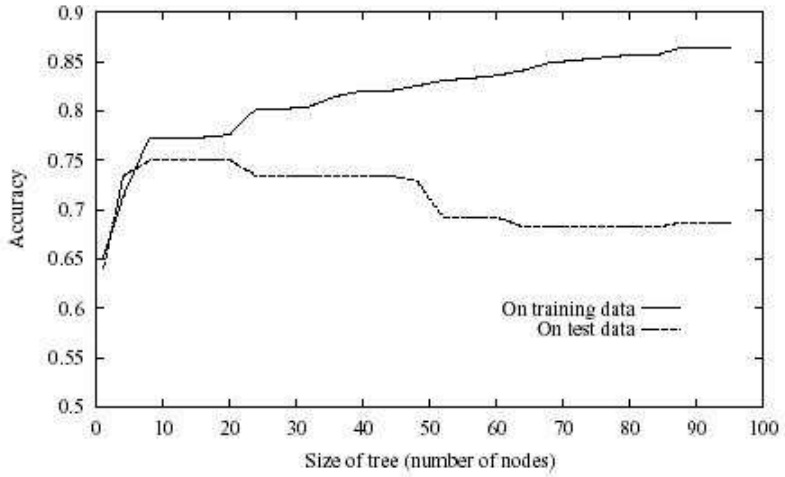


Figure 5: Overfitting in decision tree learning

7.1.1 Reduced Error Pruning

How exactly might we use a validation set to prevent overfitting?One approach is *reduced-error pruning*. Pruning a node consists of:

- removing the subtree rooted at that node,
- making it a leaf,
- and assigning it the most common classification of the training examples affiliated with that node.

Reduced-error pruning produces the smallest version of the most accurate subtree. The impact of reduced-error pruning on the accuracy of the decision tree is illustrated in Figure-6.

- *Training* set: learn the tree
- *Validation* set: prune the tree
- *Test* set: estimate future classification accuracy

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

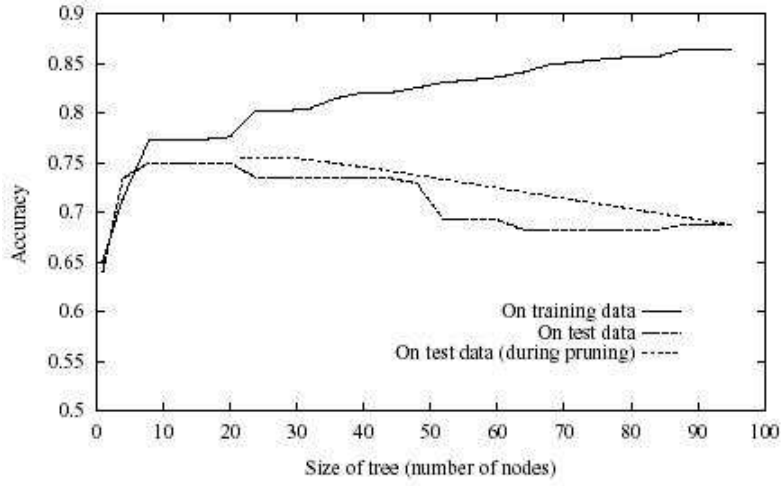


Figure 6: Effect of reduced-error pruning in decision tree learning.

7.1.2 Rule Post-Pruning

Rule-post pruning involves the following steps:

1. Infer the decision tree from the training set.
2. Convert the tree to equivalent set of rules.
3. Prune each rule independently of others.
4. Sort final rules into desired sequence for use.

In rule post-pruning, one rule is generated for each leaf node in the tree.For example, the two leftmost path of the tree in Figure-1 is translated into the rules:

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

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

Advantages of converting decision tree to rules before pruning:

- allows distinguishing among the different contexts in which a decision node is used
- removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves
- improves readibility

7.2 Incorporating 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

7.3 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

7.4 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 the same fashion.

7.5 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 minimum 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.

8 References

1. Mitchell, Tom.*Machine Learning:Decision Tree Learning* (Chapter 3), The MIT Press, 1997.
2. J.R.Quinlan. *Induction of Decision Trees*,August 1,1985
3. Online. *http://www.comp.hkbu.edu.hk/∼ymc/course/sci3790_0304/chapter4.pdf* (Lecture Notes of Decision Tree Learning-ID3 Algorithm)
4.Lecture slides for textbook *Machine Learning*,©Tom M. Mitchell, McGraw Hill,1997
5.Online.*A simplified ID3 implementation in C++ and Java: http://www.ida.his.se/ida/kurser/ai-symbolsystem/kursmaterial/archive/assignments/assignment3/id3.html*
6.Online.A complete, java implementation of the decision-tree algorithm: *http://www.cogs.susx.ac.uk/users/christ/crs/sai/lec15.html*
7.Online.*Machine Learning Algorithms in Java: www.aifb.uni-karlsruhe.de/Lehre/Winter2002-03/ kdd/download/weka/Tutorial.ps*
8.Online.*C4.5 tutorial and implementation in C: http://www2.cs.uregina.ca/ hamilton/courses/831/notes/ml/dtrees/c4.5/tutorial.html*