Learning

Chapter 18 and Parts of Chapter 20

- AI systems are complex and may have many parameters.
- It is impractical and often impossible to encode all the knowledge a system needs.
- Different types of data may require very different parameters.
- Instead of trying to hard code all the knowledge, it makes sense to learn it.

Learning from Observations

- Supervised Learning – learn a function from a set of training examples which are preclassified feature vectors.

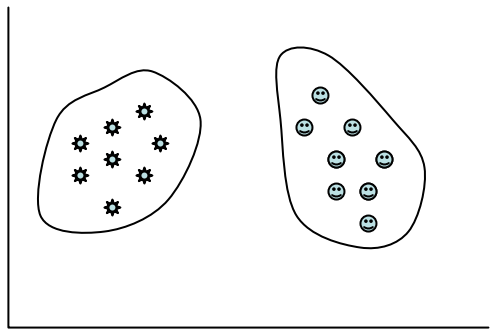
feature vector	class
(square, red)	I
(square, blue)	I
(circle, red)	II
(circle blue)	II
(triangle, red)	I
(triange, green)	I
(ellipse, blue)	II
(ellipse, red)	II

Given a previously unseen feature vector, what is the rule that tells us if it is in class I or class II?

(circle, green) ?
(triangle, blue) ?

Learning from Observations

- Unsupervised Learning – No classes are given. The idea is to find patterns in the data. This generally involves clustering.



- Reinforcement Learning – learn from feedback after a decision is made.

Topics to Cover

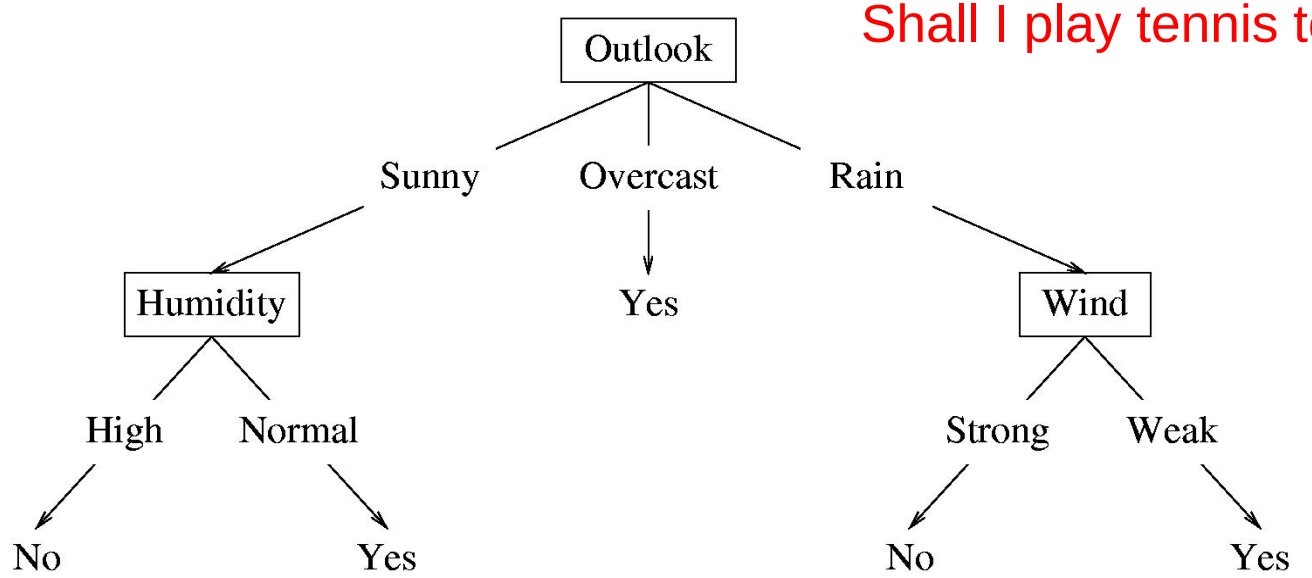
- Inductive Learning
  - decision trees
  - ensembles
  - Bayesian decision making
  - neural nets
  - kernel machines
- Unsupervised Learning
  - Expectation Maximumization (EM) algorithm

Decision Trees

- Theory is well-understood.
- Often used in pattern recognition problems.
- Has the nice property that you can easily understand the decision rule it has learned.

Decision Tree Hypothesis Space

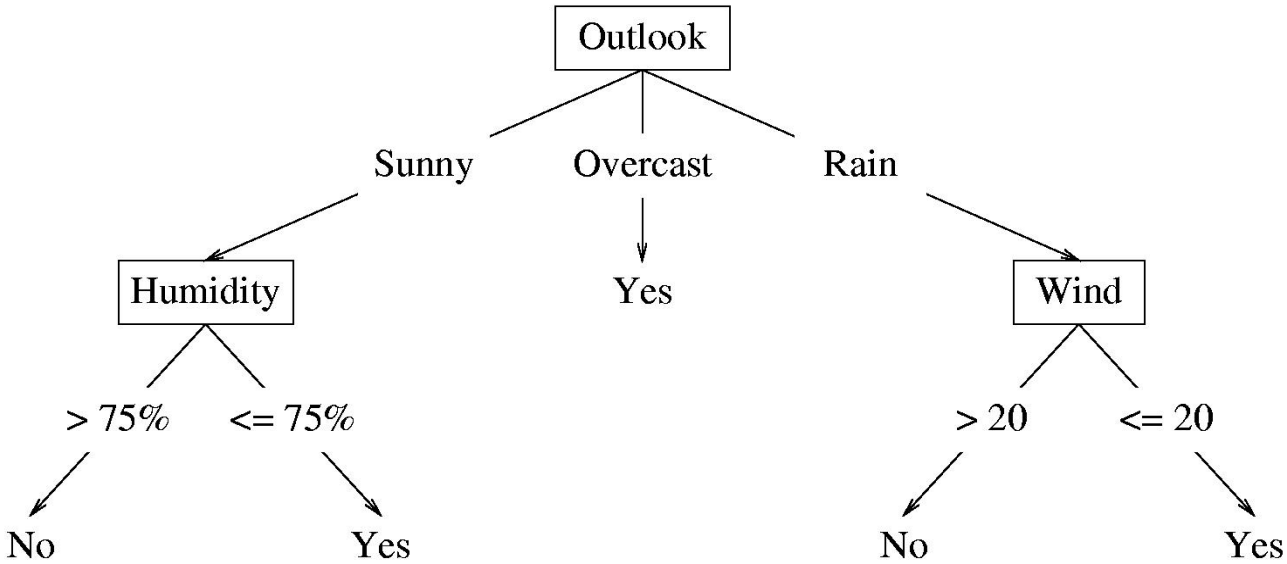
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

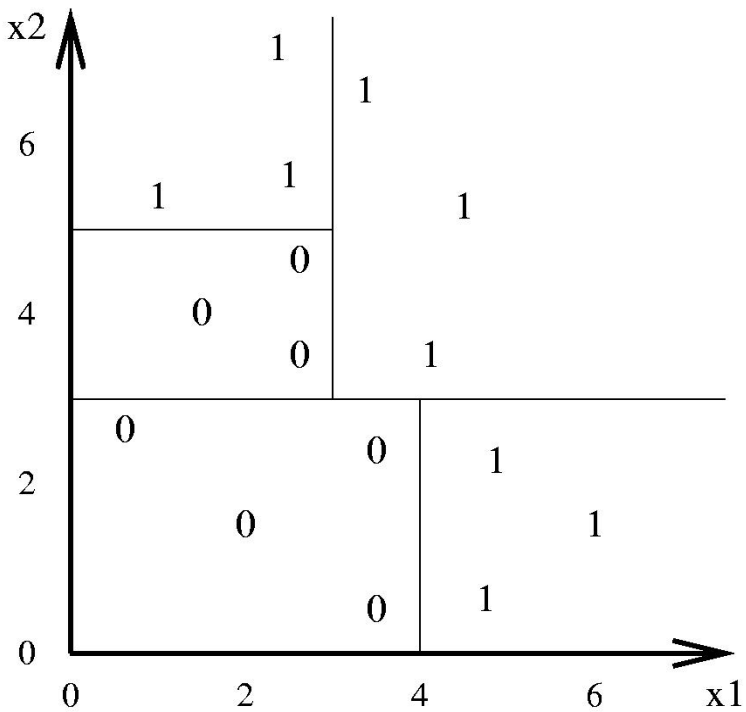
Decision Tree Hypothesis Space

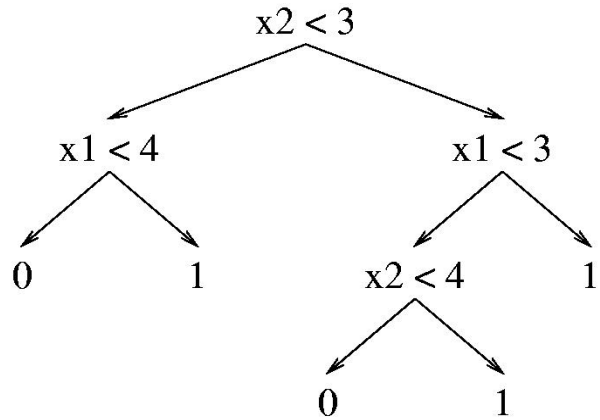
If the features are continuous, internal nodes may test the value of a feature against a threshold.



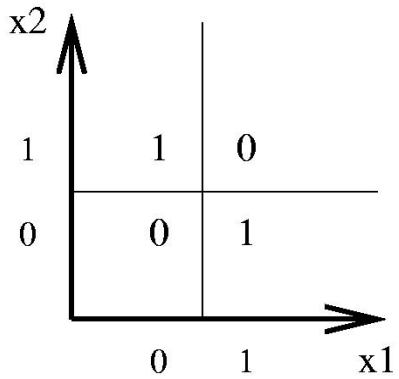
Decision Tree Decision Boundaries

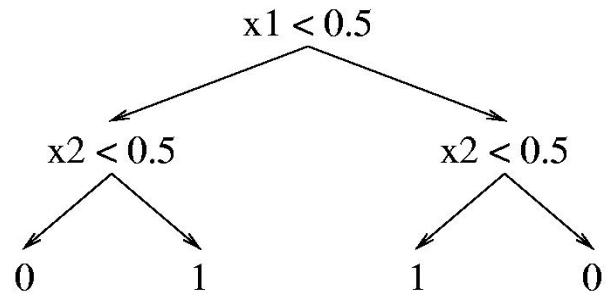
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees Can Represent Any Boolean Function





The tree will in the worst case require exponentially many nodes, however.

Learning Decision Trees

Decision trees provide a very popular and efficient hypothesis space.

- **Variable Size**. Any boolean function can be represented.
- **Deterministic**.
- **Discrete and Continuous Parameters**.

Learning Algorithm for Decision Trees

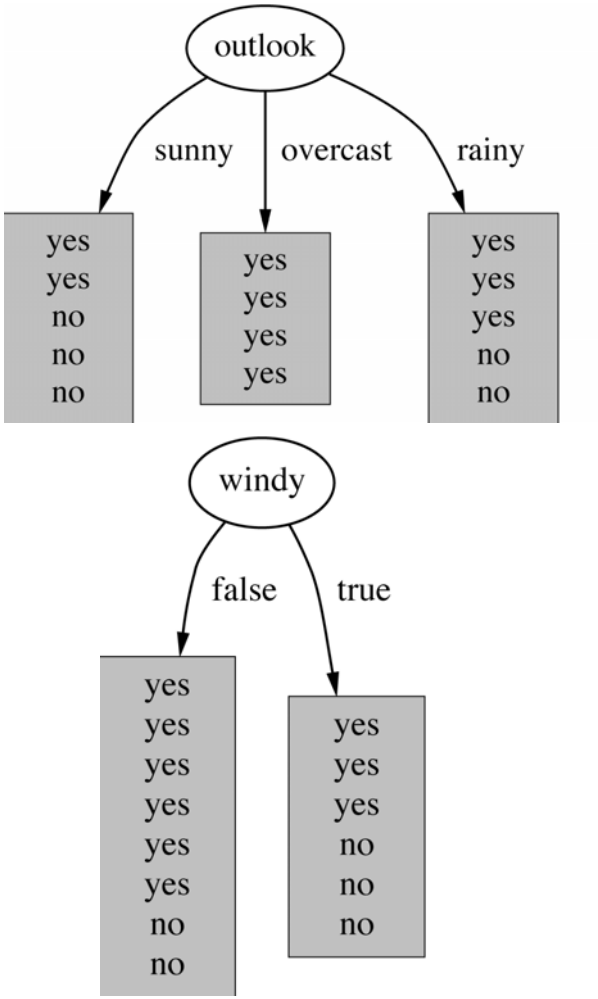
The same basic learning algorithm has been discovered by many people independently:

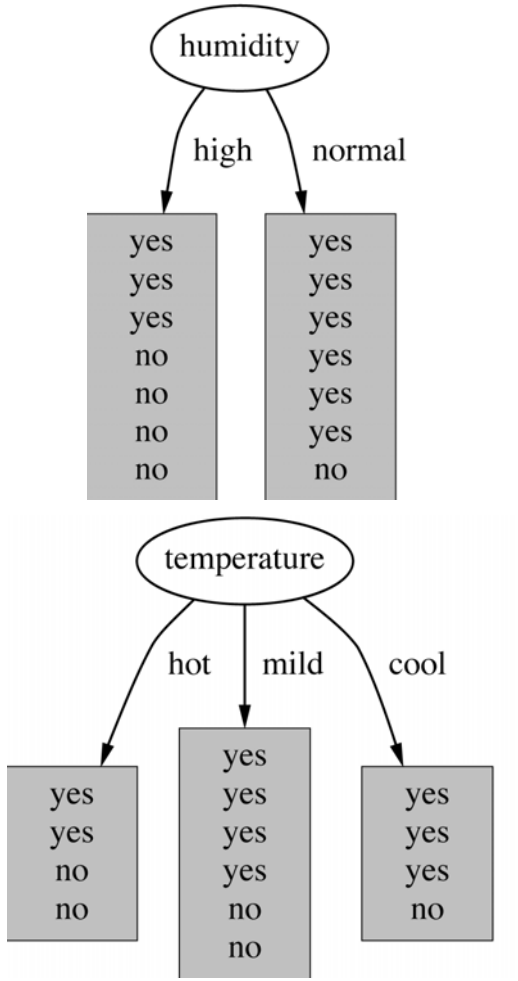
GrowTree(S)
if ($y = 0$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(0)
else if ($y = 1$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(1)
else
    choose best attribute x_j
    $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
    $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
    return new node(x_j, GrowTree(S_0), GrowTree(S_1))

How do we choose the best attribute?

What should that attribute do for us?

Which attribute to select?





witten&eibe

Criterion for attribute selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes
- Need a good measure of purity!
  - Maximal when?
  - Minimal when?

Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data.

ChooseBestAttribute(S)

choose j to minimize J_j, computed as follows:

$S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;

$S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;

$y_0 =$ the most common value of y in S_0

$y_1 =$ the most common value of y in S_1

$J_0 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$

$J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$

$J_j = J_0 + J_1$ (total errors if we split on this feature)

return j

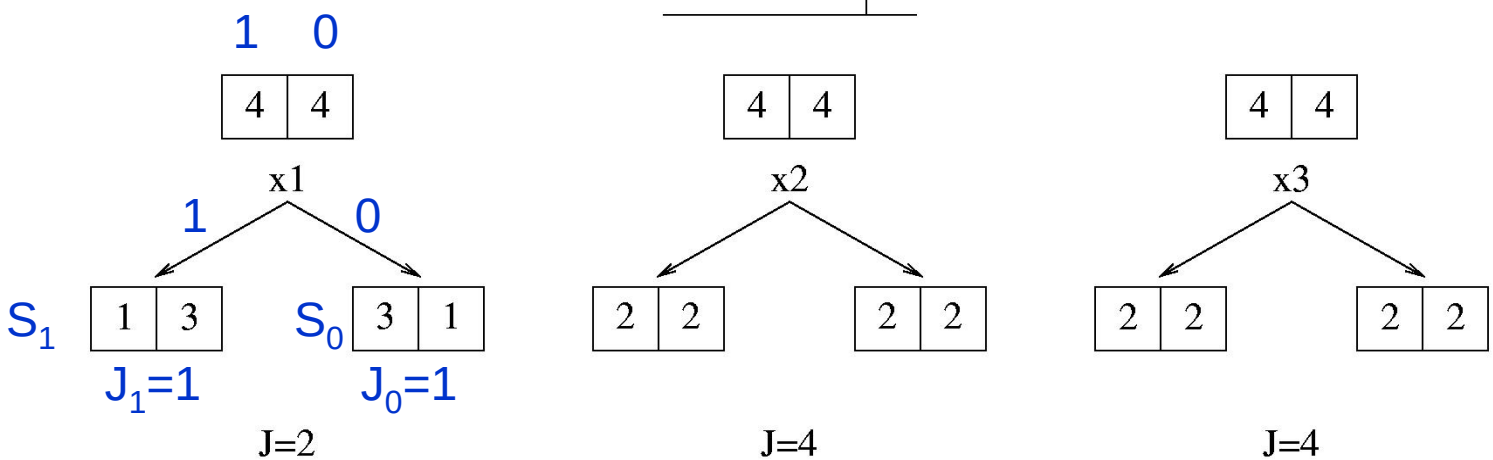
Choosing the Best Attribute—An Example

Original Training Set S

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

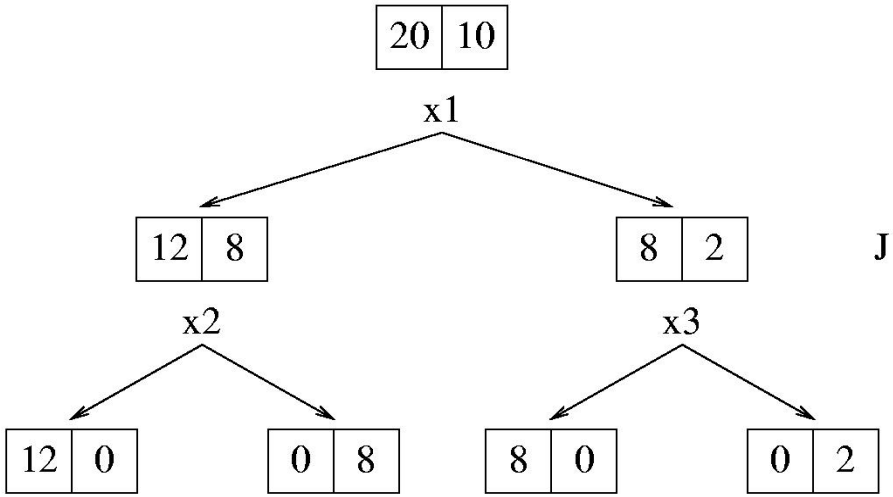
S_1
y_1 =

S_2
y_2 =



Choosing the Best Attribute (3)

Unfortunately, this measure does not always work well, because it does not detect cases where we are making "progress" toward a good tree.



J = 10

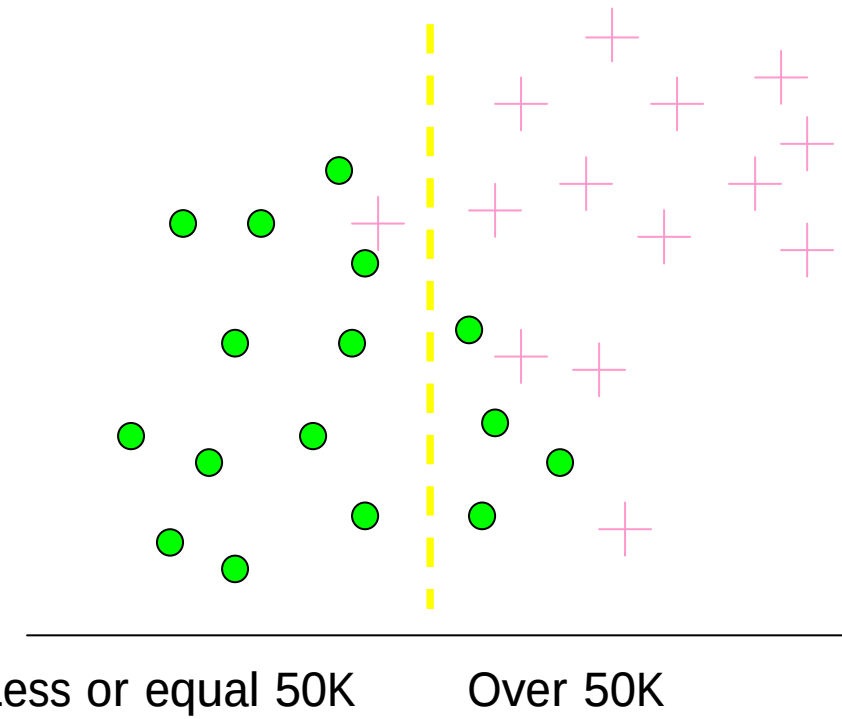
high error here

But perfect splits at the next level down.

Information Gain

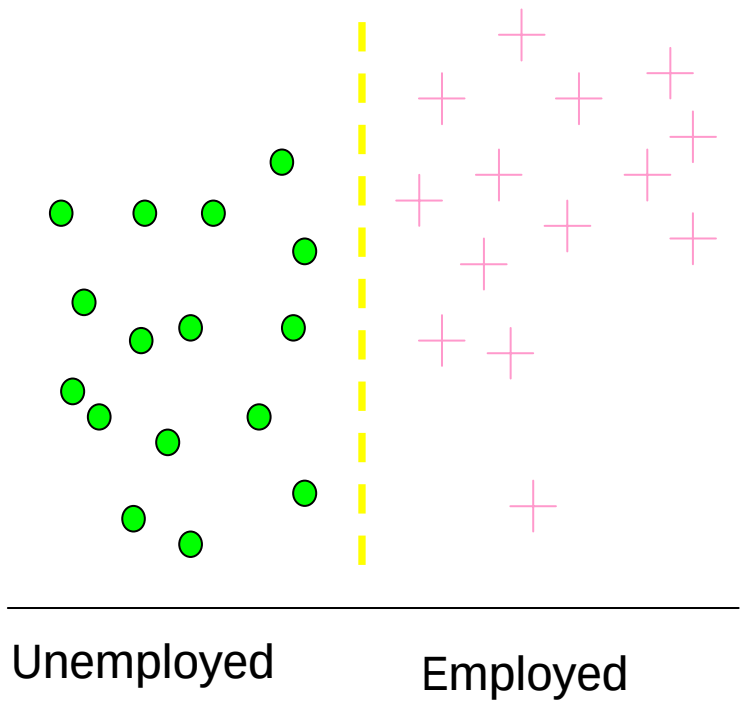
Which test is more informative?

Split over whether Balance exceeds 50K



Less or equal 50K Over 50K

Split over whether applicant is employed

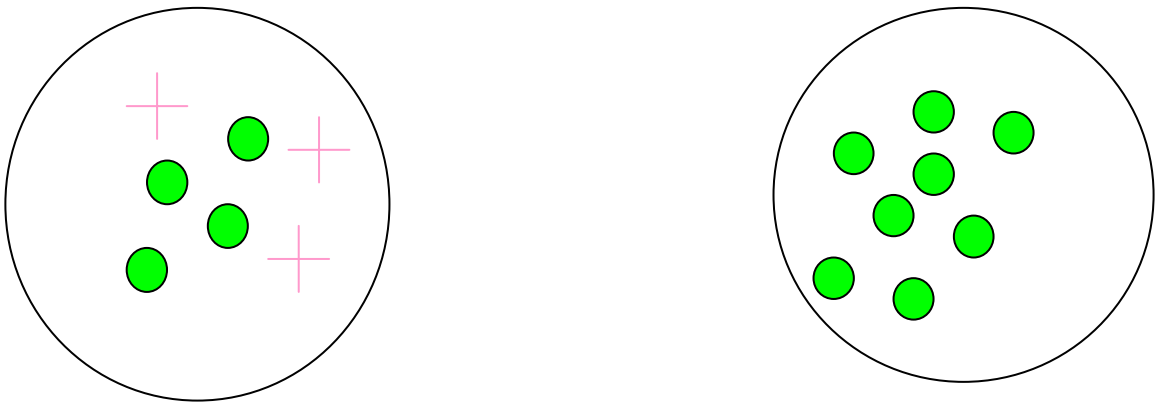


Unemployed Employed

Information Gain

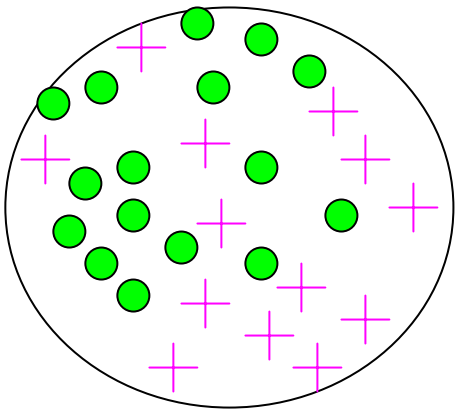
Impurity/Entropy (informal)

– Measures the level of **impurity** in a group of examples

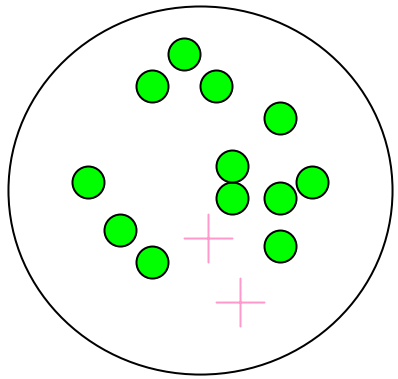


Impurity

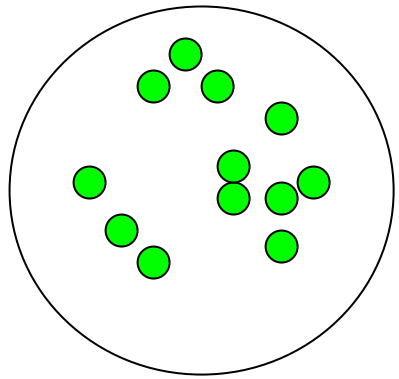
Very impure group



Less impure

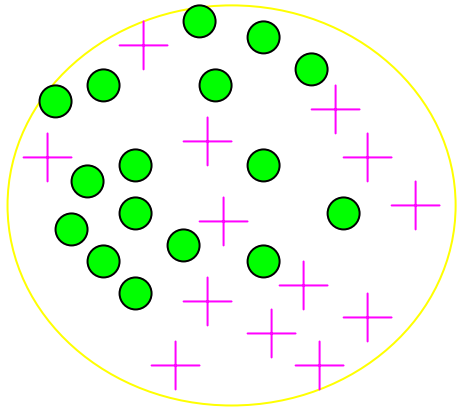


Minimum impurity



Entropy: a common way to measure impurity

- Entropy = $\sum_i - p_i \log_2 p_i$



p_i is the probability of class i

Compute it as the proportion of class i in the set.

- Entropy comes from information theory. The higher the entropy the more the information content.

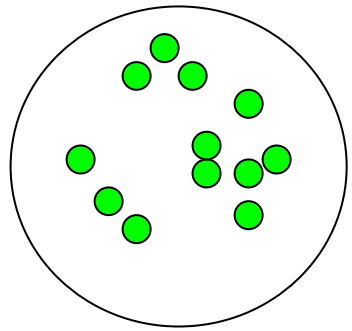
What does that mean for learning from examples?

2-Class Cases:

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = $- 1 \log_2 1 = 0$

  not a good training set for learning

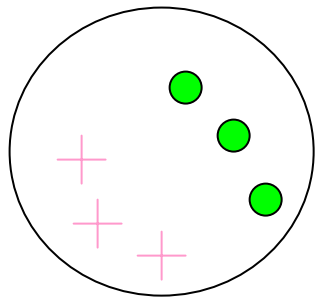
Minimum impurity



- What is the entropy of a group with 50% in either class?
  - entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

  good training set for learning

Maximum impurity



Information Gain

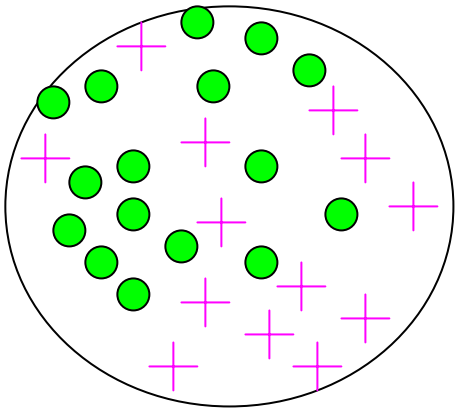
- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

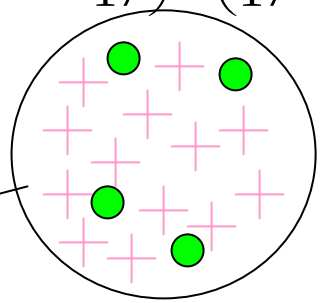
Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

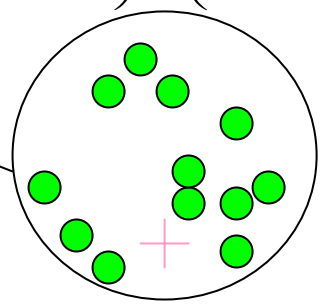
Entire population (30 instances)





17 instances

child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Entropy-Based Automatic Decision Tree Construction

Training Set S

x1=(f11,f12,…f1m)

x2=(f21,f22, f2m)

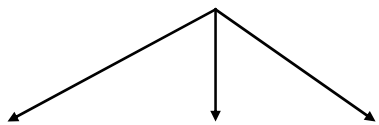
.

.

xn=(fn1,f22, f2m)

Node 1

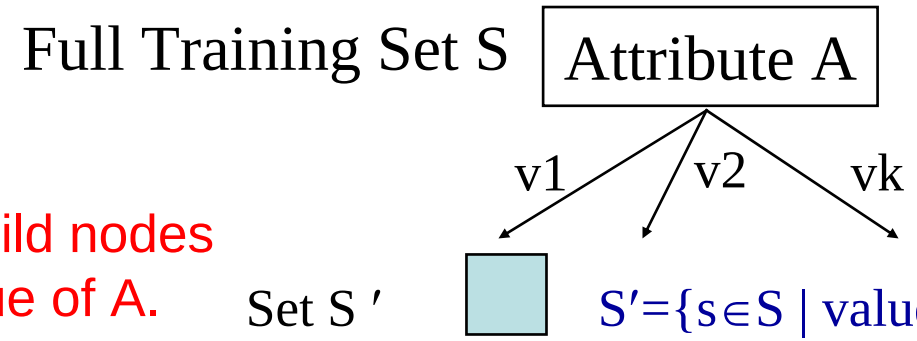
What feature should be used?



What values?

Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.

Using Information Gain to Construct a Decision Tree



Choose the attribute A with highest information gain for the full training set at the root of the tree.

Construct child nodes for each value of A. Each has an associated subset of vectors in which A has a particular value.

repeat recursively till when?

Information gain has the disadvantage that it prefers attributes with large number of values that split the data into small, pure subsets. Quinlan's gain ratio did some normalization to improve this.

Information Content

The information content I(C;F) of the class variable C with possible values {c1, c2, … cm} with respect to the feature variable F with possible values {f1, f2, … , fd} is defined by:

$$I(C;F) = \sum_{i=1}^{m} \sum_{j=1}^{d} P(C = c_i, F = f_j) log_2 \frac{P(C = c_i, F = f_j)}{P(C = c_i)P(F = f_j)}$$

- $P(C = c_i)$ is the probability of class C having value c_i.
- $P(F=f_j)$ is the probability of feature F having value f_j.
- $P(C=c_i,F=f_j)$ is the joint probability of class $C = c_i$ and variable $F = f_j$.

These are estimated from frequencies in the training data.

Simple Example

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

How would you distinguish class I from class II?

Example (cont)

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

$$
\begin{aligned}
I(C,X) &= P(C=I,X=1)log_2\frac{P(C=I,X=1)}{P(C=I)P(X=1)} \\
&+ P(C=I,X=0)log_2\frac{P(C=I,X=0)}{P(C=I)P(X=0)} \\
&+ P(C=II,X=1)log_2\frac{P(C=II,X=1)}{P(C=II)P(X=1)} \\
&+ P(C=II,X=0)log_2\frac{P(C=II,X=0)}{P(C=II)P(X=0)} \\
&= .5log_2\frac{.5}{.5\times.75} + 0 + .25log_2\frac{.25}{.5\times.25} + .25log_2\frac{.25}{.5\times.75} \\
&= 0.311
\end{aligned}
$$

$$
\begin{aligned}
I(C,Y) &= .5log_2\frac{.5}{.5\times.5} + 0 + .5log_2\frac{.5}{.5\times.5} + 0 \\
&= 1.0
\end{aligned}
$$

$$
\begin{aligned}
I(C,Z) &= .25log_2\frac{.25}{.5\times.5} + .25log_2\frac{.25}{.5\times.5} + .25log_2\frac{.25}{.5\times.5} + .25log_2\frac{.25}{.5\times.5} \\
&= 0.0
\end{aligned}
$$

Which attribute is best? Which is worst? Does it make sense?

Using Information Content

- Start with the root of the decision tree and the whole training set.
- Compute I(C,F) for each feature F.
- Choose the feature F with highest information content for the root node.
- Create branches for each value f of F.
- On each branch, create a new node with reduced training set and repeat recursively.

Non-Boolean Features

- **Features with multiple discrete values**

  Construct a multiway split?

  Test for one value versus all of the others?

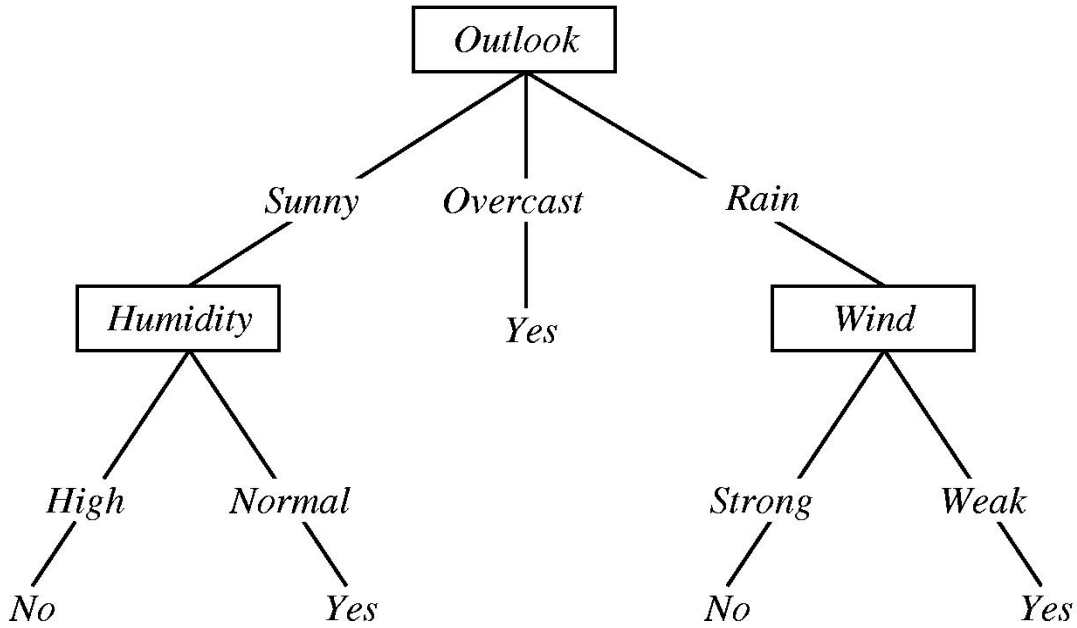
  Group the values into two disjoint subsets?

- **Real-valued features**

  Consider a threshold split using each observed value of the feature.

Whichever method is used, the mutual information can be computed to choose the best split.

Overfitting in Decision Trees



Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Overfitting

Consider error of hypothesis h over

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

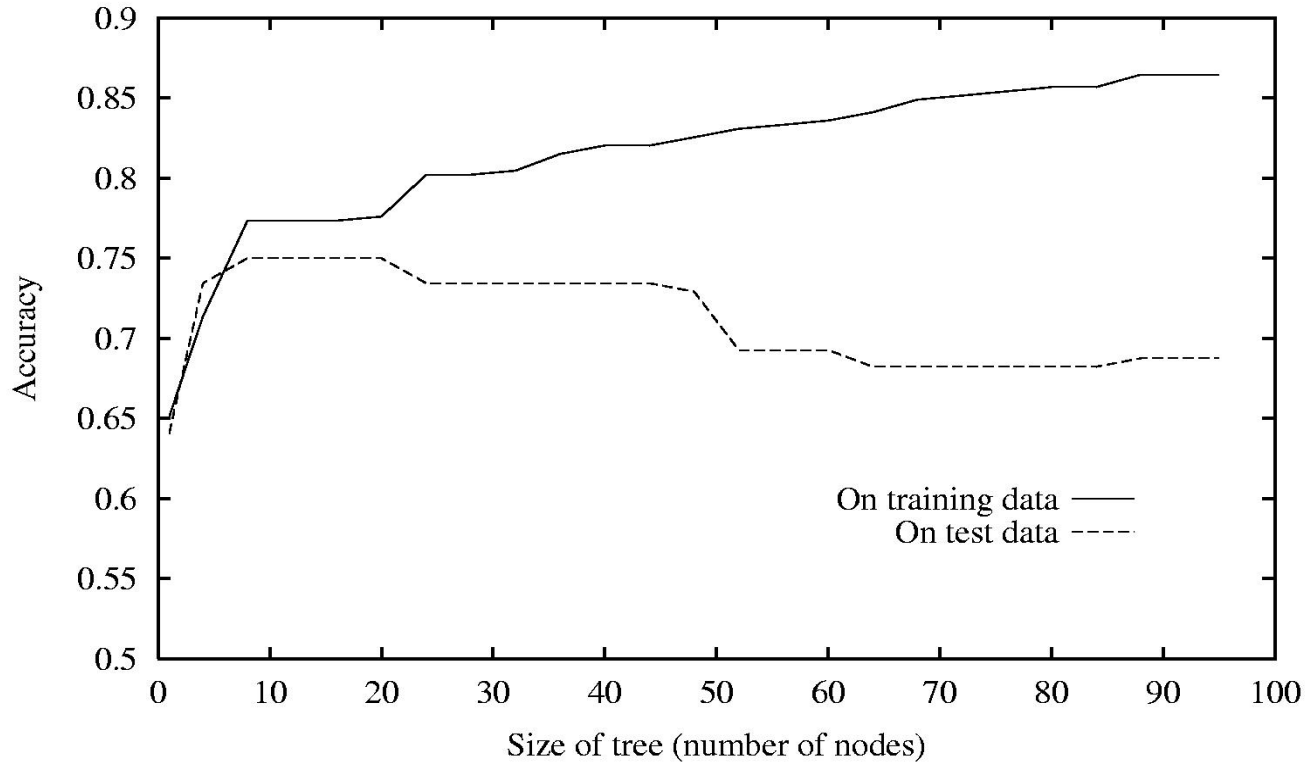
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

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

and

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

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
- Add complexity penalty to performance measure

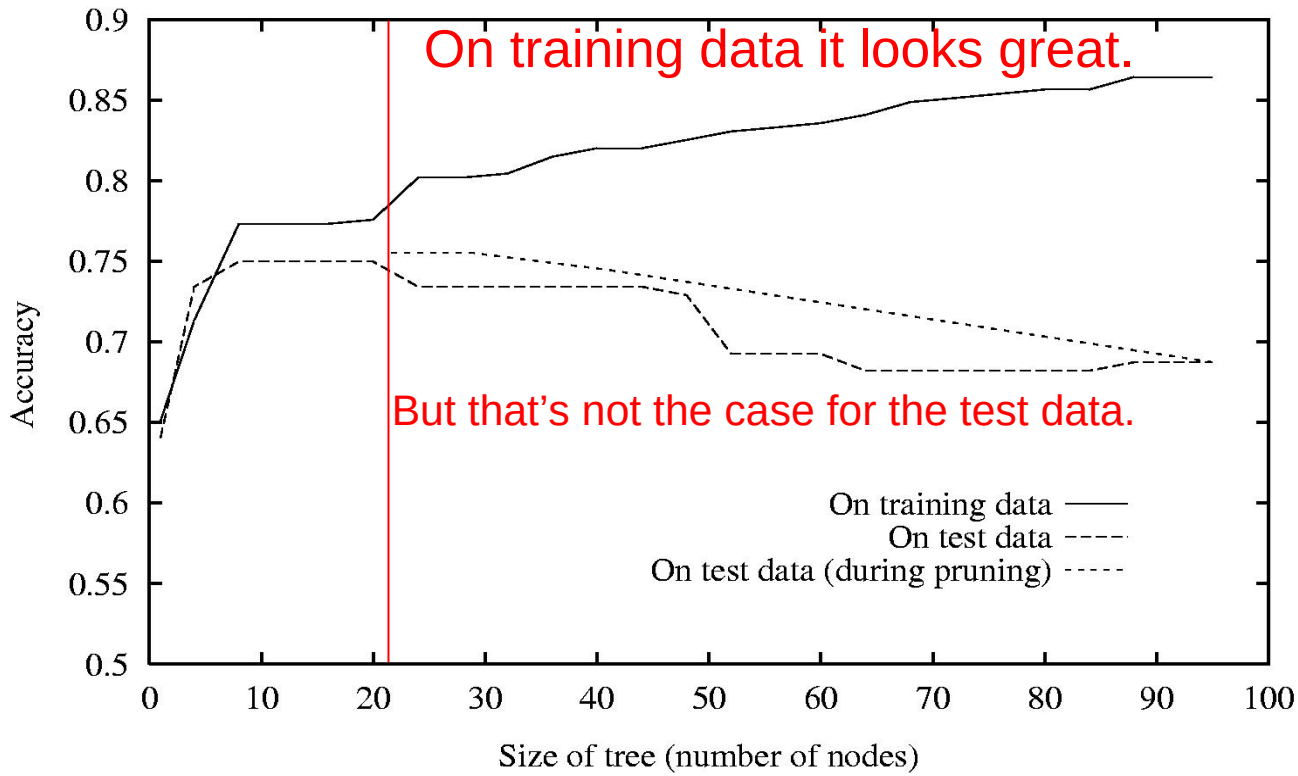
Reduced-Error Pruning

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

Effect of Reduced-Error Pruning



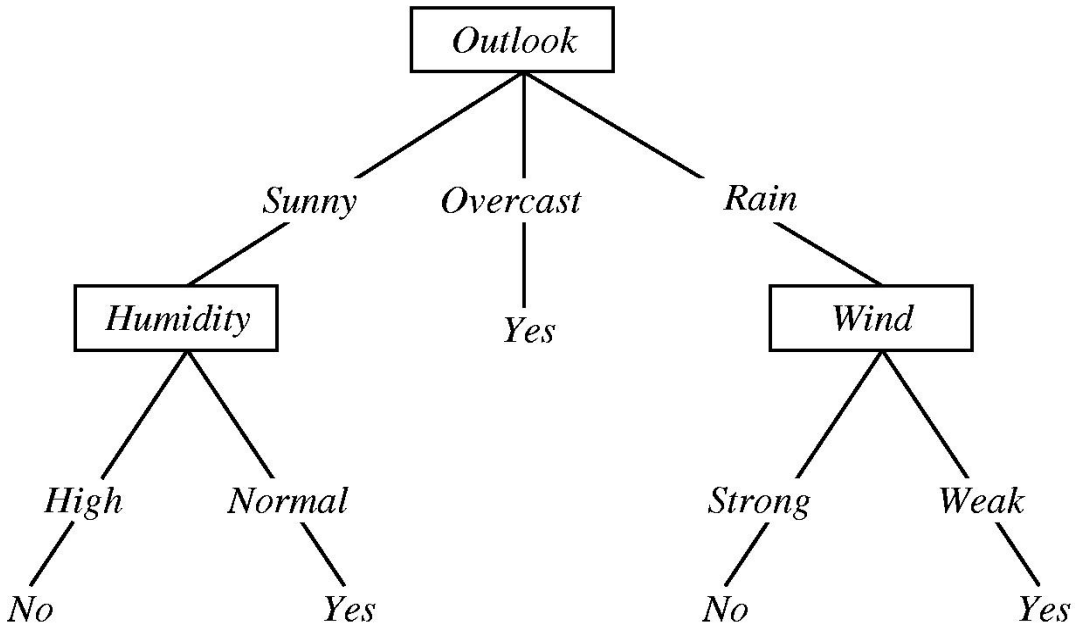
The tree is pruned back to the red line where it gives more accurate results on the test data.

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 $(Outlook = Sunny)\ AND\ (Humidity = High)$

THEN $PlayTennis = No$

IF $(Outlook = Sunny)\ AND\ (Humidity = Normal)$

THEN $PlayTennis = Yes$

...

Scaling Up

- ID3, C4.5, etc. assume data fits in main memory
  (OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data
  (OK for up to millions of examples)
- VFDT: at most one sequential scan
  (OK for up to billions of examples)

Decision Trees: Summary

- Representation=decision trees
- Bias=preference for small decision trees
- Search algorithm=
- Heuristic function=information gain or information content or others
- Overfitting and pruning
- Advantage is simplicity and easy conversion to rules.