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.

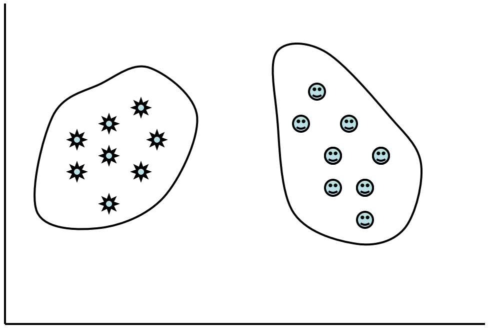
feature vector (shape,color)	class
(square, red)	I
(square, blue)	I
(circle, red)	II
(circle blue)	II
(triangle, red)	I
(triangle, green)	I
(ellipse, blue)	II
(ellipse, red)	II

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
  - K-Means Clustering
  - Expectation Maximization (EM) algorithm

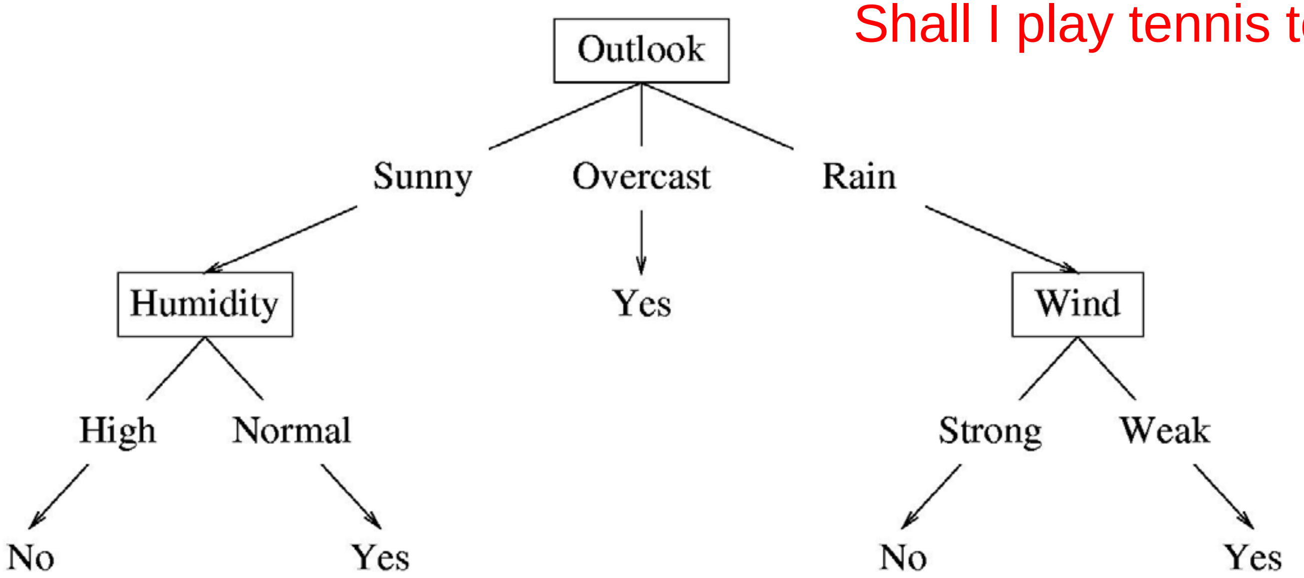
Decision Trees

- 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 $h(\mathbf{x})$.

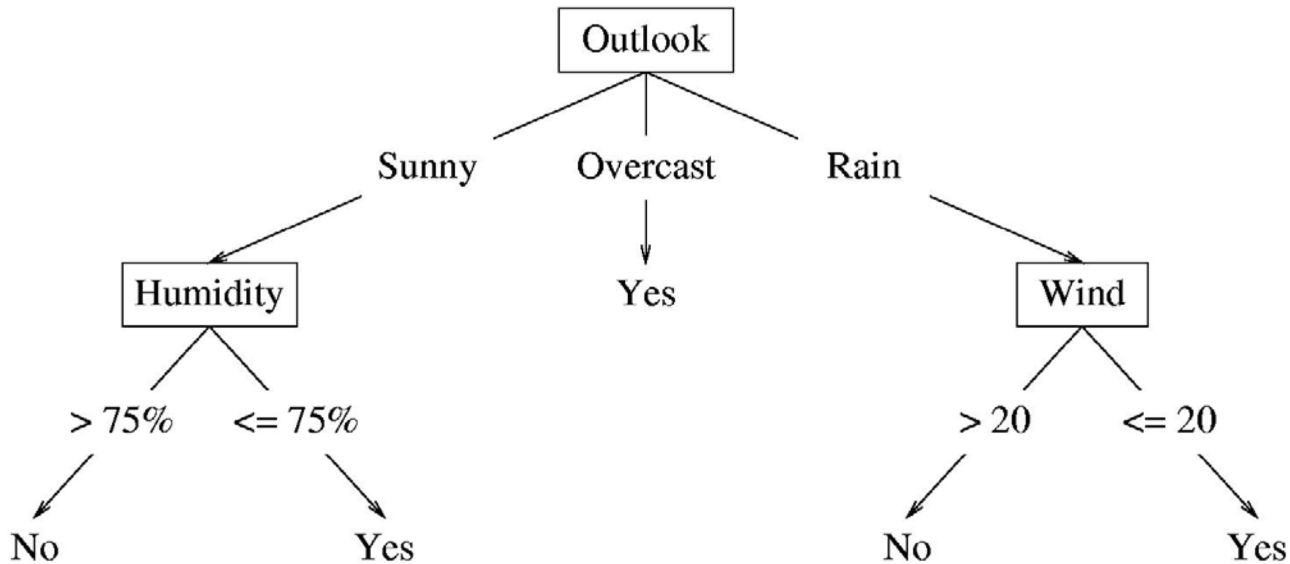
Shall I play tennis today?



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.

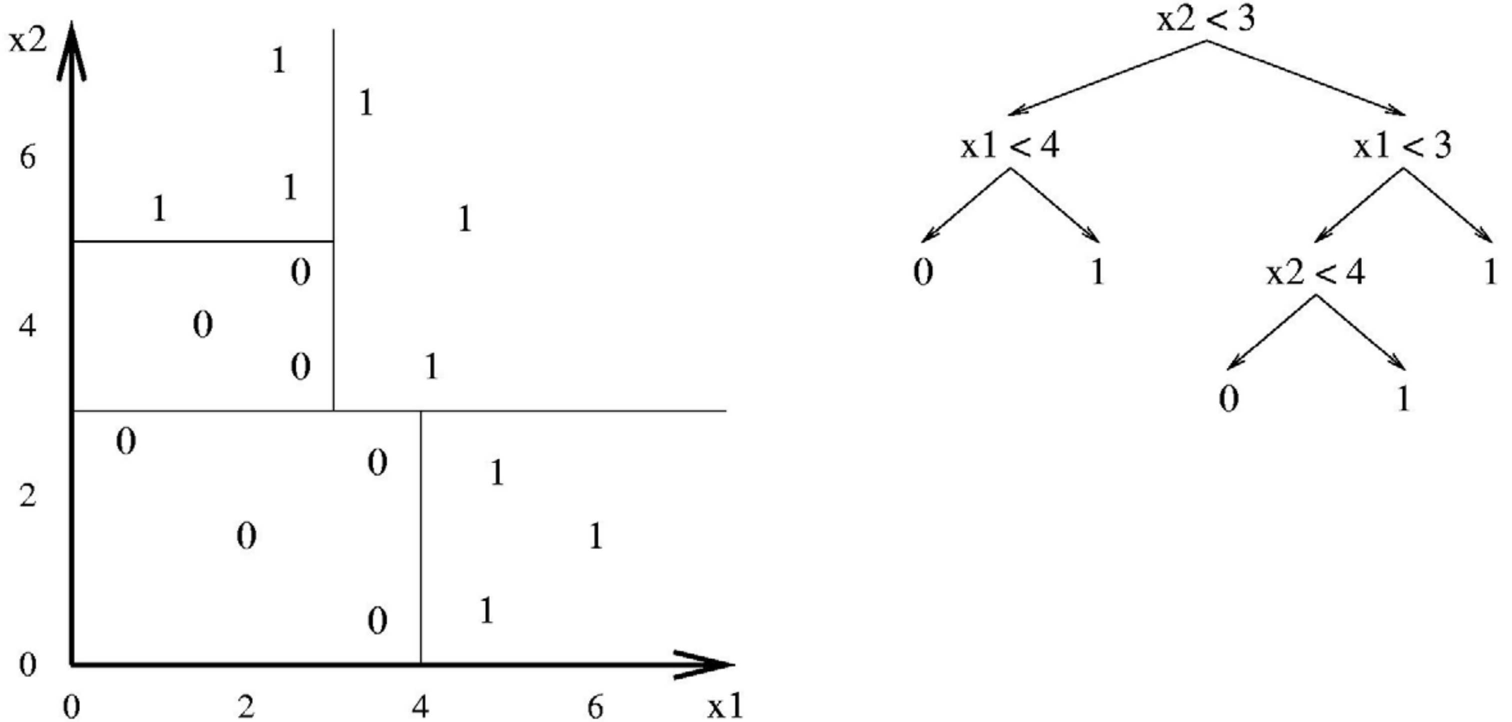
Decision Tree Hypothesis Space

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

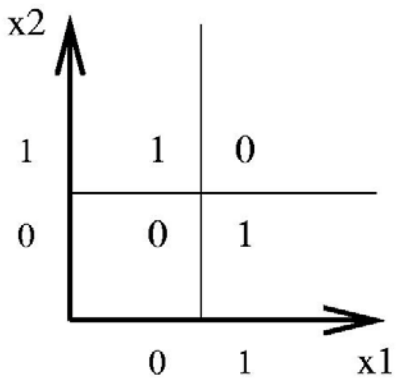


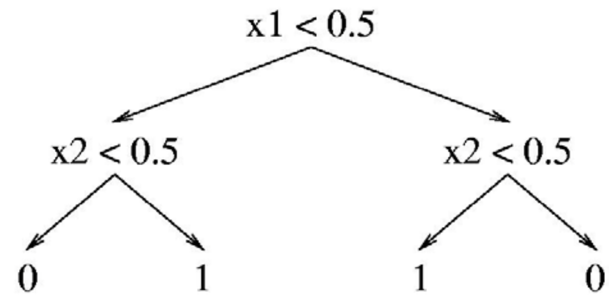
Decision Tree Decision Boundaries

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:

$\text{GROWTREE}(S)$

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 $x_j = 0$;

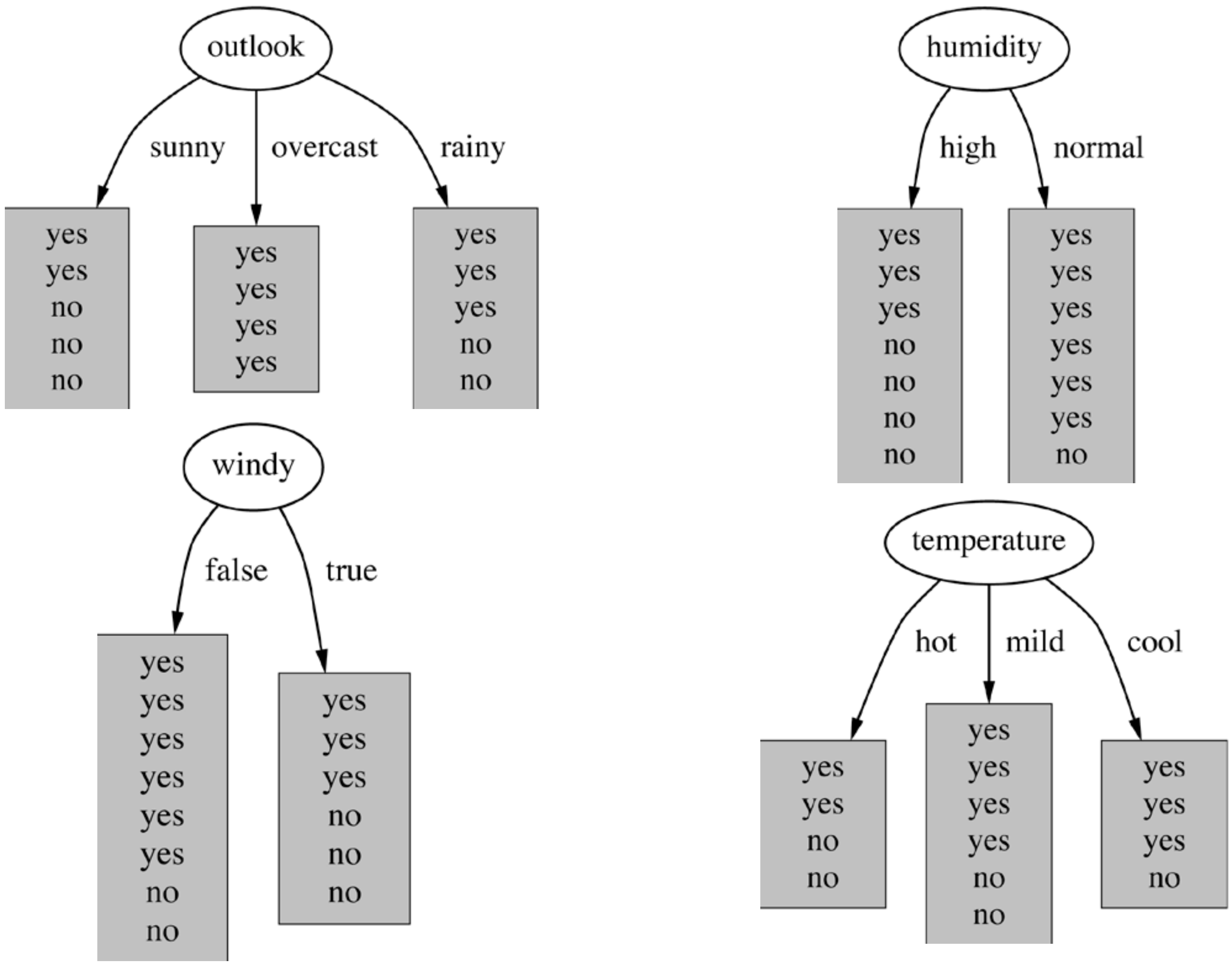
$S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;

return new node$(x_j, \text{GROWTREE}(S_0), \text{GROWTREE}(S_1))$

How do we choose the best attribute?

What should that attribute do for us?

Which attribute to select?



witten&eibe

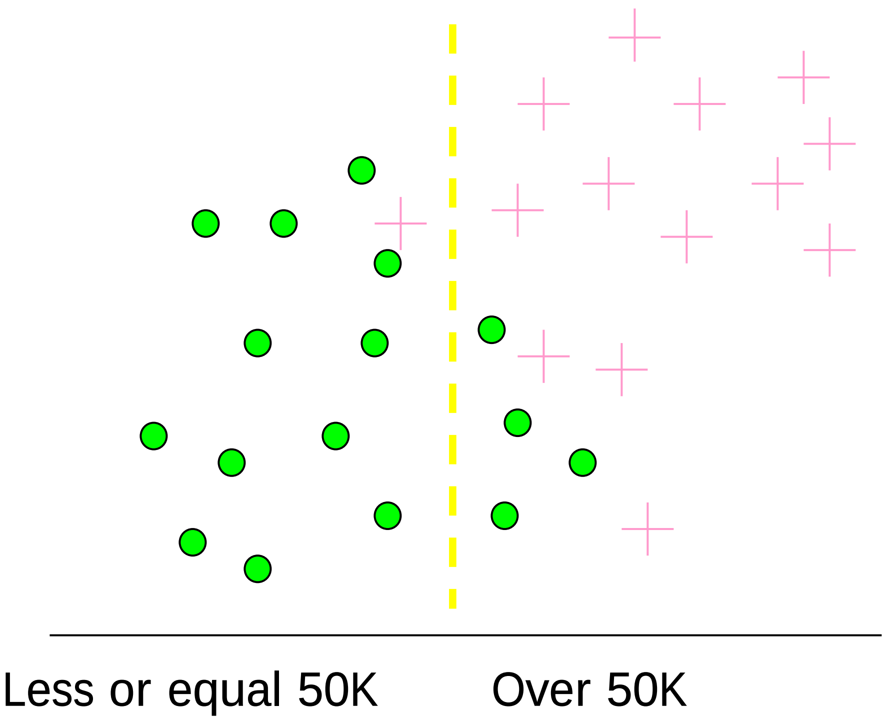
Criterion for attribute selection

- 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?

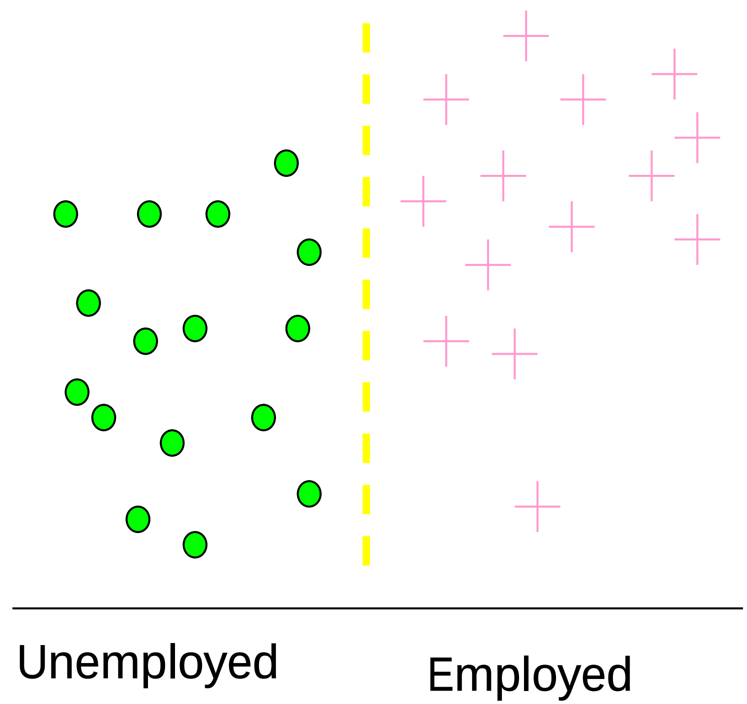
Information Gain

Which test is more informative?

Split over whether Balance exceeds 50K

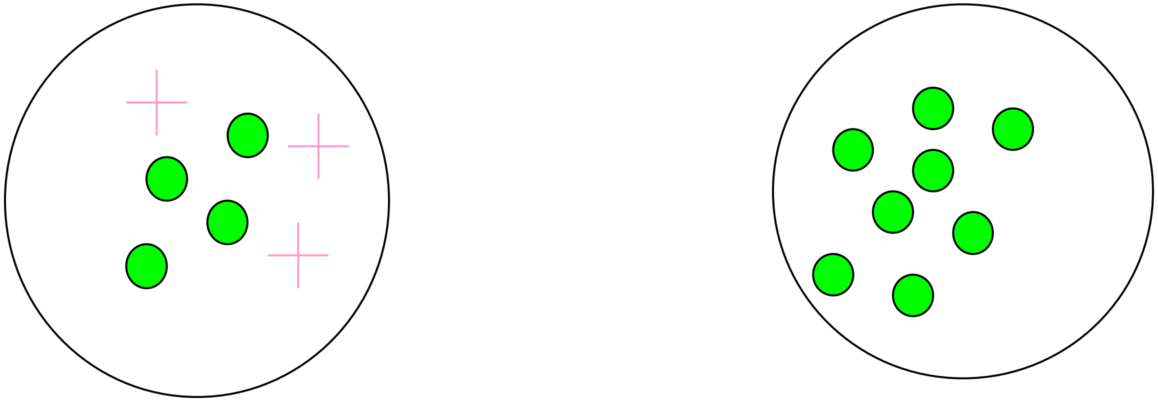


Split over whether applicant is employed



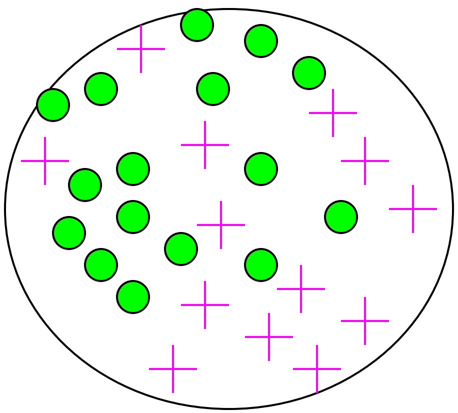
Impurity/Entropy (informal)

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

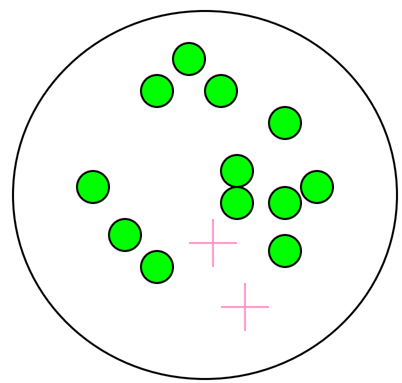


Impurity

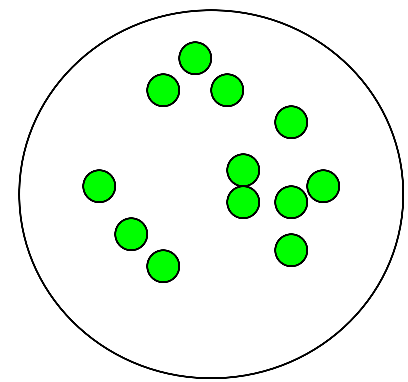
Very impure group



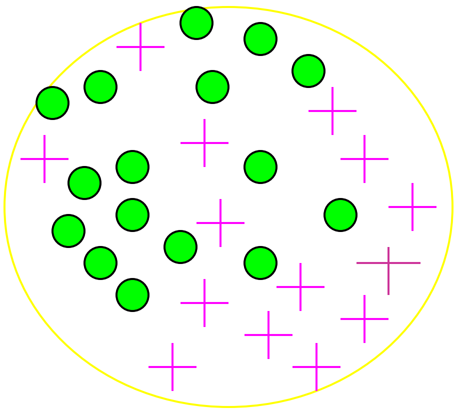
Less impure



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.

  16/30 are green circles; 14/30 are pink crosses
  $\log_2(16/30) = -.9$; $\log_2(14/30) = -1.1$
  Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

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

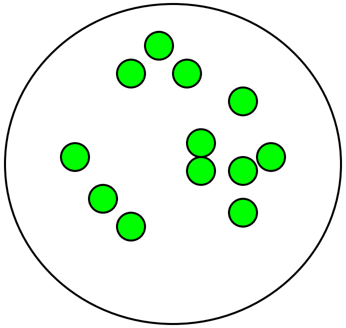
  What does that mean for learning from examples?

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

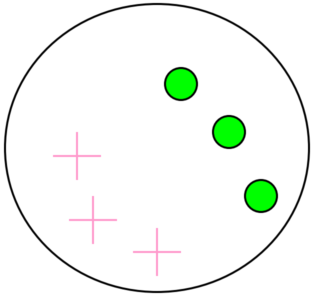
Minimum impurity



- 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



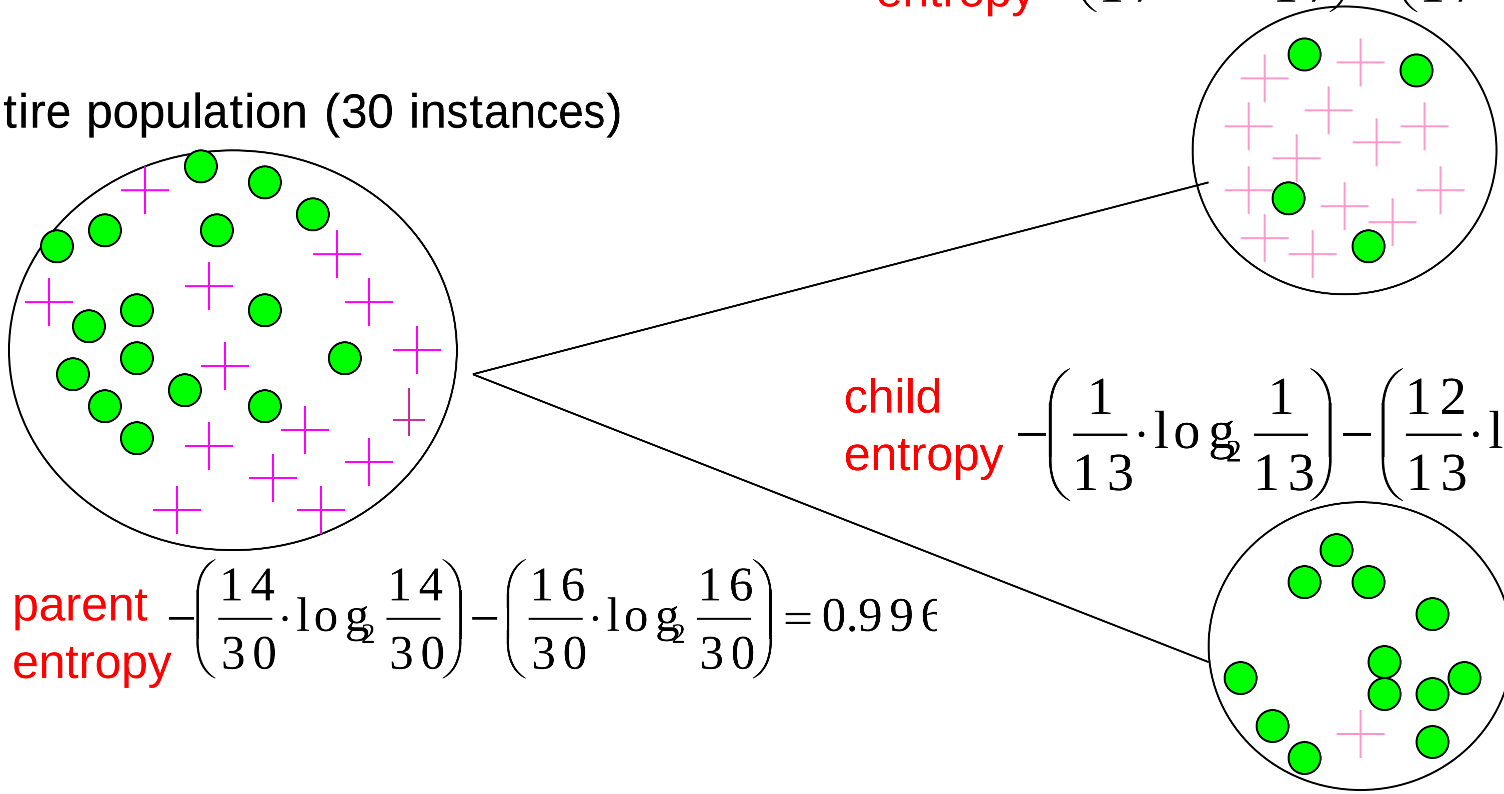
Information Gain

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



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

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 for this split

Entropy-Based Automatic Decision Tree Construction

Training Set S

$x_1=(f_{11},f_{12},\ldots f_{1m})$

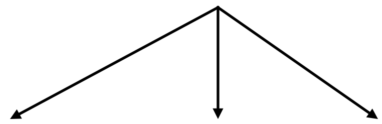
$x_2=(f_{21},f_{22}, \quad f_{2m})$

.

.

$x_n=(f_{n1},f_{22}, \quad f_{2m})$

Node 1
What feature
should be used?



What values?

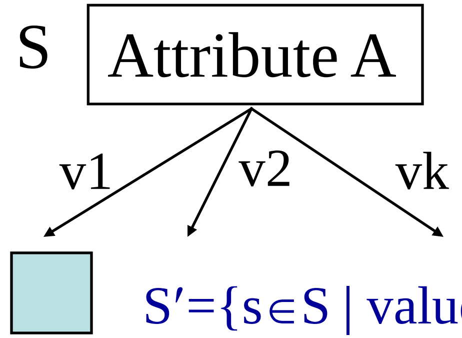
Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.

Using Information Gain to Construct a Decision Tree

1

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

Full Training Set S



2

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

Set S ′

$S' = \{ s \in S \mid value(A) = v1 \}$

3

repeat recursively till when?

Simple Example

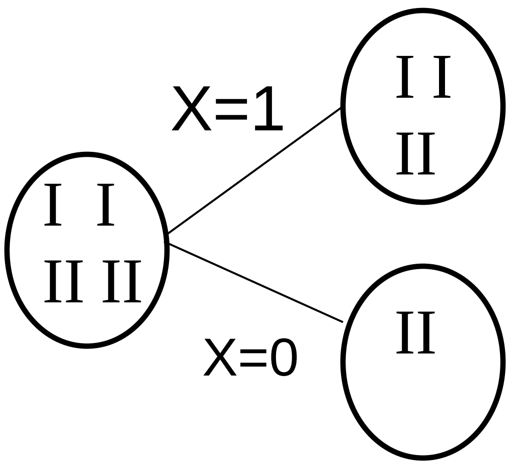
Training Set: 3 features and 2 classes

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?

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

Split on attribute X



If X is the best attribute,
this node would be further split.

$$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$$
$$= .5284 + .39$$
$$= .9184$$

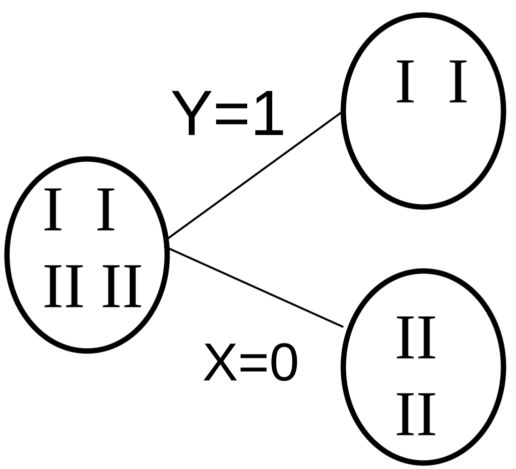
$$E_{child2} = 0$$

$E_{parent} = 1$

GAIN = 1 – (3/4)(.9184) – (1/4)(0) = .3112

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

Split on attribute Y



$E_{child1} = 0$

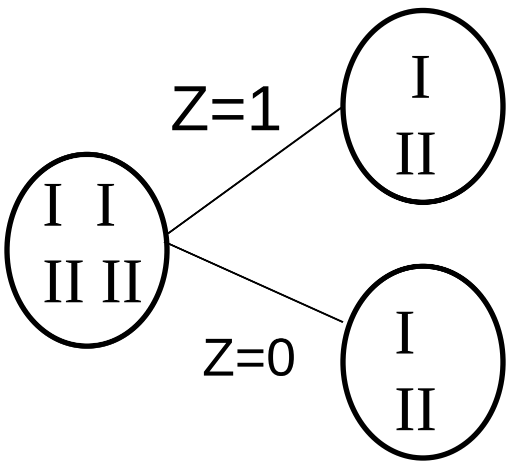
$E_{child2} = 0$

$E_{parent} = 1$

GAIN = 1 –(1/2) 0 – (1/2)0 = 1; BEST ONE

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

Split on attribute Z



$E_{child1} = 1$

$E_{child2} = 1$

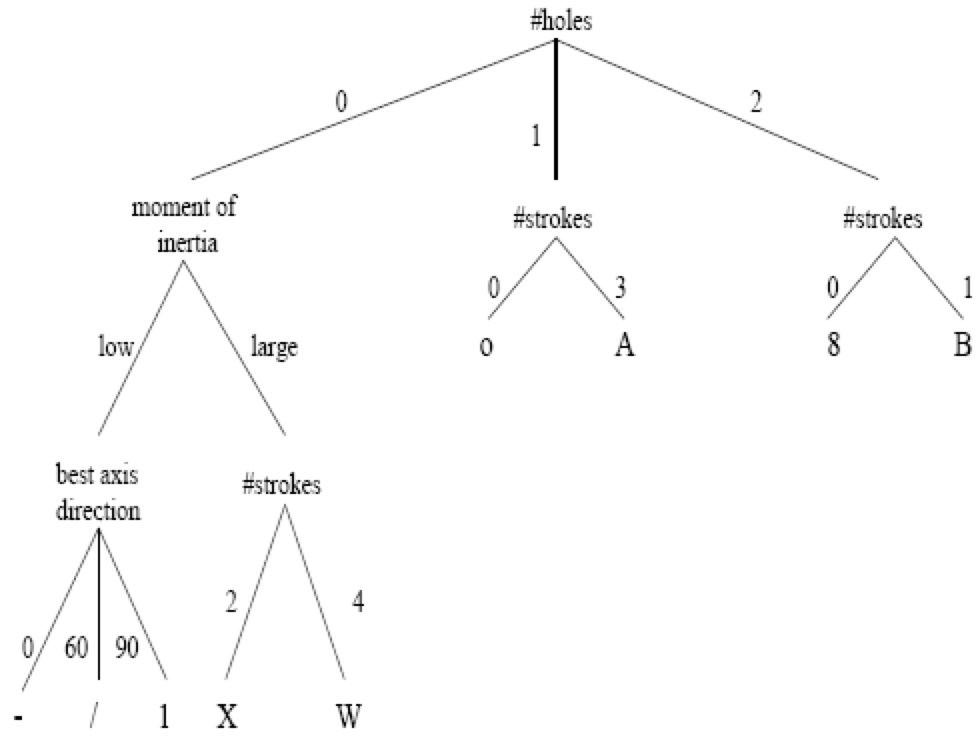
$E_{parent} = 1$

GAIN = 1 – (1/2)(1) – (1/2)(1) = 0 ie. NO GAIN; WORST

(class) character	area	height	width	number #holes	number #strokes	(cx,cy) center	best axis	least inertia
'A'	medium	high	3/4	1	3	1/2,2/3	90	medium
'B'	medium	high	3/4	2	1	1/3,1/2	90	large
'8'	medium	high	2/3	2	0	1/2,1/2	90	medium
'0'	medium	high	2/3	1	0	1/2,1/2	90	large
'1'	low	high	1/4	0	1	1/2,1/2	90	low
'W'	high	high	1	0	4	1/2,2/3	90	large
'X'	high	high	3/4	0	2	1/2,1/2	?	large
'*'	medium	low	1/2	0	0	1/2,1/2	?	large
'-'	low	low	2/3	0	1	1/2,1/2	0	low
'/'	low	high	2/3	0	1	1/2,1/2	60	low

Portion of a training set for character recognition

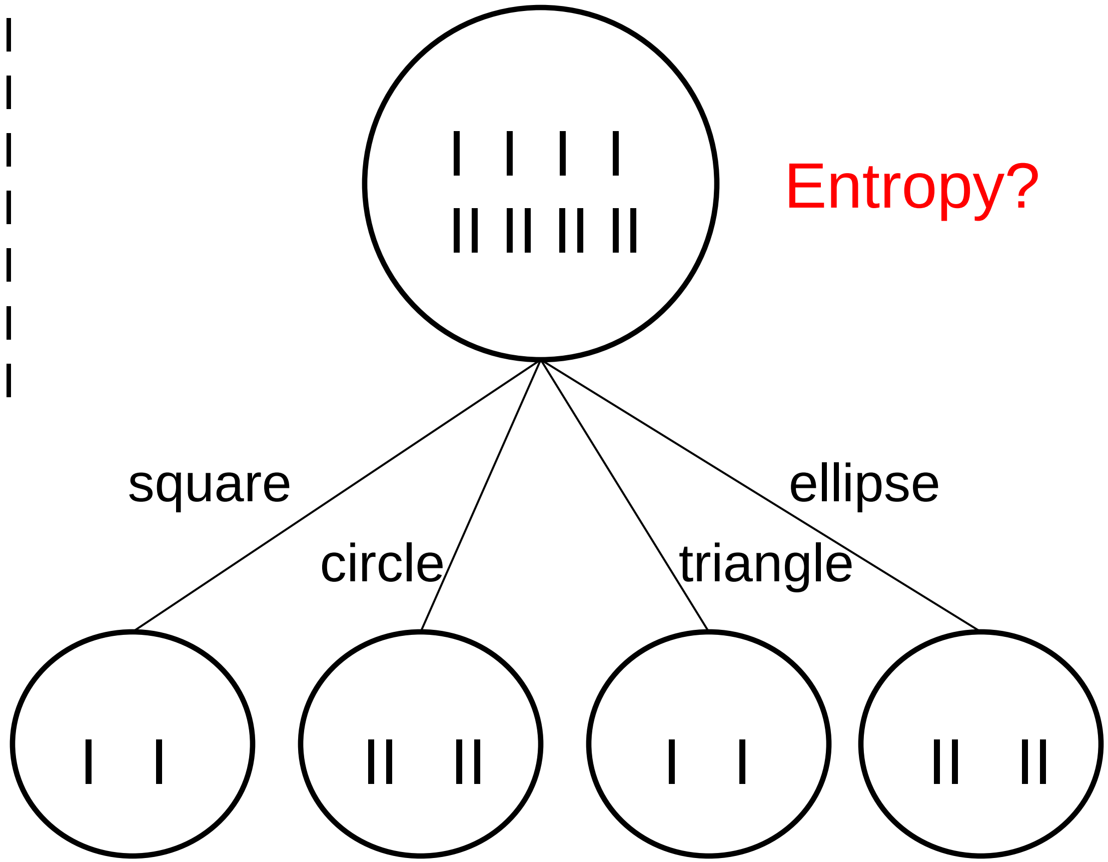
Decision tree for this training set.



What would be different about a real training set?

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

Try the shape feature

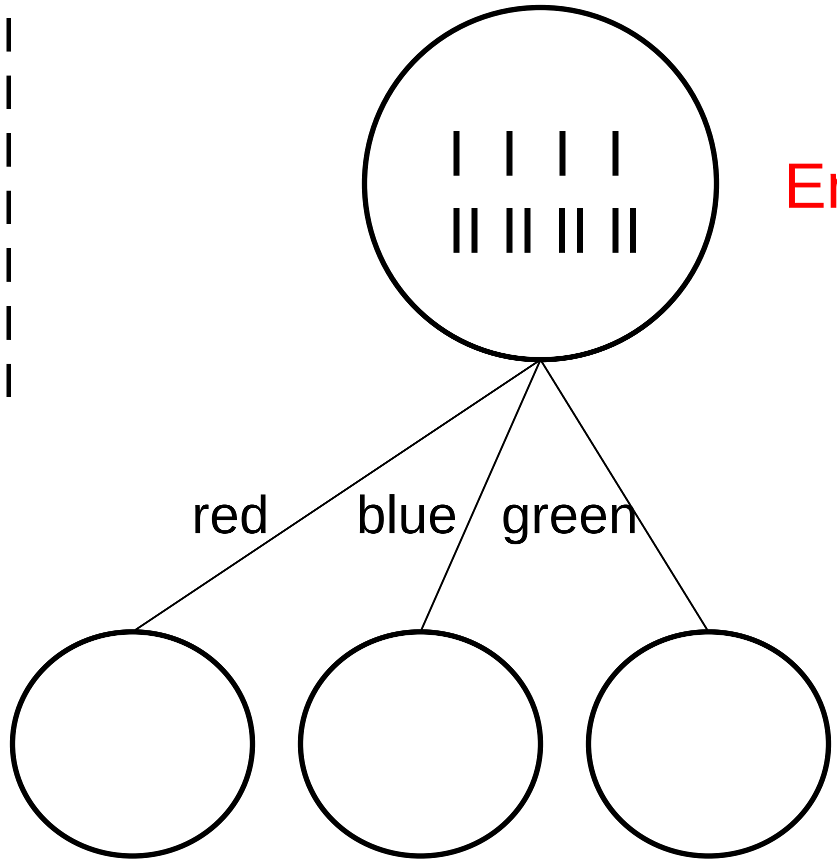


Entropy? Entropy? Entropy? Entropy?

GAIN?

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

Try the color feature



Entropy?

Entropy? Entropy? Entropy?

GAIN?

Many-Valued Features

- Your features might have a large number of discrete values.

Example: pixels in an image have (R,G,B)

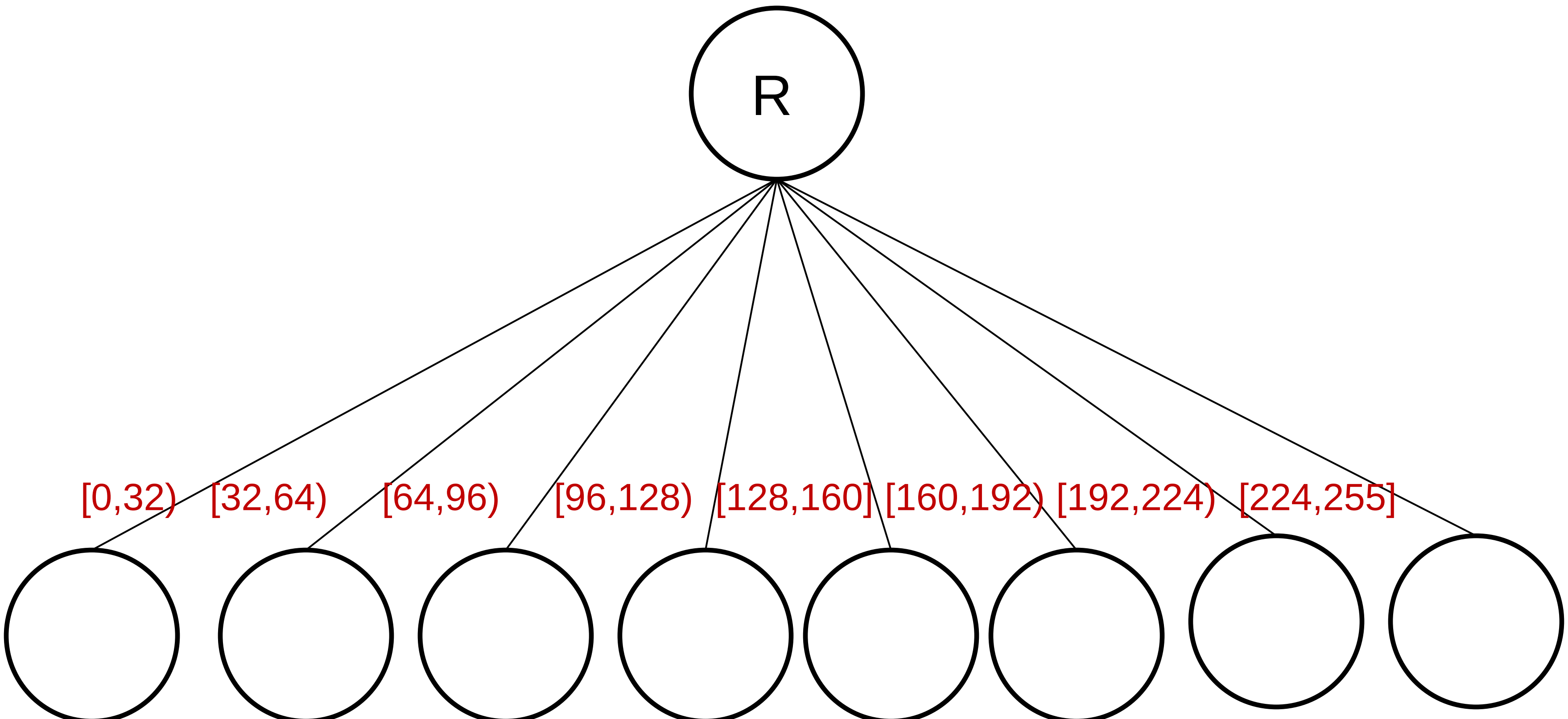
which are each integers between 0 and 255.

- Your features might have continuous values.

Example: from pixel values, we compute gradient magnitude, a continuous feature

Solution to Both

- We often group the values into bins



Training and Testing

- Divide data into a training set and a separate testing set.
- Construct the decision tree using the training set only.
- Test the decision tree on the training set to see how it's doing.
- Test the decision tree on the testing set to report its real performance.

Measuring Performance

- Given a test set of labeled feature vectors

e.g. (square,red) I

- Run each feature vector through the decision tree
- Suppose the decision tree says it belongs to class X and the real label is Y
- If (X=Y) that’s a correct classification
- If (X<>Y) that’s an error

Measuring Performance

- Accuracy = # correct / # total
- In a 2-class problem, where the classes are positive or negative (ie. for cancer)
  - # true positives TP
  - # true negatives TN
  - # false positives FP
  - # false negatives FN
- Precision = TP / (TP + FP)

  How many of the ones you said were cancer really were cancer?
- Recall = TP / (TP + FN)

  How many of the ones who had cancer did you call cancer?

Measuring Performance

- For multi-class problems, we often look at the confusion matrix.

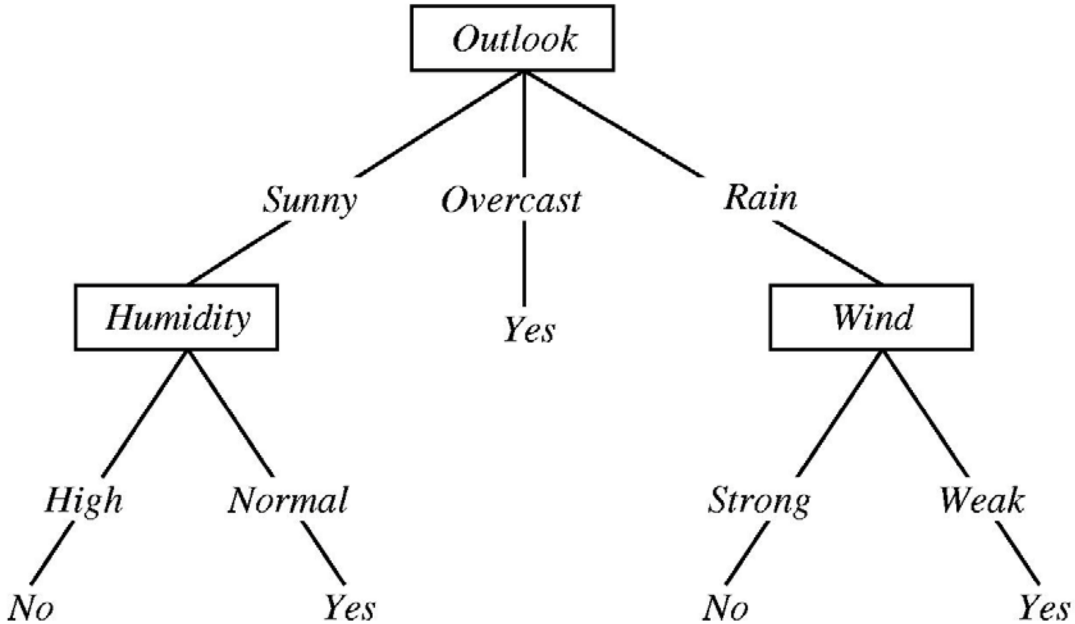
assigned class

true class

	A	B	C	D	E	F	G
A							
B							
C							
D							
E							
F							
G							

C(i,j) = number of times (or percentage) class i is given label j.

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

Error here means (1 – accuracy).

- 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

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

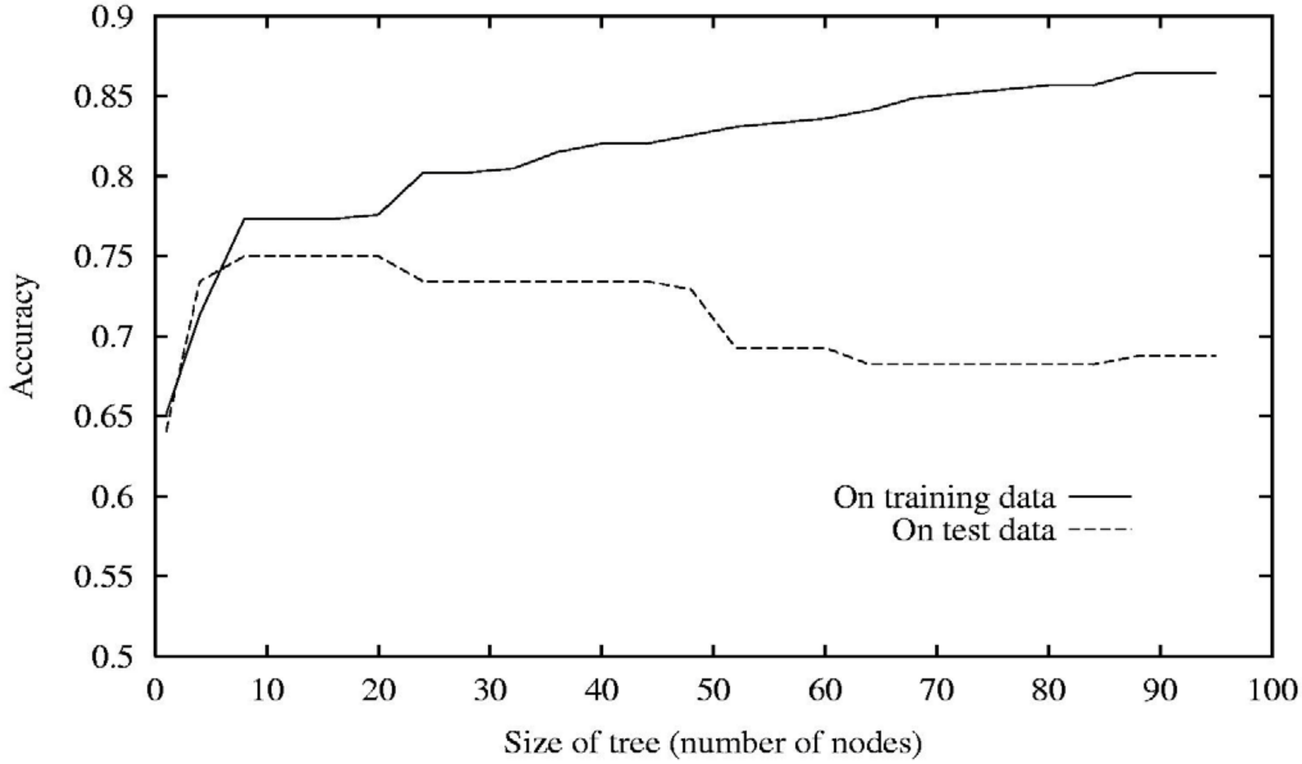
and

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

Hypothesis here means classification by the decision tree.

What happens as the decision tree gets bigger and bigger?

Overfitting in Decision Tree Learning



Error on the training data goes down.
Error on the testing data goes up.

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

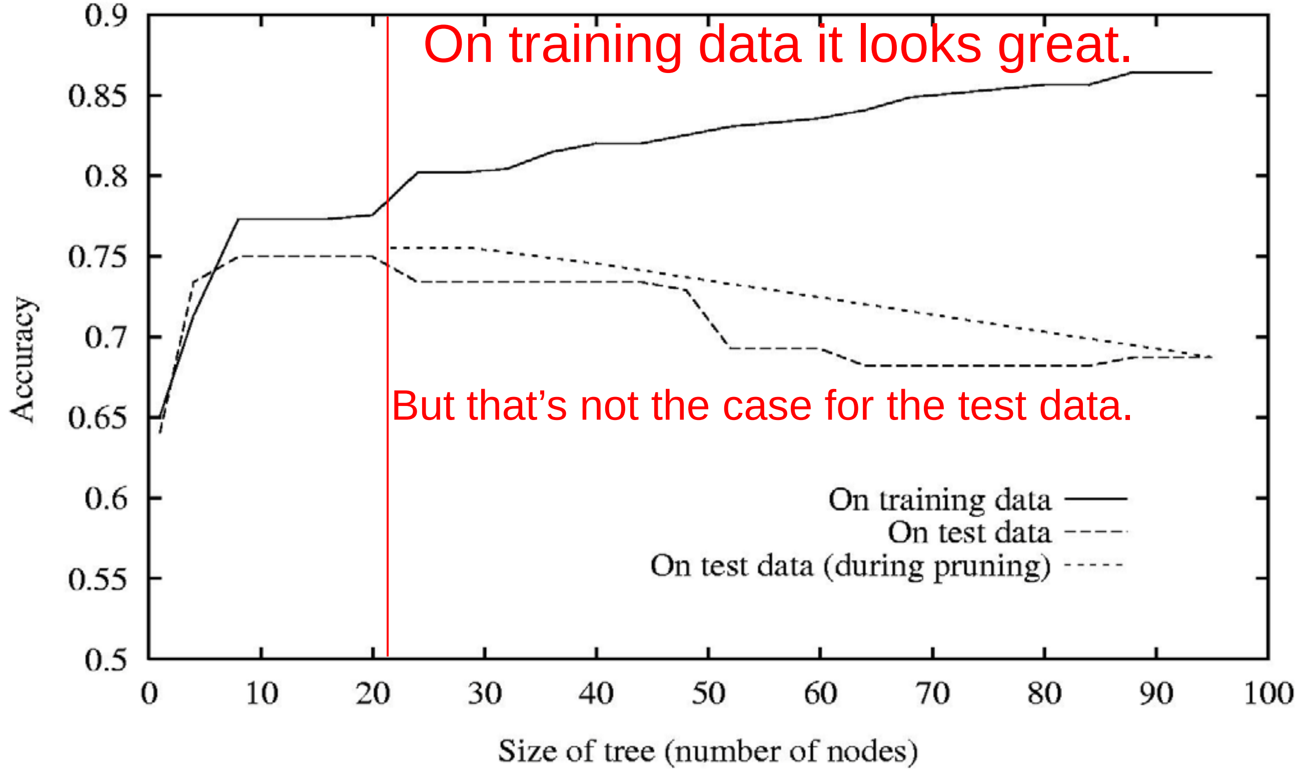
Split data into *training* and *validation* set

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

Then you have to have a separate testing set!

Effect of Reduced-Error Pruning



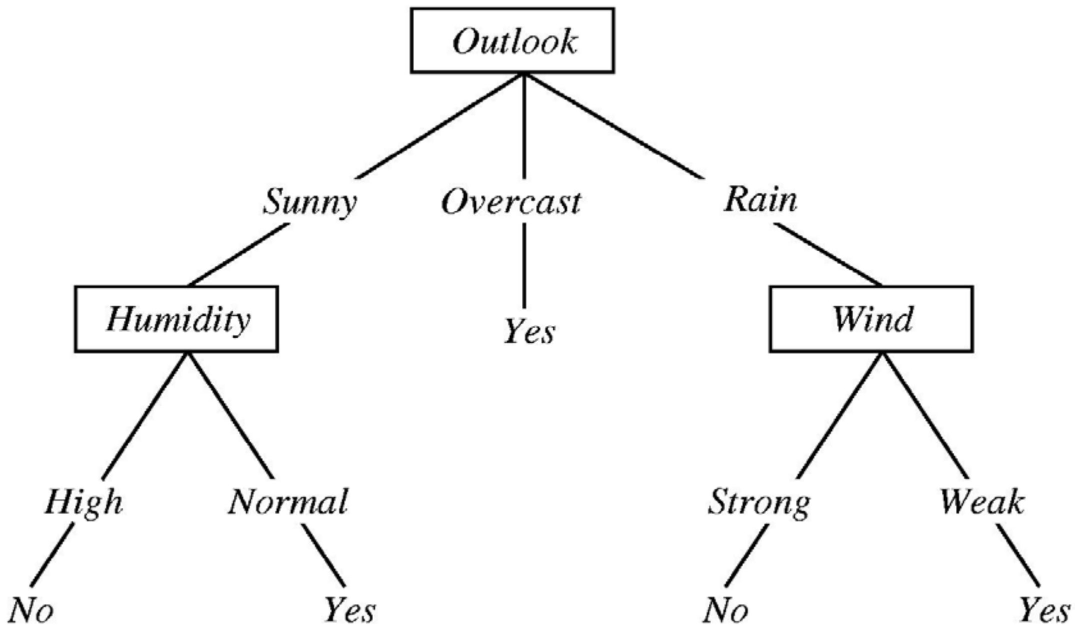
The tree is pruned back to the red line where it gives more accurate results on the test data.

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

THEN $\quad PlayTennis = No$

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

THEN $\quad PlayTennis = Yes$

$\ldots$

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=none
- Heuristic function=information gain or information content or others
- Overfitting and pruning
- Advantage is simplicity and easy conversion to rules.