Decision Trees

The University of Texas at Dallas

Learning Decision Trees

Decision trees provide a very popular and efficient hypothesis space.

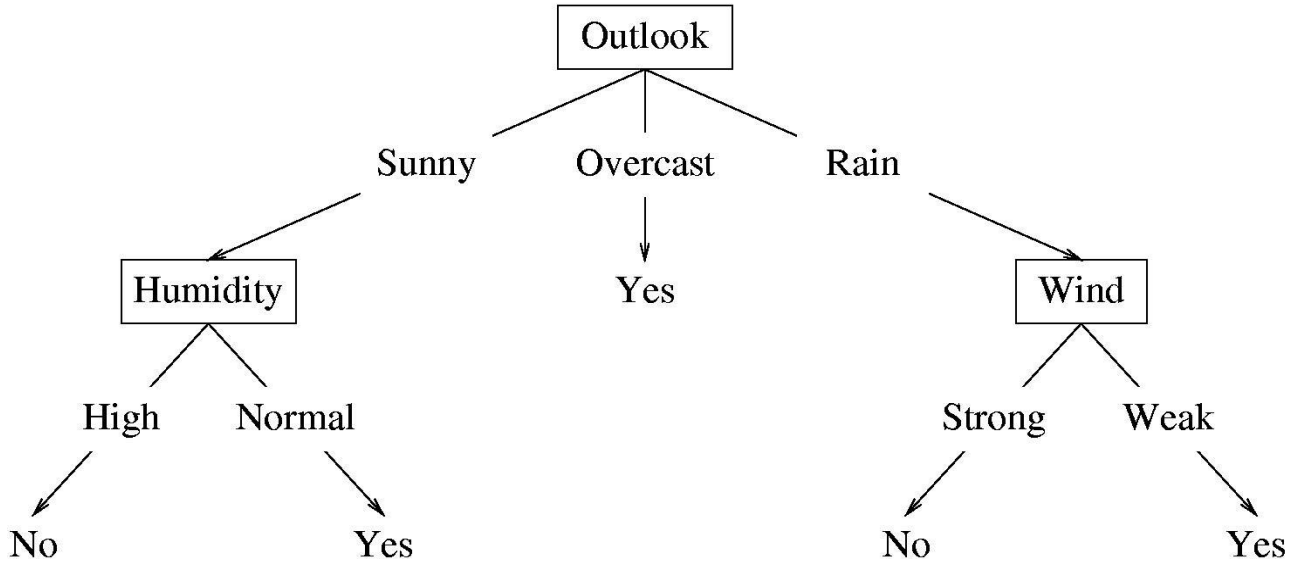
- **Variable Size**. Any boolean function can be represented.
- **Deterministic**.
- **Discrete and Continuous Parameters**.

Learning algorithms for decision trees can be described as

- **Constructive Search**. The tree is built by adding nodes.
- **Eager.**
- **Batch** (although online algorithms do exist).

Decision Tree Hypothesis Space

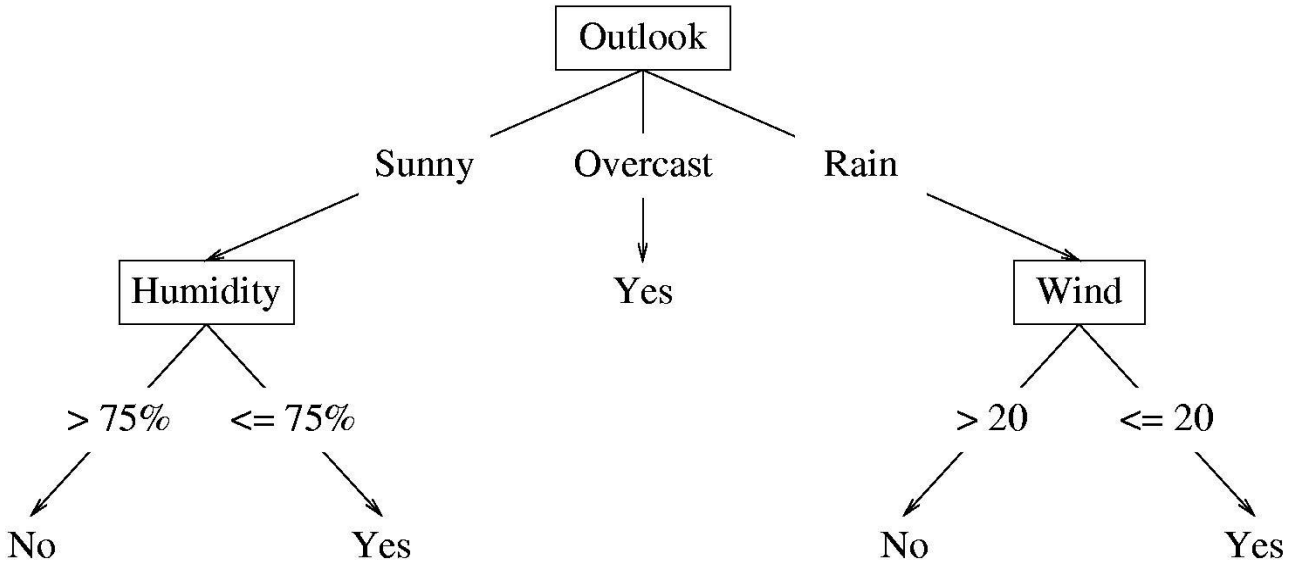
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



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.

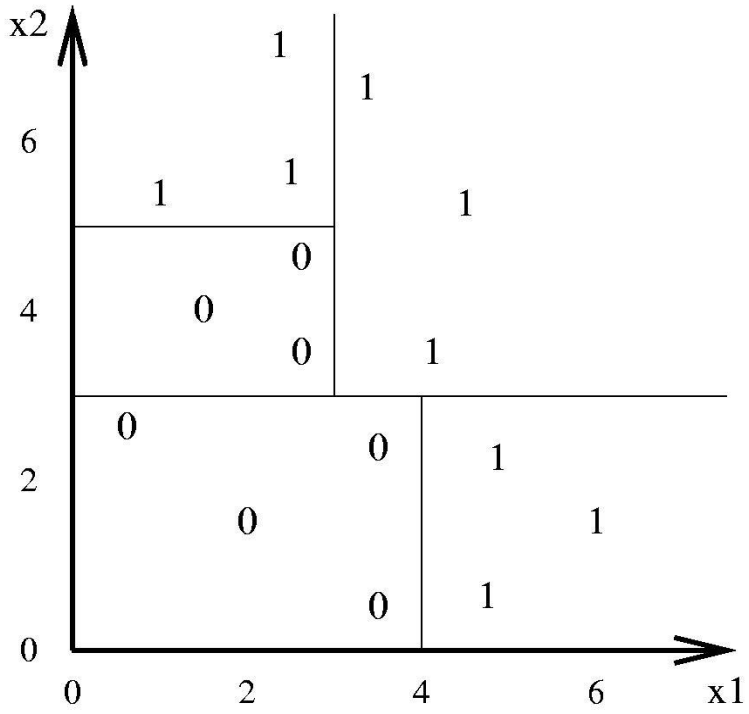
Decision Tree Hypothesis Space

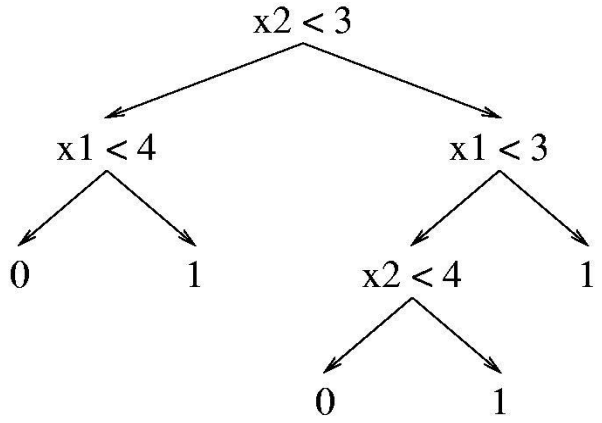
If the features are continuous, internal nodes may test the value of a feature against a threshold.



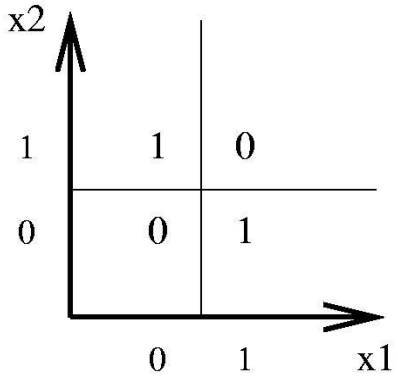
Decision Tree Decision Boundaries

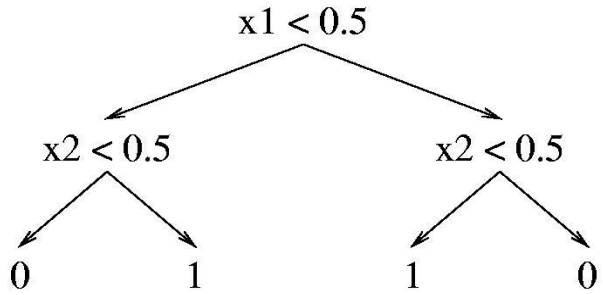
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees Can Represent Any Boolean Function





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

Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$
- **etc.**

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))

Choosing the best Attribute?

- Fundamental principle underlying tree creation
  - Simplicity
  - Occam's Razor: Simplest model that explains the data should be preferred
- Each node divides the data into subsets
  - Make each subset as **pure** as possible.

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 $y \neq y_0$

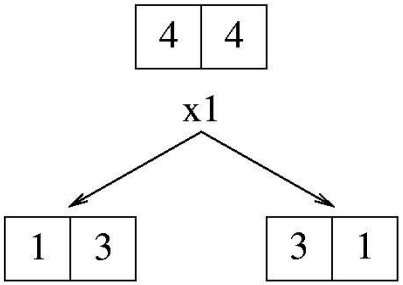
$J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$

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

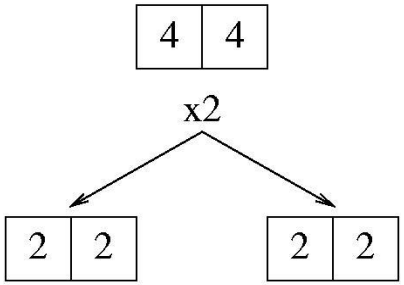
return j

Choosing the Best Attribute—An Example

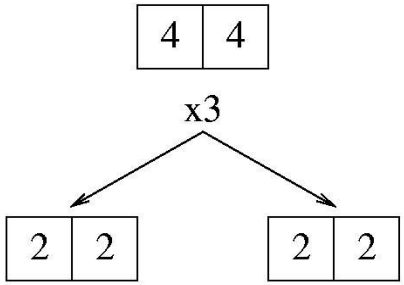
x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



J=2



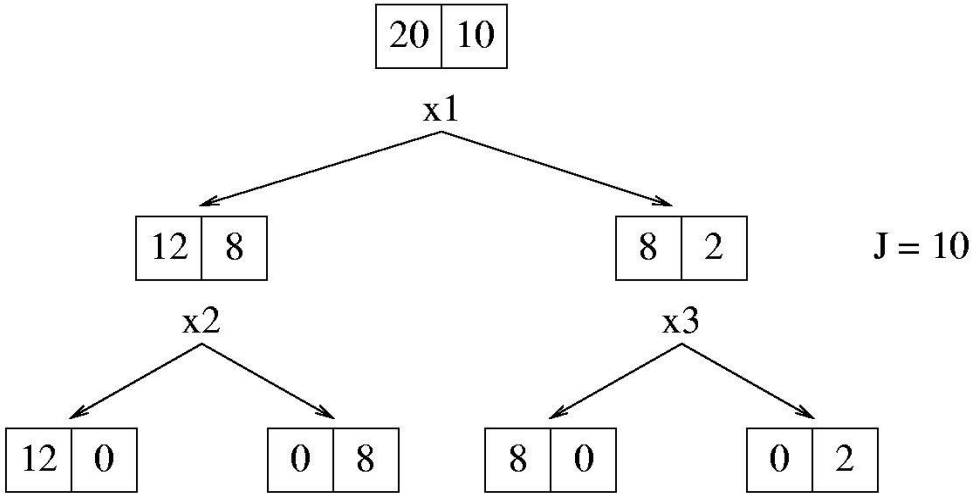
J=4



J=4

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.



A better heuristic from Information Theory

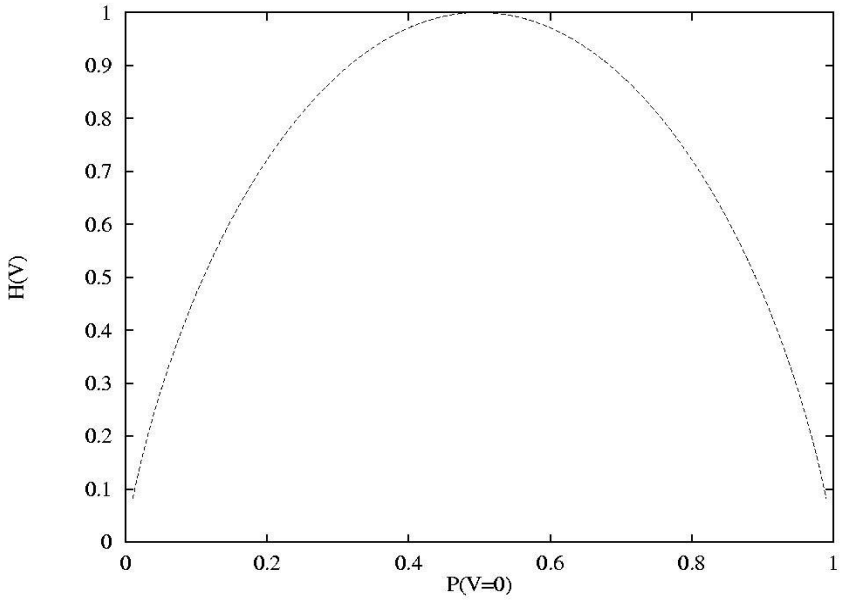
- Entropy as a measure of (im)purity
  - $Entropy(V) = -\sum_j P(class_j)\, lg[P(class_j)]$
  - j indexes the classes
  - P($class_j$) is the fraction of examples that have $class_j$
  - Entropy is 0 if all classes are the same
  - Entropy is maximum if the classes are equally likely.

Entropy

The *entropy* of V, denoted $H(V)$ is defined as follows:

$$H(V) = \sum_{v=0}^{1} -P(H = v) \lg P(H = v).$$

This is the average surprise of describing the result of one "trial" of V (one coin toss).



Entropy can be viewed as a measure of uncertainty.

High, Low Entropy

- "High Entropy"
  - X is from a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable
- "Low Entropy"
  - X is from a varied (peaks and valleys) distribution
  - Histogram has many lows and highs
  - Values sampled from it are more predictable

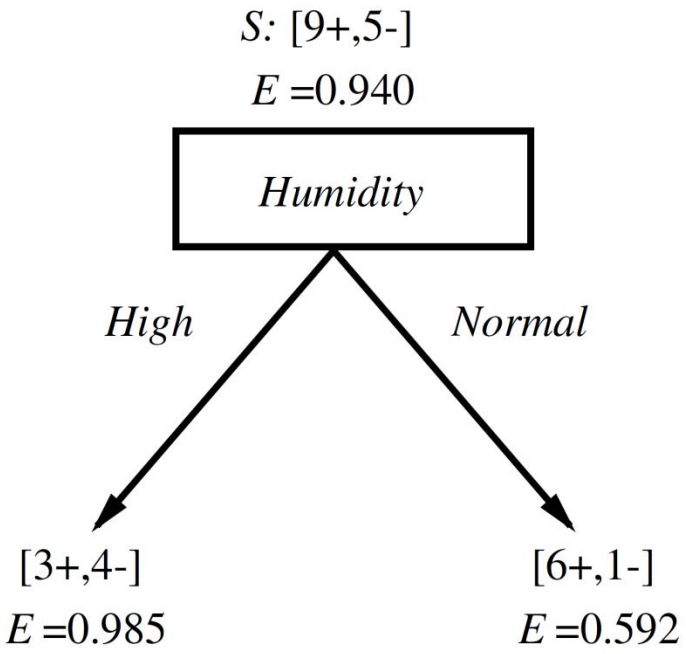
Information Gain: Using Entropy to make decisions

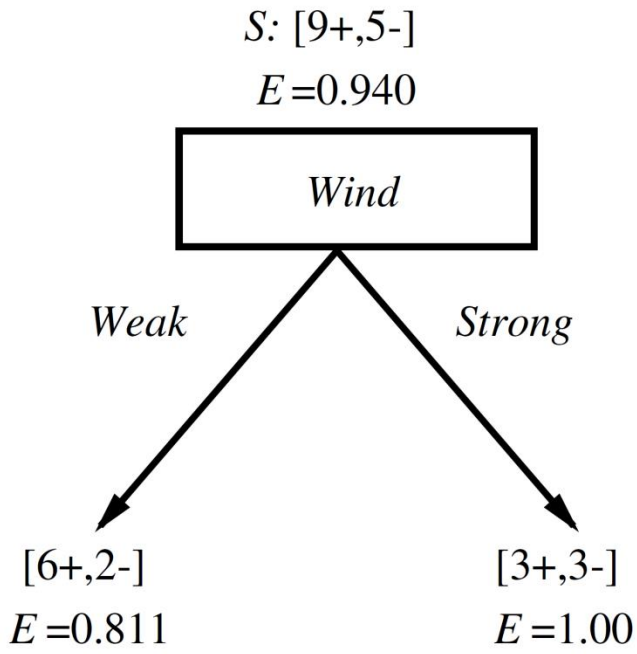
- $Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} P(v) Entropy(S_v)$
- Entropy is impurity in data
- Entropy(S): current impurity
- The second term measures the expected impurity after partitioning the data with respect to the A, i.e. new impurity
- Gain=Reduction in impurity
  - We want to be as pure as possible, i.e. maximize reduction in impurity
- So we want to maximize Gain!

Read Wikipedia articles on Entropy and Mutual Information

When do I play tennis?

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





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

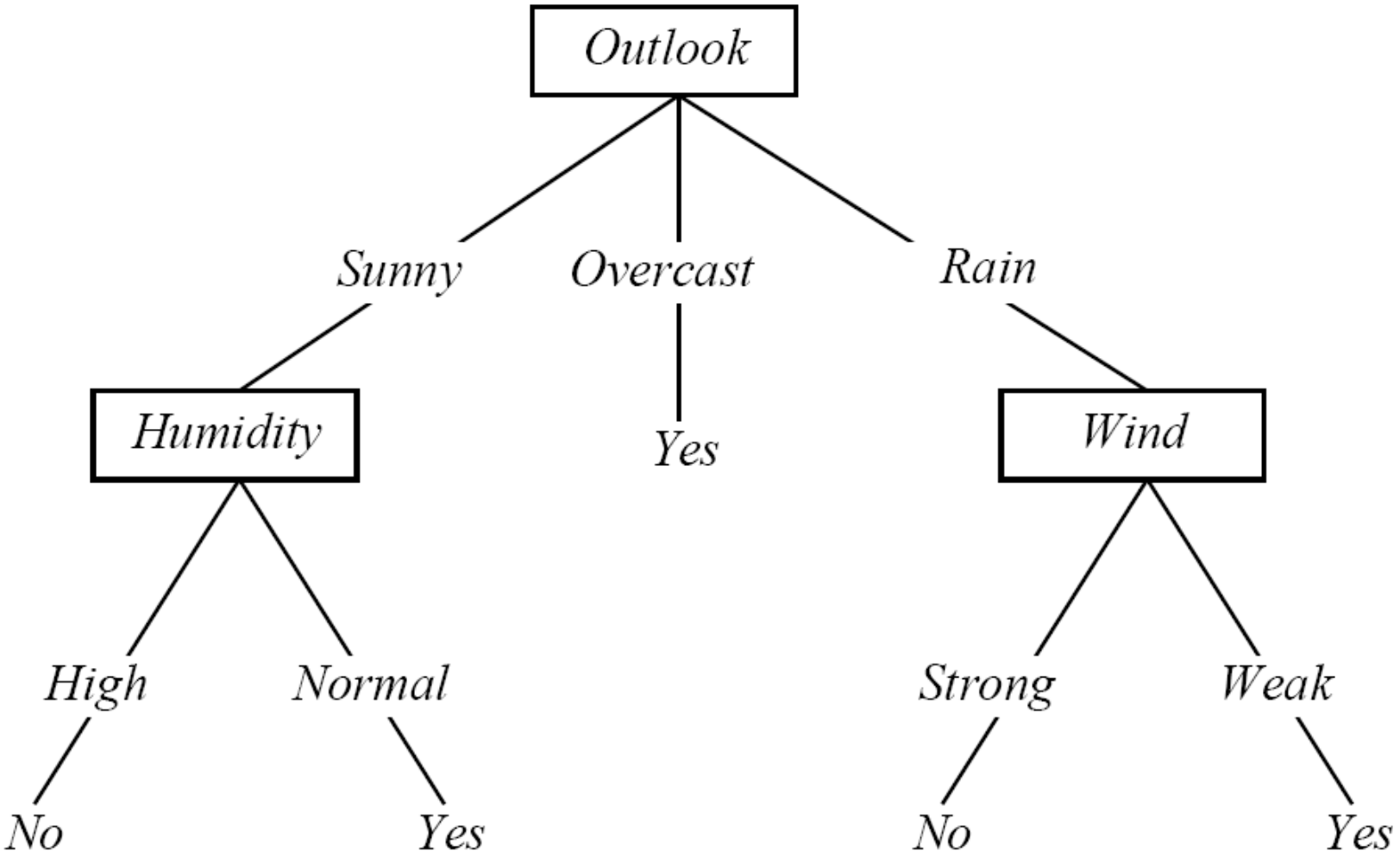
Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

When do I play tennis?

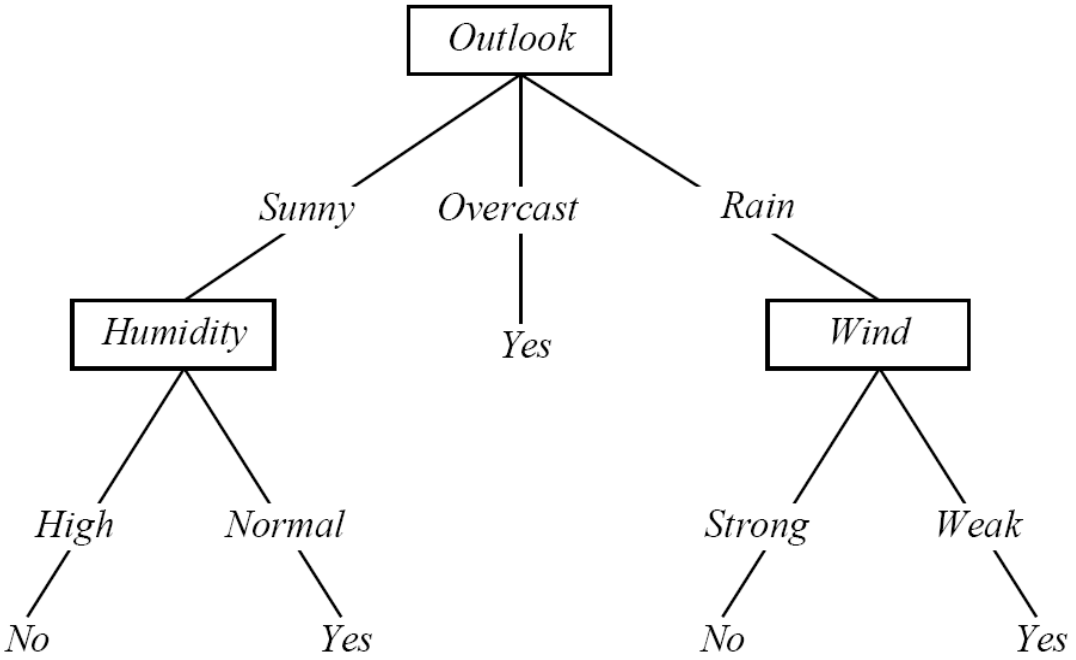
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

Decision Tree



Is the decision tree correct?

- Let's check whether the split on Wind attribute is correct.
- We need to show that Wind attribute has the highest information gain.



When do I play tennis?

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

Wind attribute – 5 records match

Note: calculate the entropy only on examples that got "routed" in our branch of the tree (Outlook=Rain)

Day					PlayTennis
D1					No
D2					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

Calculation

- *S = {D4, D5, D6, D10, D14}*
- Entropy:

  H(S) = – 3/5log(3/5) – 2/5log(2/5) = 0.971
- Information Gain

  IG(S,Temp) = H(S) – H(S|Temp) = 0.01997

  IG(S, Humidity) = H(S) – H(S|Humidity) = 0.01997

  IG(S,Wind) = H(S) – H(S|Wind) = 0.971

Decision Trees: Hypothesis Spaces and Search methods recap

- We search a variable-sized hypothesis space
  - We start at empty and grow it as we build it
- Space is Complete: All target concepts are included in this space
- Local search: No Backtracking
- Batch: At each step, we use all training examples to make a statistically based decision.

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.

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

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

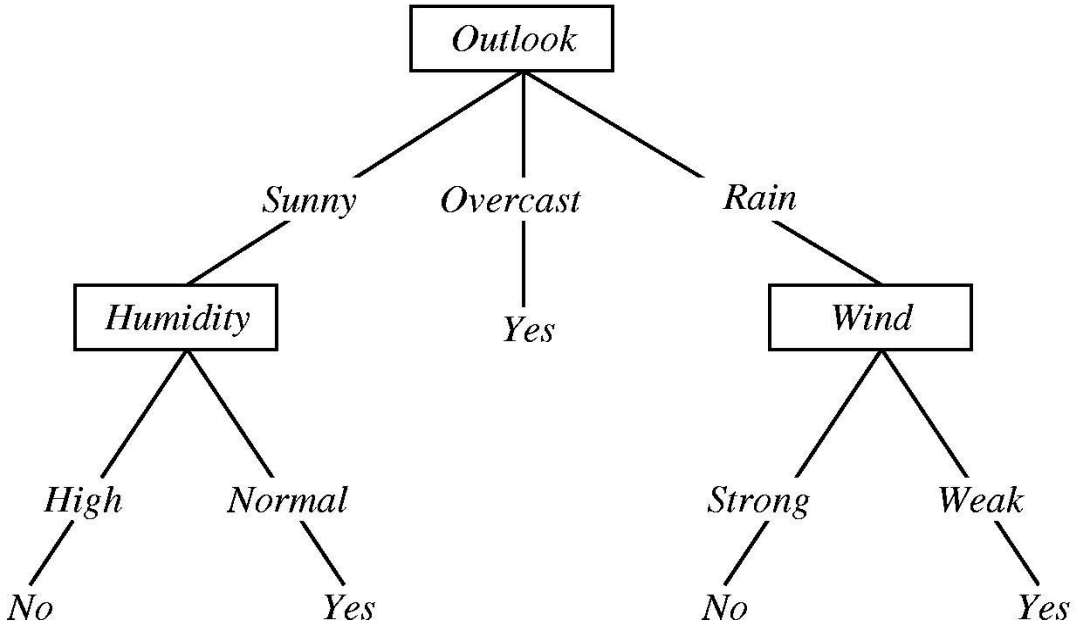
$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

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

A deeper look at Gain Ratio

- What is Split In Information of Date?
  - Data set size n, each has a different date.
- What is Split In Information of an attribute "Weather=Snowing" in Texas?
  - Snows one day and is sunny or overcast on others
- Heuristic
  - First compute Gain
  - Apply Gain ratio only on attributes which have above average Gain.

Overfitting in Decision Trees

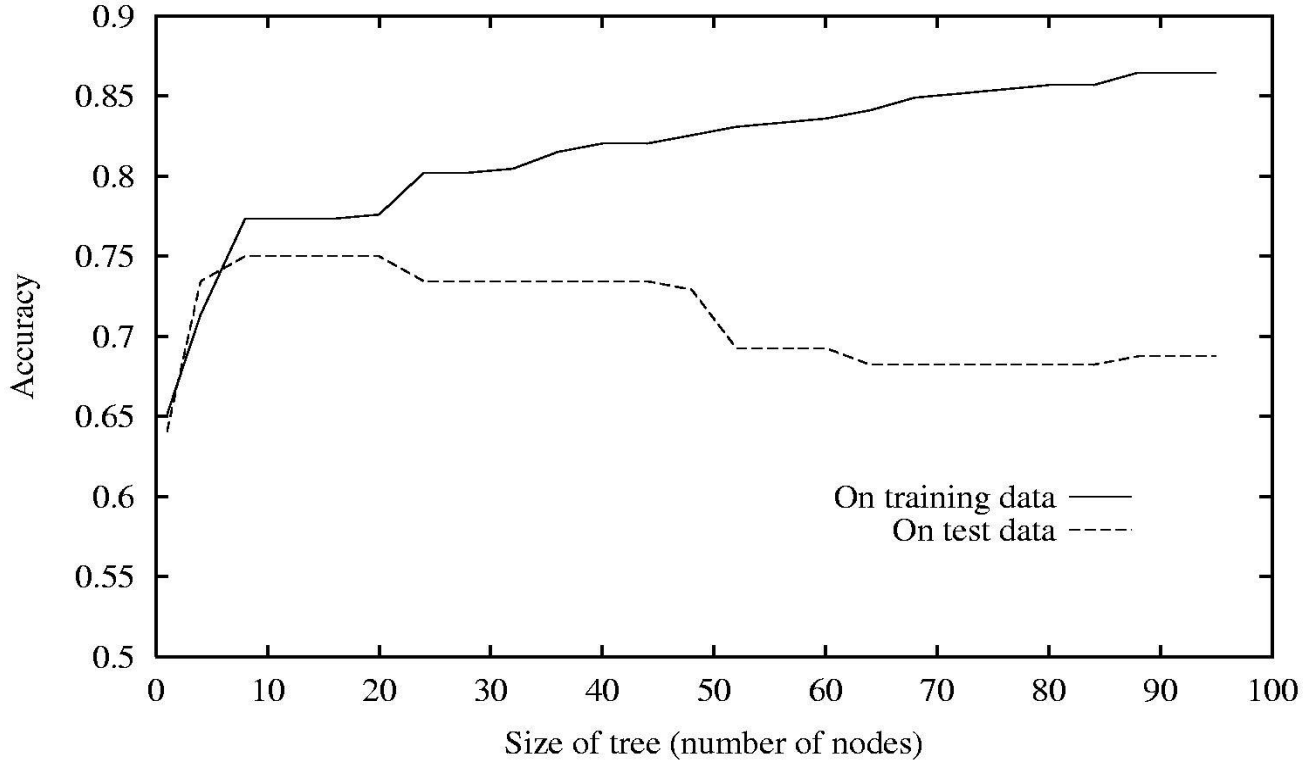


Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Overfitting in Decision Tree Learning



Sources of Overfitting

- Noise
- Small number of examples associated with each leaf
  - What if only one example is associated with a leaf. Can you believe it?
  - Coincidental regularities
- Generalization is the most important criteria
  - Your method should work well on examples which you have not seen before.

Avoiding Overfitting

- Two approaches
  - Stop growing the tree when data split is not statistically significant
  - Grow tree fully, then post-prune
- Key Issue: What is the correct tree size?
  - Divide data into training and validation set
    - Random noise in two sets might be different
  - Apply statistical test to estimate whether expanding a particular node is likely to produce an improvement beyond the training set
  - Add a complexity penalty

Reduced-Error Pruning

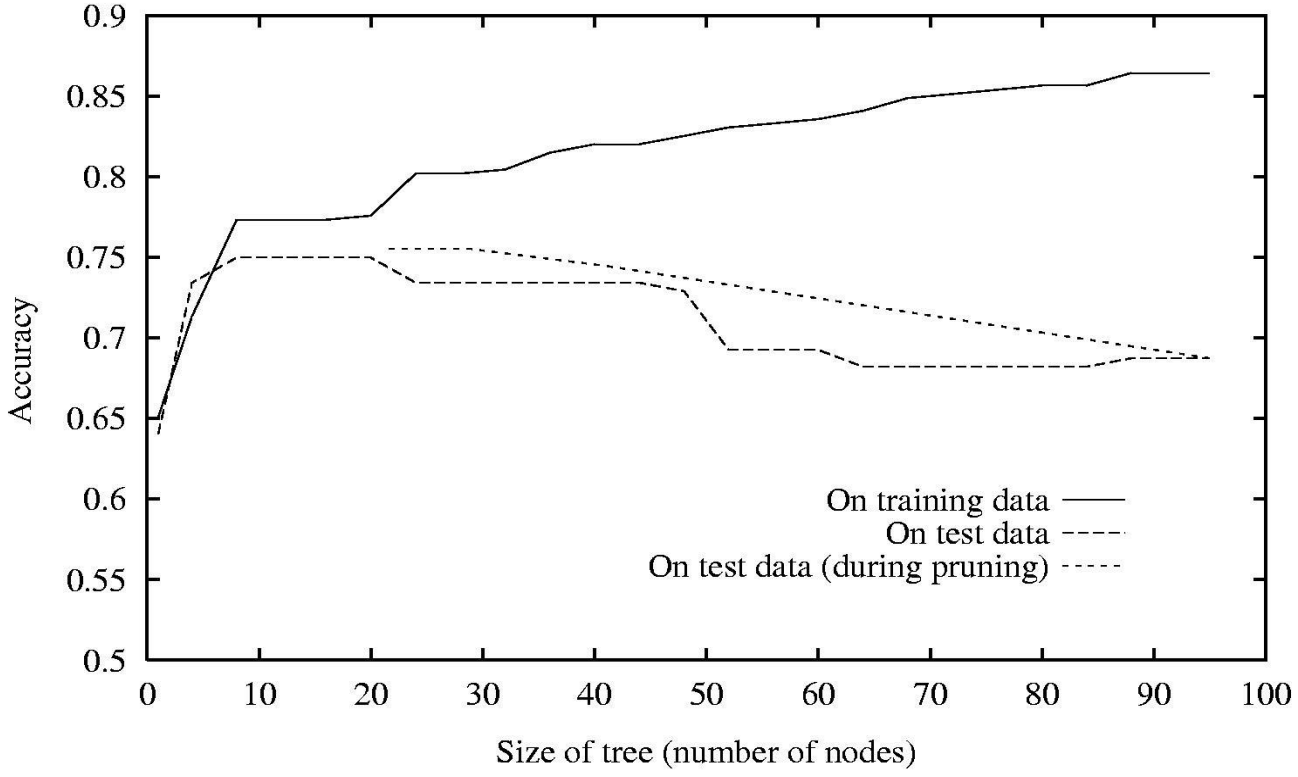
Split data into *training* and *validation* set

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

Leaf nodes added because of coincidental regularities are likely to be pruned because the same coincidences are unlikely to occur in the validation set

Effect of Reduced-Error Pruning

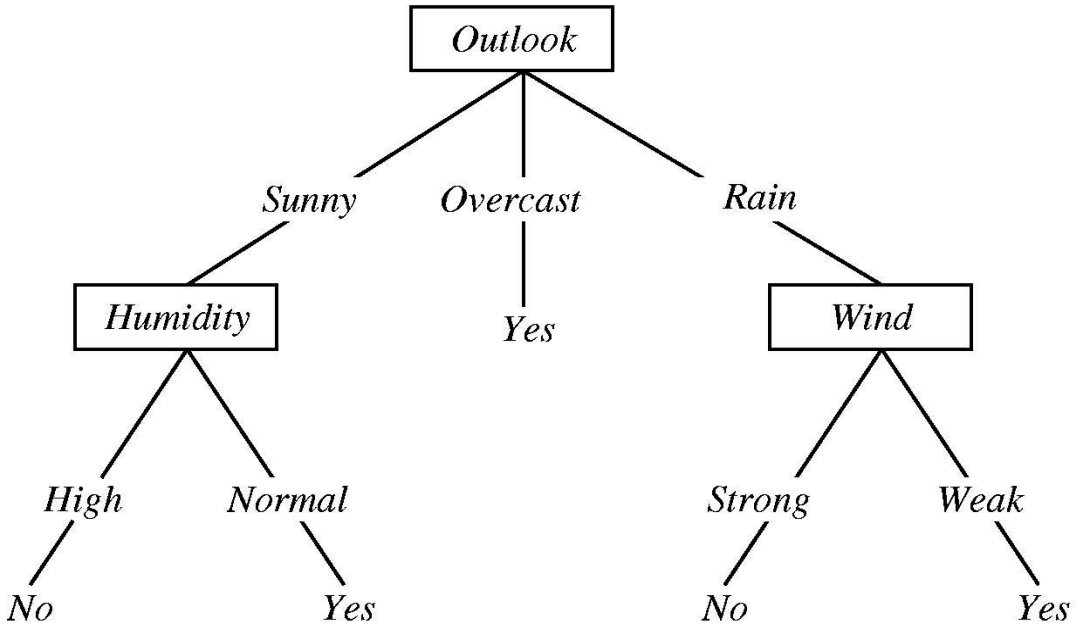


Rule Post-Pruning

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

THEN $PlayTennis = No$

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

THEN $PlayTennis = Yes$

...

Handling Missing Values

- Missing values: Some attribute-values in an example are missing
  - Example: patient data. You don't expect blood test results for everyone.
- Treat the missing value as another value
- Ignore instances having missing values
  - Problematic because you are throwing away important data. Data is scarce.

Handling Missing Values

- Probabilistic approach
  - Assign a probability to each possible value of A
  - Let us assume that A(x=1)=0.4 and A(x=0)=0.6
  - A fractional 0.4 of instance goes to branch A(x=1) and 0.6 to branch A(x=0)
  - Use fractional instances to compute gain
- Classification
  - Most probable classification

Handling Continuous attributes

- Thresholding
- How to select Thresholds?

40	48	60	72	80	90
no	no	yes	yes	yes	no

- Pick a threshold that has the highest gain!
- Sort the examples and calculate gain at all points where the classification changes from “yes to no” or “no to yes”
  - Provably maximizes the information gain.

Summary: Decision Trees

- Representation
- Tree growth
- Choosing the best attribute
- Overfitting and pruning
- Special cases: Missing Attributes and Continuous Attributes
- Many forms in practice: CART, ID3, C 4.5