CSCI
Machine Learning

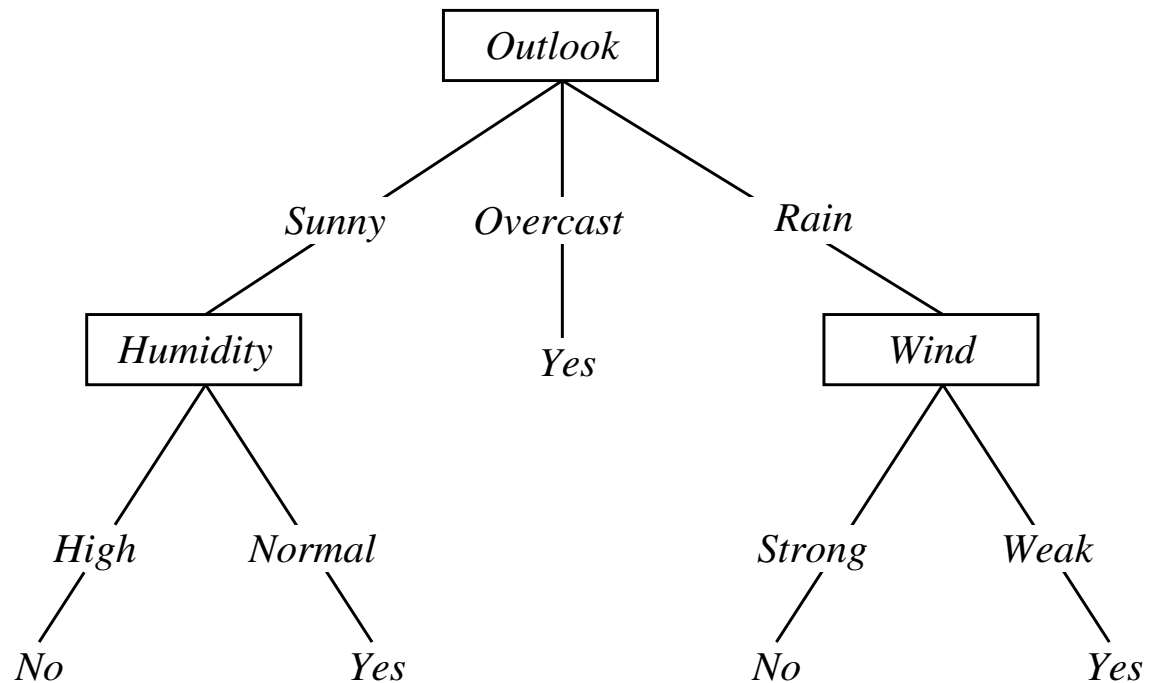Mehdi Allahyari

Georgia Southern University

(sources: slides are based on material from a variety of sources, including Pedro Domingos, Ali Borji, Doug Downey, Tom Mitchell, Emily Fox, Raquel Urtasun, Andrew Ng, and others)

Example

Decision Tree for *PlayTennis*

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

Decision Trees

- ## Decision tree representation
  - Each internal node tests an attribute
  - Each branch corresponds to an attribute value
  - Each leaf node assigns a classification
- ## When to Consider Decision Trees
  - Instances describable by attribute-value pairs
  - Target function is discrete (or continuous) valued
  - Disjunctive hypothesis may be required
  - Possibly noisy training data

- ## Examples
  - Equipment or medical diagnosis
  - Credit risk analysis

Function approximation

Problem Setting:
- Set of possible instances X
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H = \{ h \mid h : X \rightarrow Y \}$

Input:
- Training examples $\{<x^{(i)}, y^{(i)}>\}$ of unknown target function f
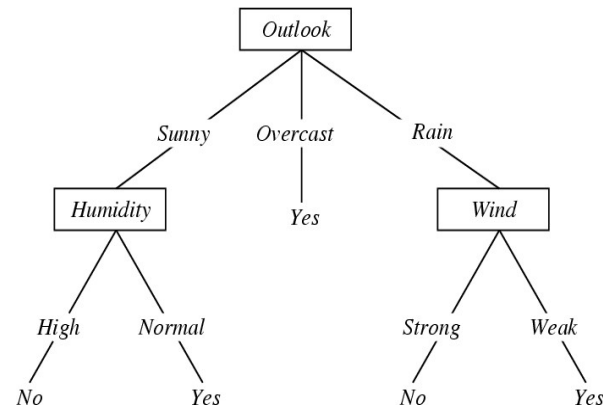
superscript: i^{th} training example

Output:
- Hypothesis $h \in H$ that best approximates target function f

Decision Trees

- ## A Decision tree for
- ## F: <Outlook, Humidity, Wind, Temp> → PlayTennis?

- Each internal node: test one attribute X_i

- Each branch from a node: selects one value for X_i

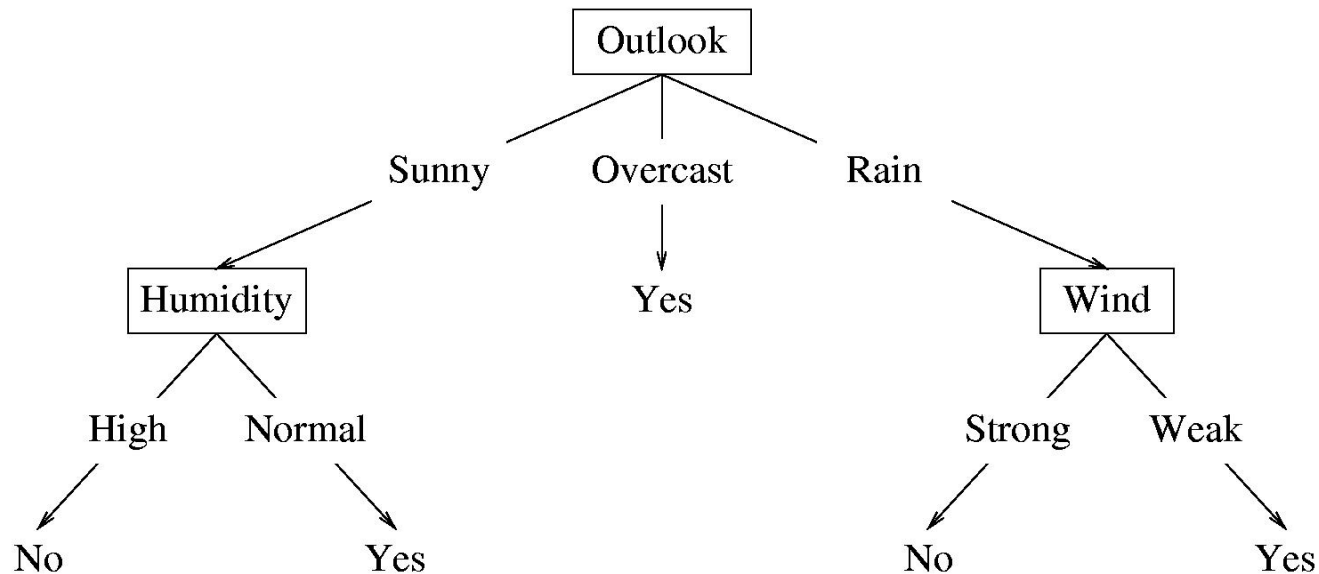- Each leaf node: predict Y (or P(Y|X))

Decision Tree Learning

Problem Setting:

- ## Set of possible instances X

  - each instance x in X is a feature vector

  - e.g., *<Humidity=low, Wind=weak, Outlook=rain, Temp=hot>*

- ## Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued

- ## Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y
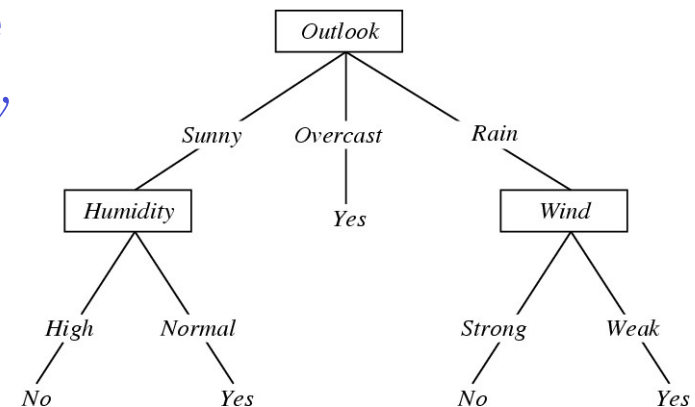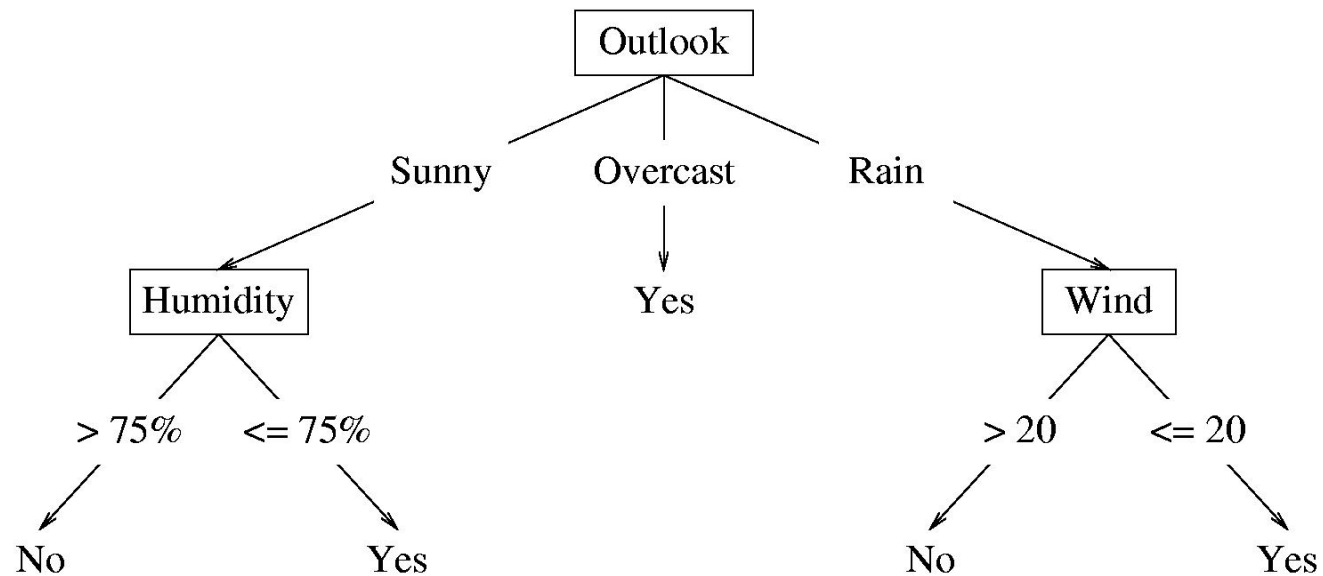
Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.

- **Leaf nodes** specify the class $h(\mathbf{x})$.

```
                          ┌─────────┐
                          │ Outlook │
                          └─────────┘
                  Sunny      Overcast      Rain
              ┌───────────┐    │      ┌────────┐
              │ Humidity  │   Yes     │  Wind  │
              └───────────┘           └────────┘
            High    Normal         Strong    Weak

            No        Yes            No        Yes
```

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.
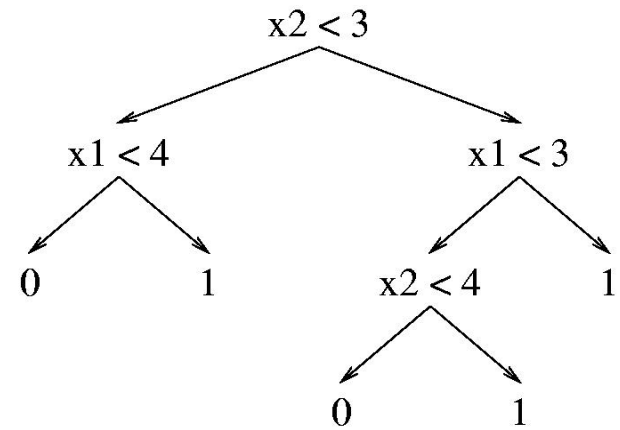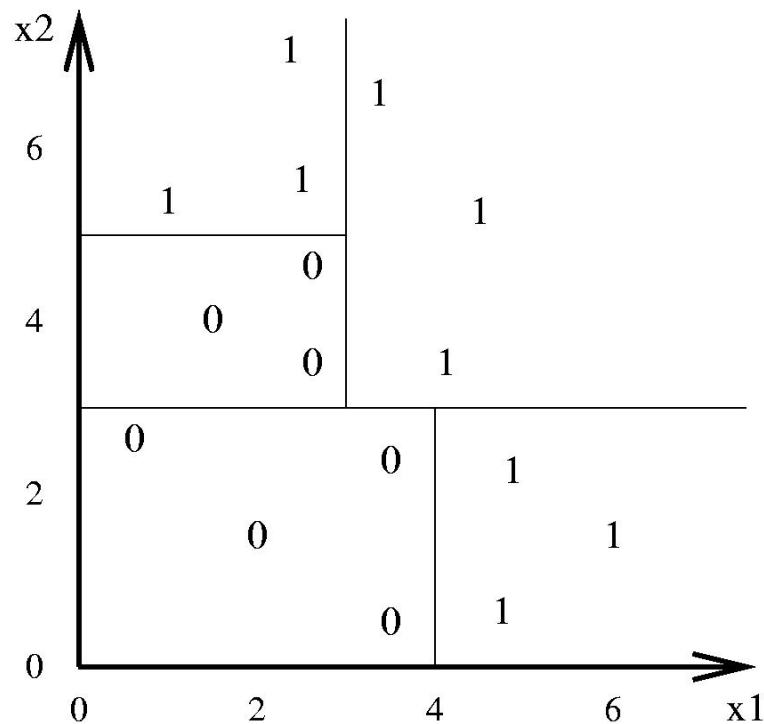
Decision Tree Hypothesis Space

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

```
                        ┌─────────┐
                        │ Outlook │
                        └─────────┘
                 Sunny     Overcast     Rain

          ┌──────────┐       Yes      ┌──────┐
          │ Humidity │                │ Wind │
          └──────────┘                └──────┘
        > 75%    <= 75%            > 20      <= 20

        No          Yes          No            Yes
```
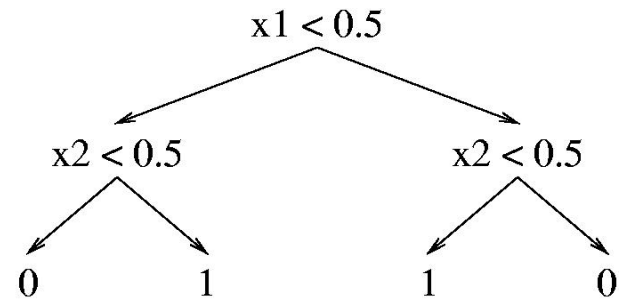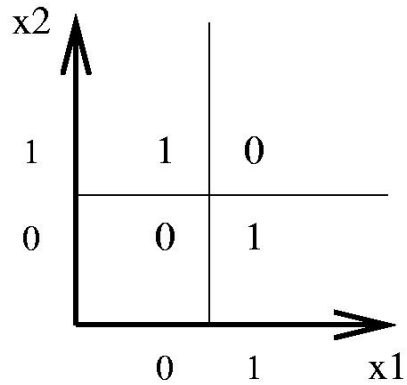
Decision Tree Decision Boundaries

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

x2

1   1   0

0   0   1

0   1   x1

$x1 < 0.5$

$x2 < 0.5$ $x2 < 0.5$

0   1   1   0

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))

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?

[29+,35-]   A1=?   [29+,35-]   A2=?

   t   f   t   f

[21+,5-]   [8+,30-]   [18+,33-]   [11+,2-]
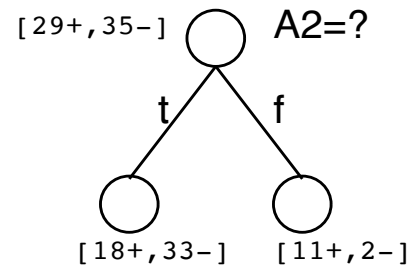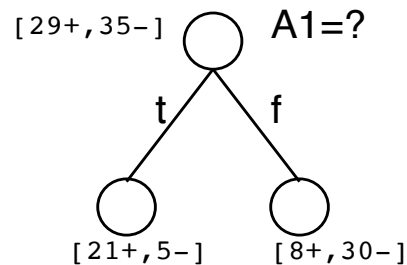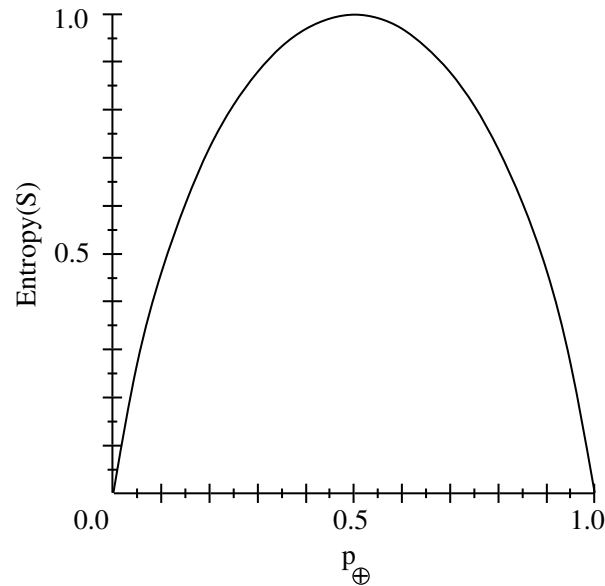
Entropy

- Entropy $H(X)$ of a random variable X

  # of possible values for X

$$H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)$$

- $H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

- Why? Information theory:
  - Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=I$
  - So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

Sample 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 $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

- Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y = v) = -\sum_{i=1}^{n} P(X = i|Y = v) \log_2 P(X = i|Y = v)$$

- Conditional entropy $H(X|Y)$ of X given Y :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y = v) H(X|Y = v)$$

- Mutual information (aka Information Gain) of X and Y :
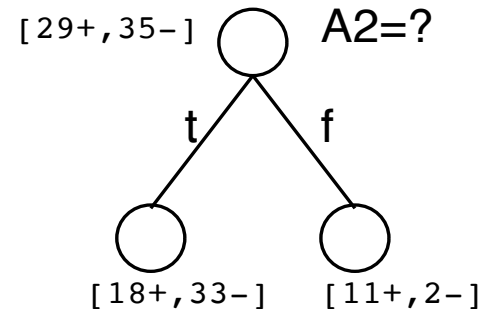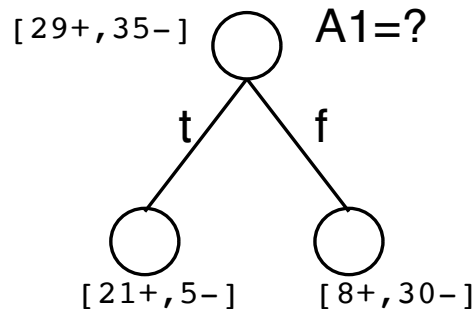
$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Information Gain

- Information Gain is the mutual information between input attribute A and target variable Y

- Information Gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A

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

$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$

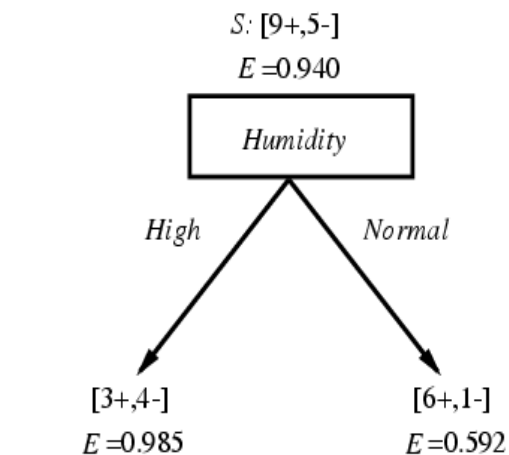$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

[29+,35−] ◯ A1=?

t / f

◯ [21+,5−]   ◯ [8+,30−]

[29+,35−] ◯ A2=?

t / f

◯ [18+,33−]   ◯ [11+,2−]

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
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

Which attribute is the best classifier?

$S: [9+,5-]$
$E = 0.940$

Humidity

High — Normal

[3+,4-]
$E = 0.985$

[6+,1-]
$E = 0.592$

Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

$S: [9+,5-]$
$E = 0.940$

Wind

Weak — Strong

[6+,2-]
$E = 0.811$

[3+,3-]
$E = 1.00$
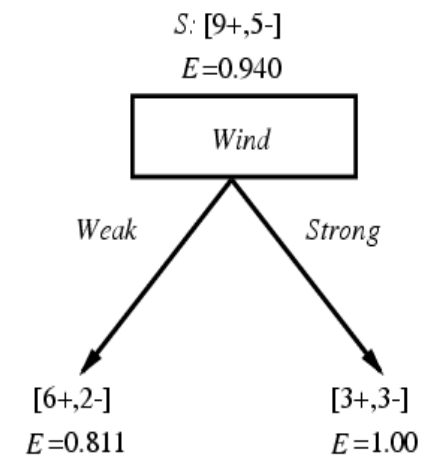
Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

{D1, D2, ..., D14}

[9+,5−]

```
Outlook
```

Sunny   Overcast   Rain

{D1,D2,D8,D9,D11}   {D3,D7,D12,D13}   {D4,D5,D6,D10,D14}

[2+,3−]   [4+,0−]   [3+,2−]

```
  ?                Yes                ?
```
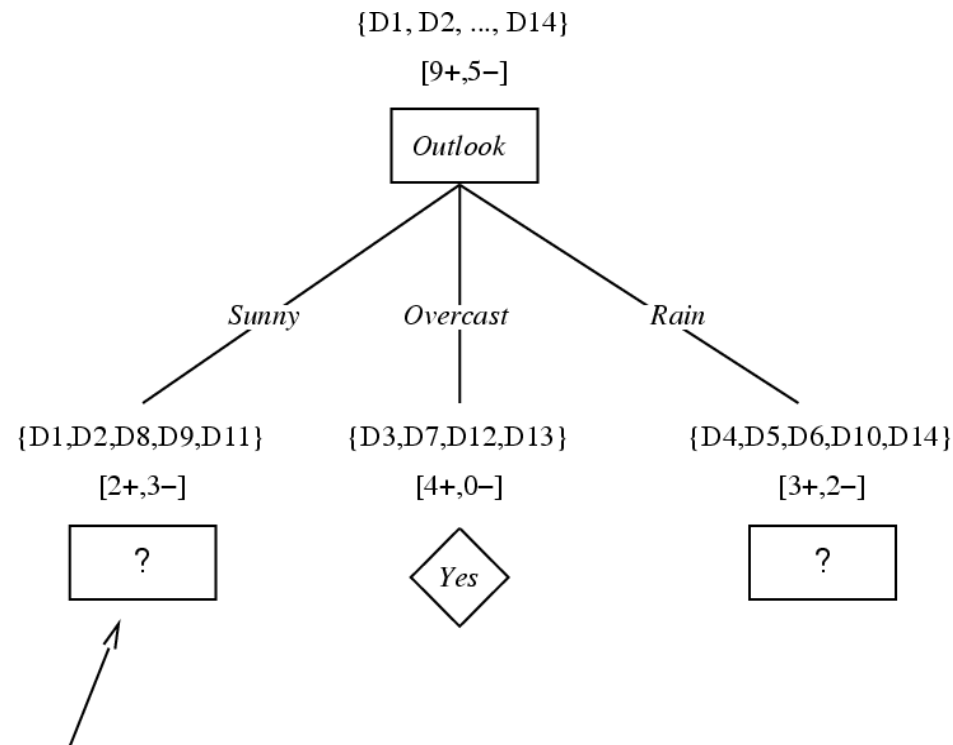
Which attribute should be tested here?

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
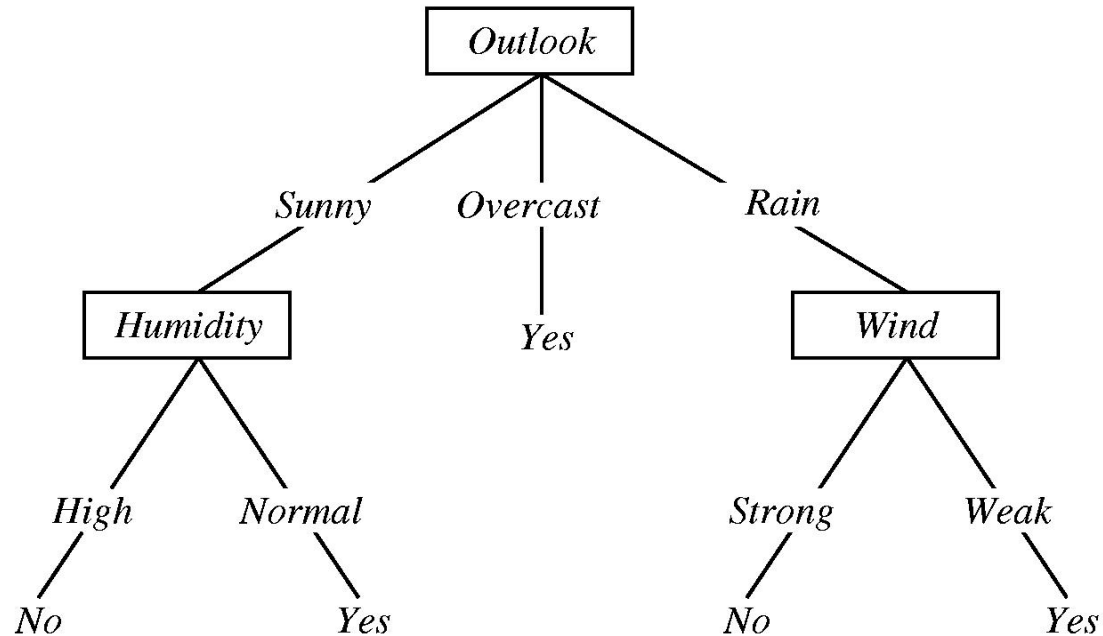
Which Tree Should We Output?

- ID3 performs heuristic search through space of decision trees

- It stops at smallest acceptable tree. Why?
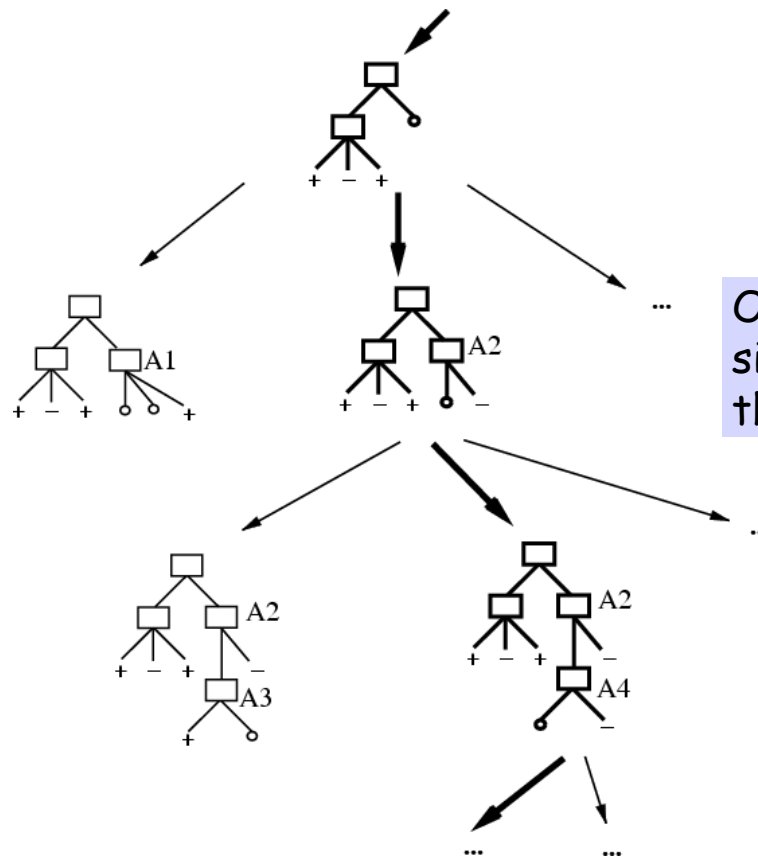
Occam's razor: prefer the simplest hypothesis that fits the data

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: $error_{\mathcal{D}}(h)$
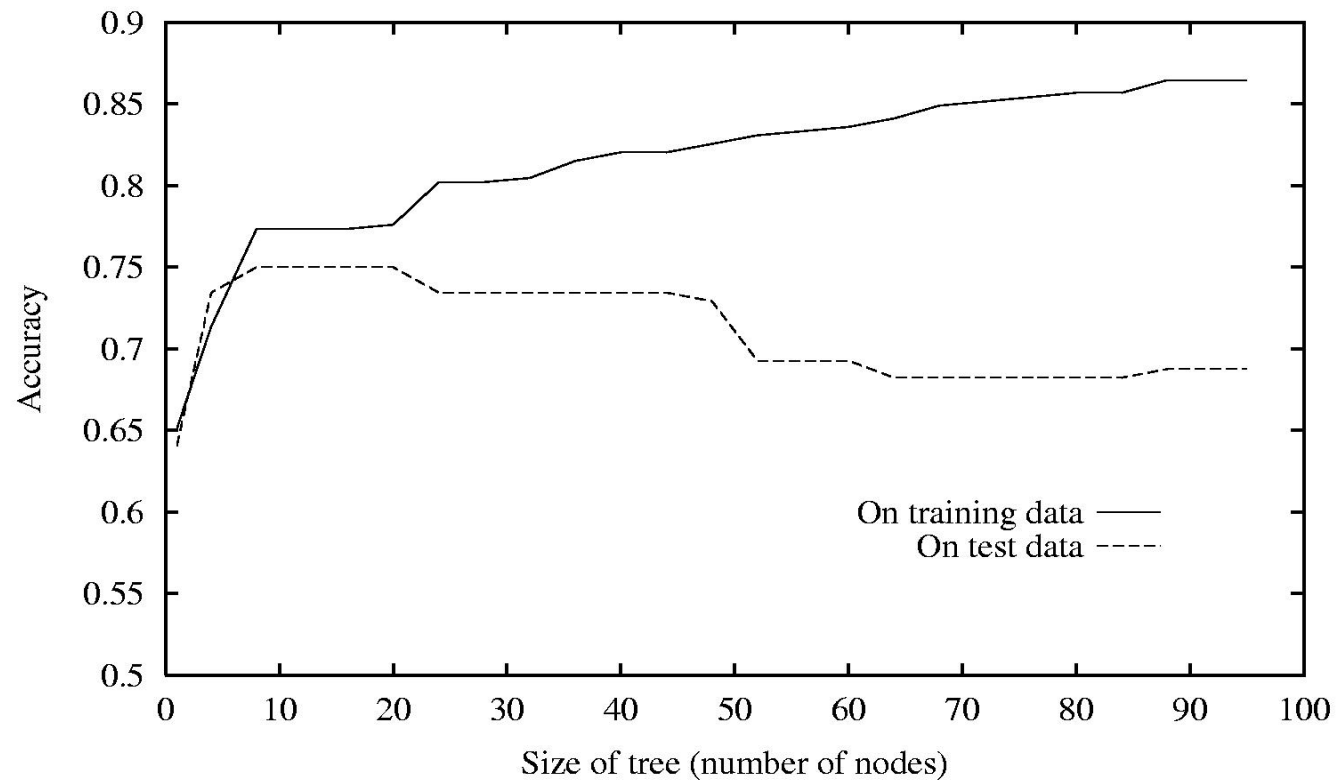
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

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

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

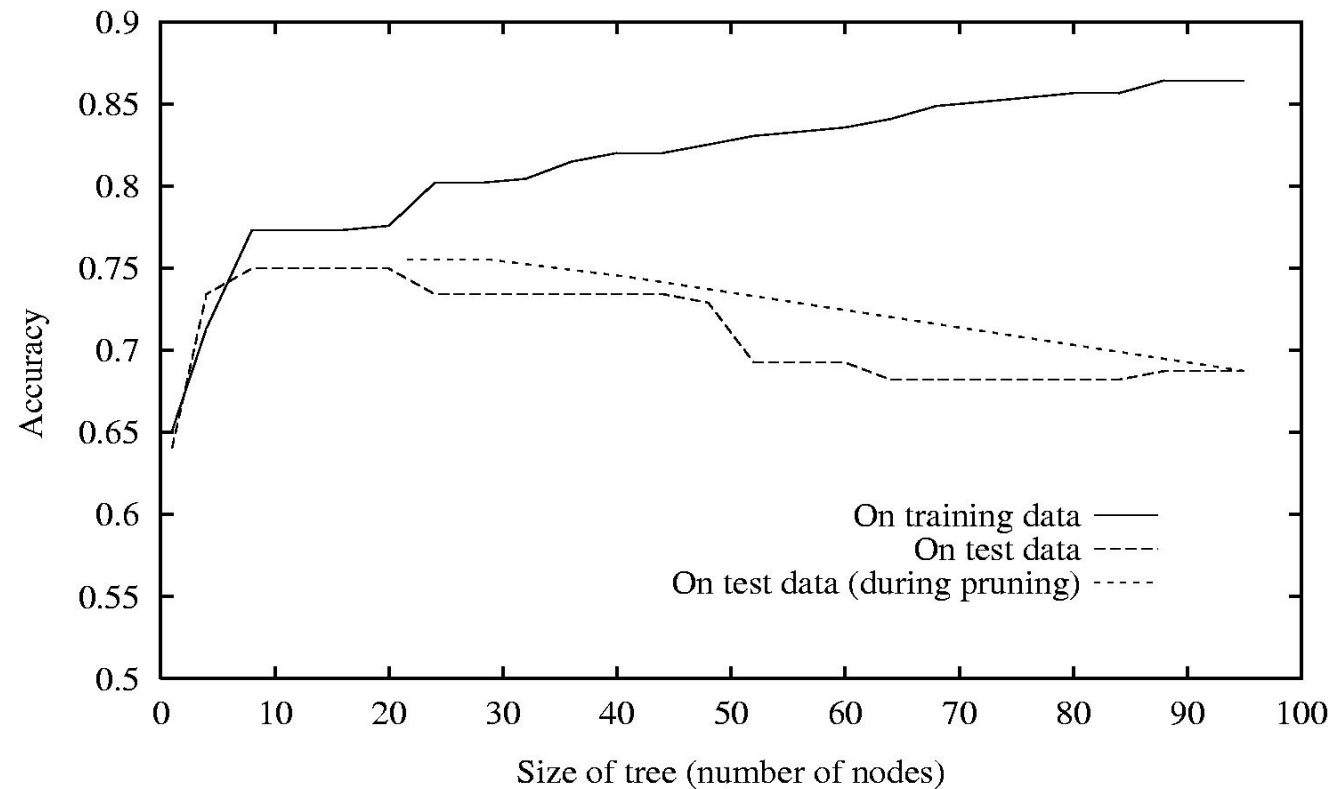- Add complexity penalty to performance measure

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)

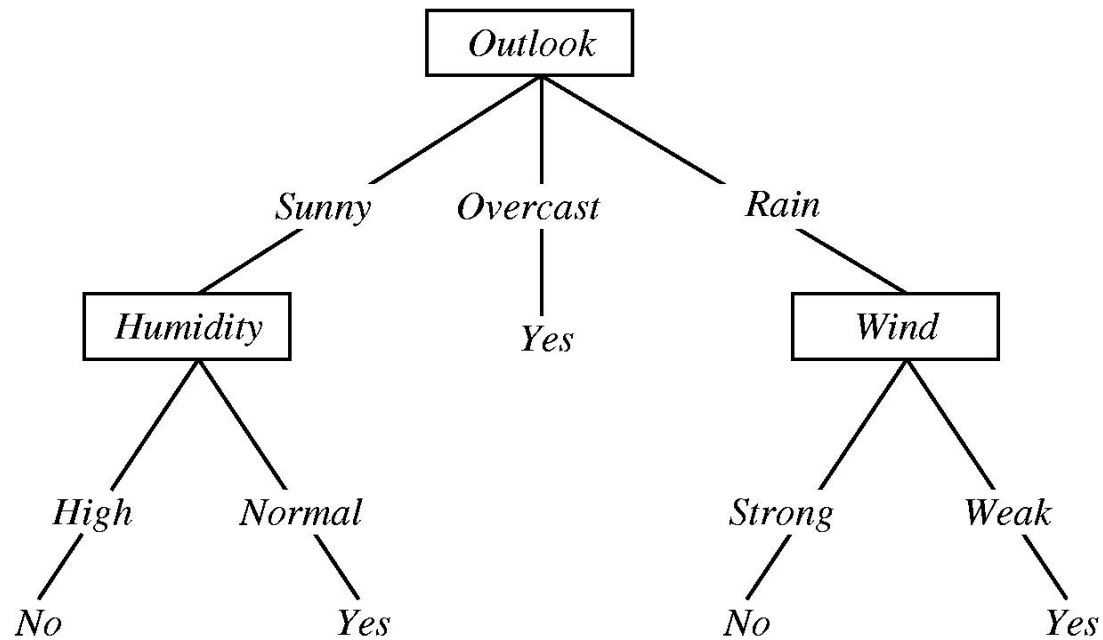2. Greedily remove the one that most improves *validation* set accuracy

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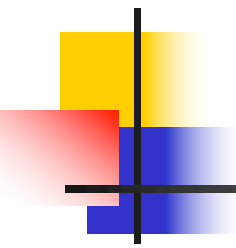
THEN $PlayTennis = Yes$

...

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
Play Tennis:	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

What you should know:

- Decision trees are one of the most popular data mining tools:
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap (to solve heuristically)
- Information gain to select attributes
- Presented for classification, can be used for regression too
- Greedy top-down learning of decision trees (ID3, C4.5, ...)
- Decision trees will overfit!!!:
  - Must use tricks to find "simple trees", e.g
  - Fixed depth/Early stopping, Pruning