Machine Learning 10-601/10-301

Matt Gormley and Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

February 3, 2021

Today:
- Decision tree learning
- Overfitting

Suggested Reading:
- Mitchell 3
- Bishop 14.4

Course Webpage: http://mlcourse.org

Decision tree learning:

One example of function approximation

Function approximation

Problem Setting:

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

superscript: i^{th} training example

Input:

- Training examples $\{<x^{(i)}, y^{(i)}>\}$ of unknown target function f that is $y^{(i)} = f(x^{(i)})$

Output:

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

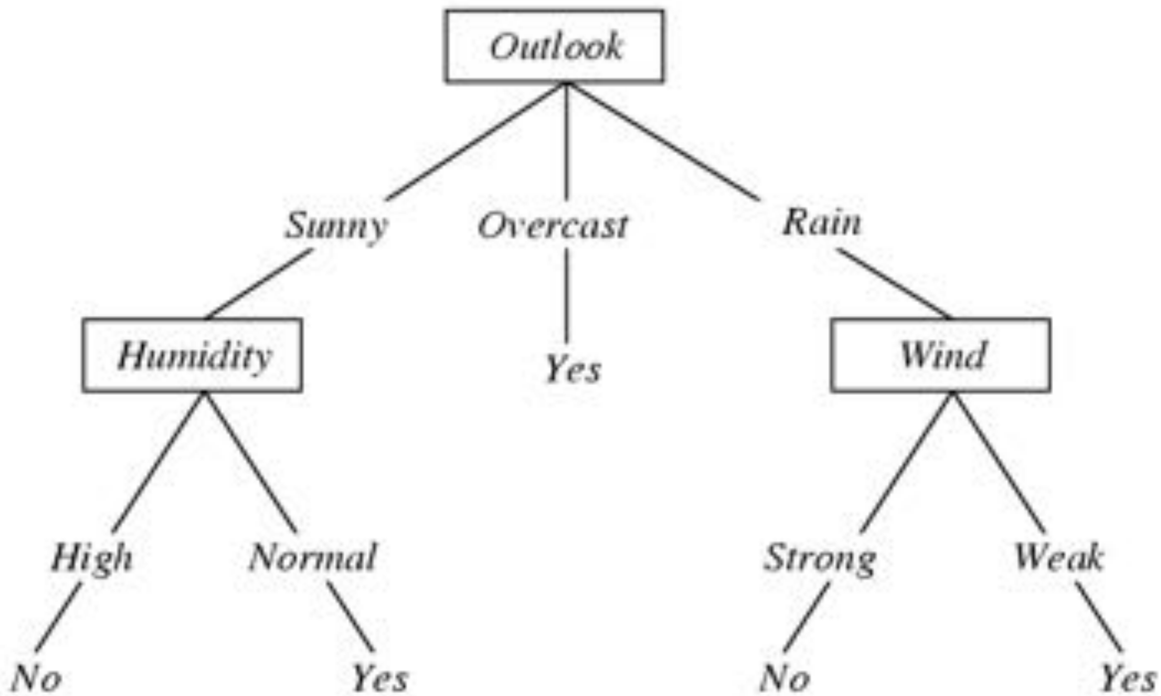
Simple Training Data Set
Learn to predict PlayTennis?

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

A Decision tree for

f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?

$< \; X_1 \qquad X_2 \qquad\qquad X_3 \qquad X_4 \; > \rightarrow \qquad Y$

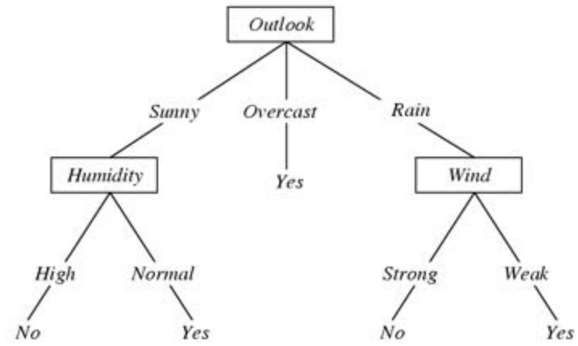


Each internal node: test one discrete-valued attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or $P(Y \mid X \in \text{leaf})$)

Decision Tree Learning



Problem Setting:

- Set of possible instances X
  - each instance x in X is vector of discrete-valued features $x = < x_1, x_2 \dots x_n >$
- 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

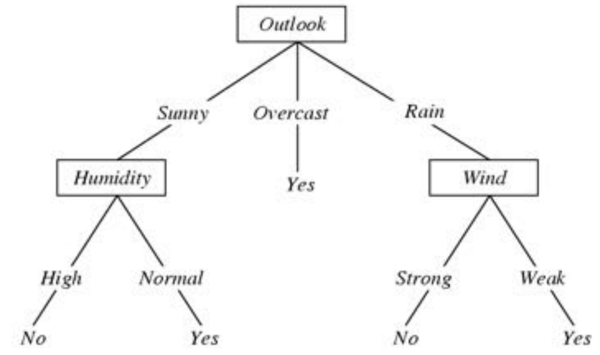
Input:

- Training examples $\{<x^{(i)},y^{(i)}>\}$ of unknown target function f

Output:

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

Decision Trees



Suppose $X = <X_1, \ldots X_n>$

where X_i are boolean-valued variables

How would you represent $Y = X_2 X_5$? $Y = X_2 \vee X_5$

How would you represent $X_2 X_5 \vee X_3 X_4 (\neg X_1)$

A Tree to Predict C-Section Risk

[Sims et al., 2000]

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Top-Down Induction of Decision Trees

[ID3, C4.5, Quinlan]

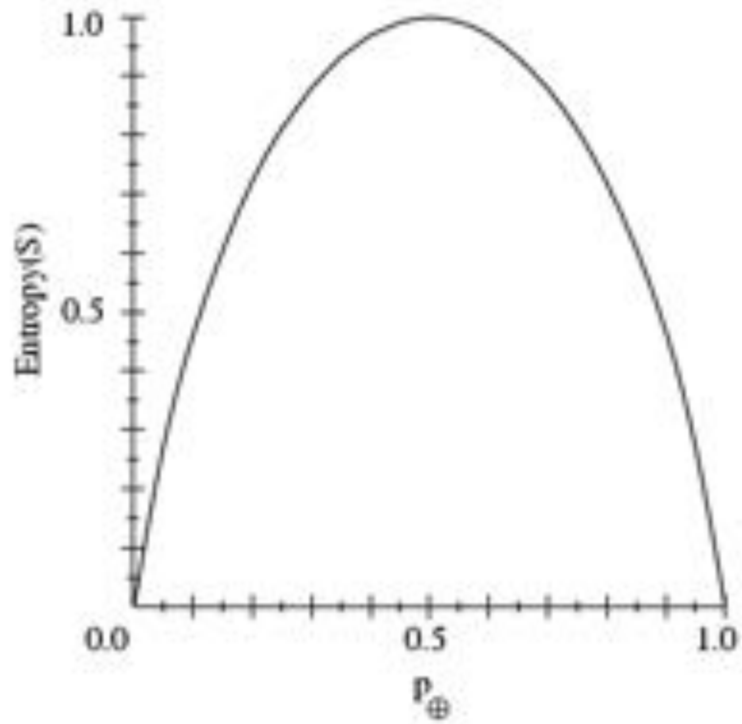
node = Root

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?

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

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

Entropy

of possible values for X

Entropy $H(X)$ of a random variable 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 possible 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:

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

Entropy

of possible values for X

Entropy *H(X)* of a random variable *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)

Recall definition:
expected value $E_{P(X)}[f(X)]$ of $f(X)$ with respect to $P(X)$

$$E_{P(X)}[f(X)] = \sum_{\text{possible values of } X} P(X=i)f(i)$$

=i) bits

m *X* is:

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

Entropy

Entropy *H(X)* of a random variable *X*

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

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* :

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

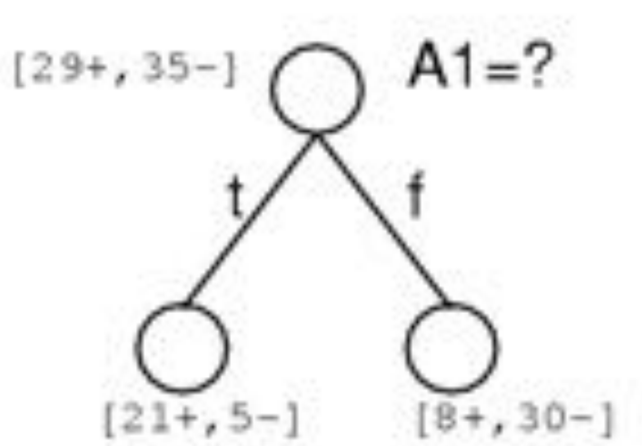
Also called Mutual Information between A,Y

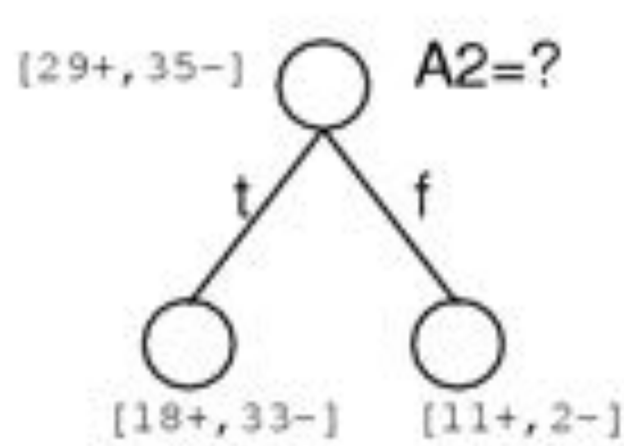
Information Gain, $I_S(A,Y)$, is the reduction in entropy of target variable Y for data sample S, due to sorting on variable A

Entropy of Y in sample S

Entropy of Y in sample S after sorting on value of A

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



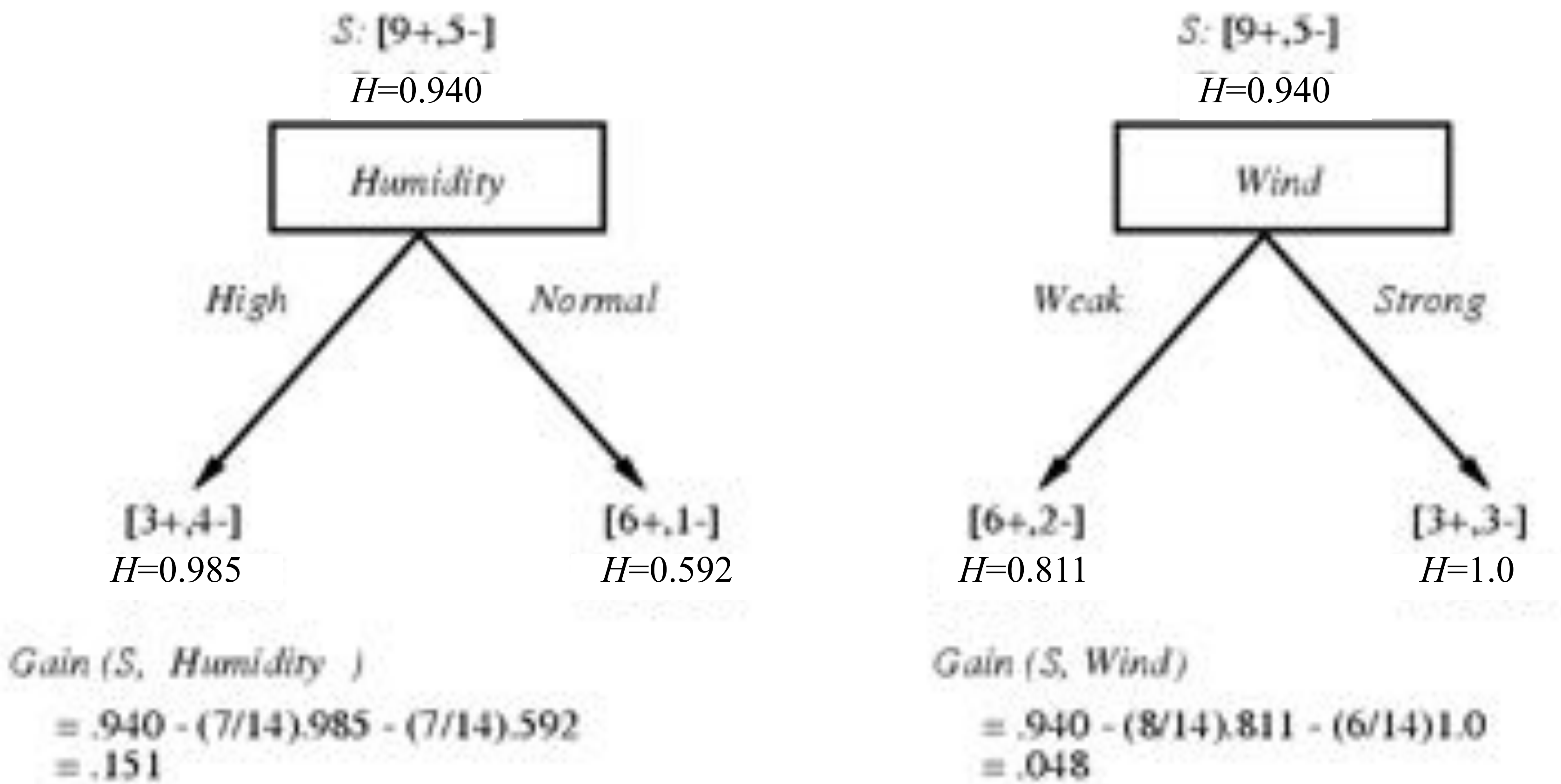


Simple Training Data Set

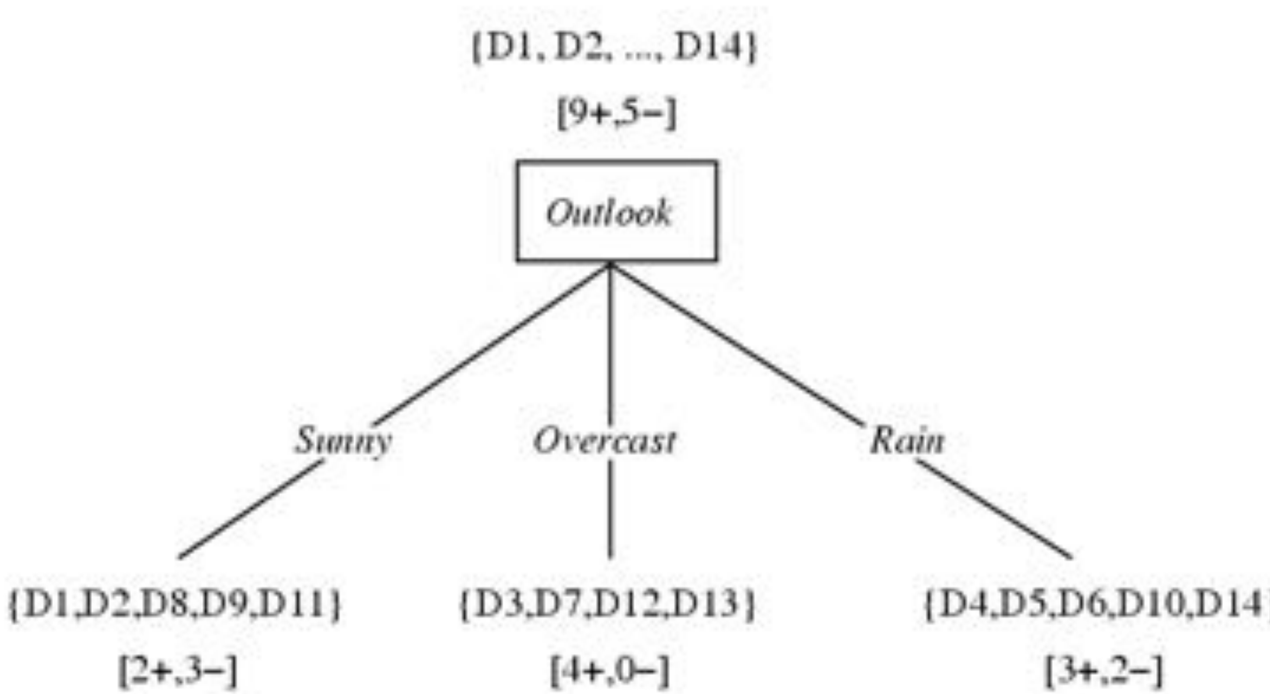
Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

Selecting the Next Attribute

Which attribute is the best classifier?

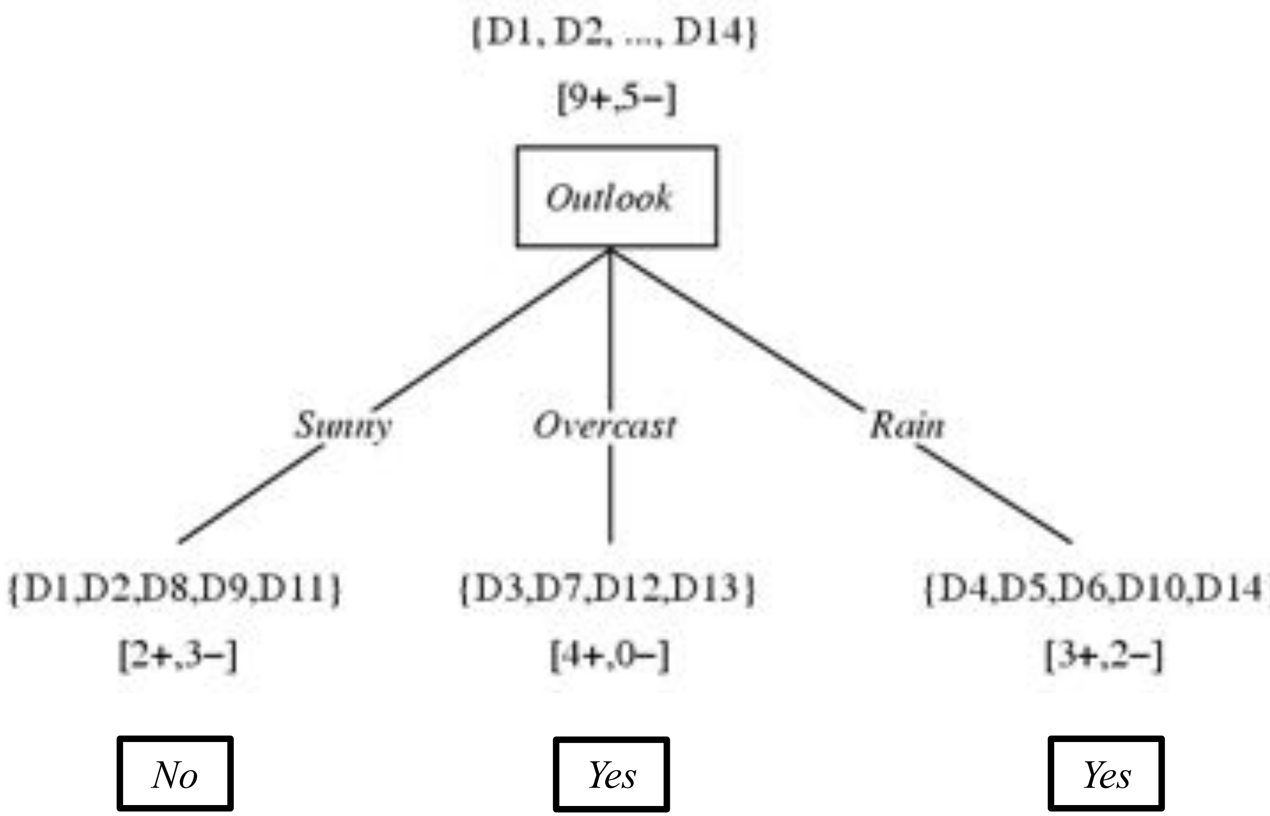


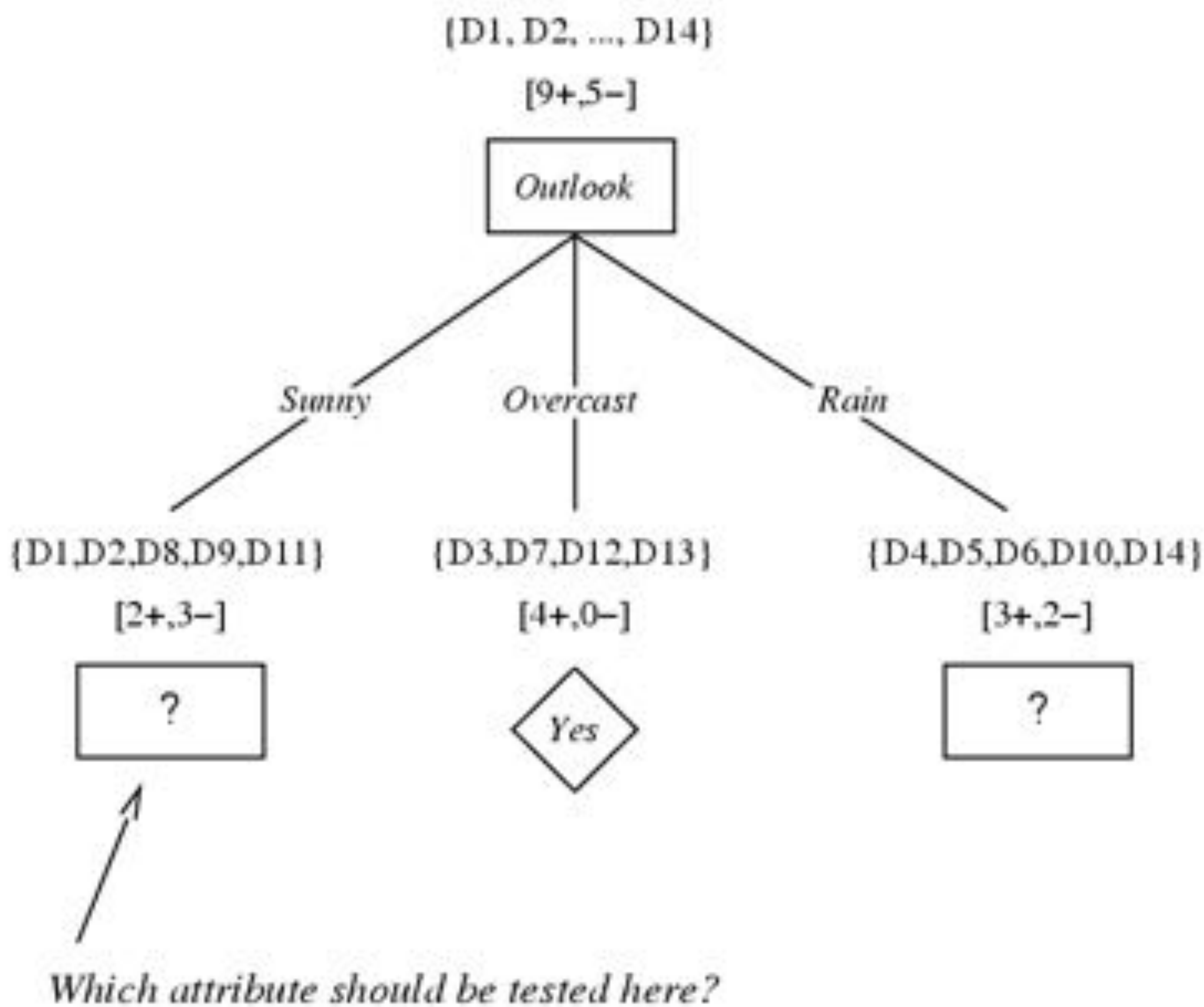
The Best Attribute is *Outlook*



Decision Stumps

A decision stump is simply a decision tree with depth 1:





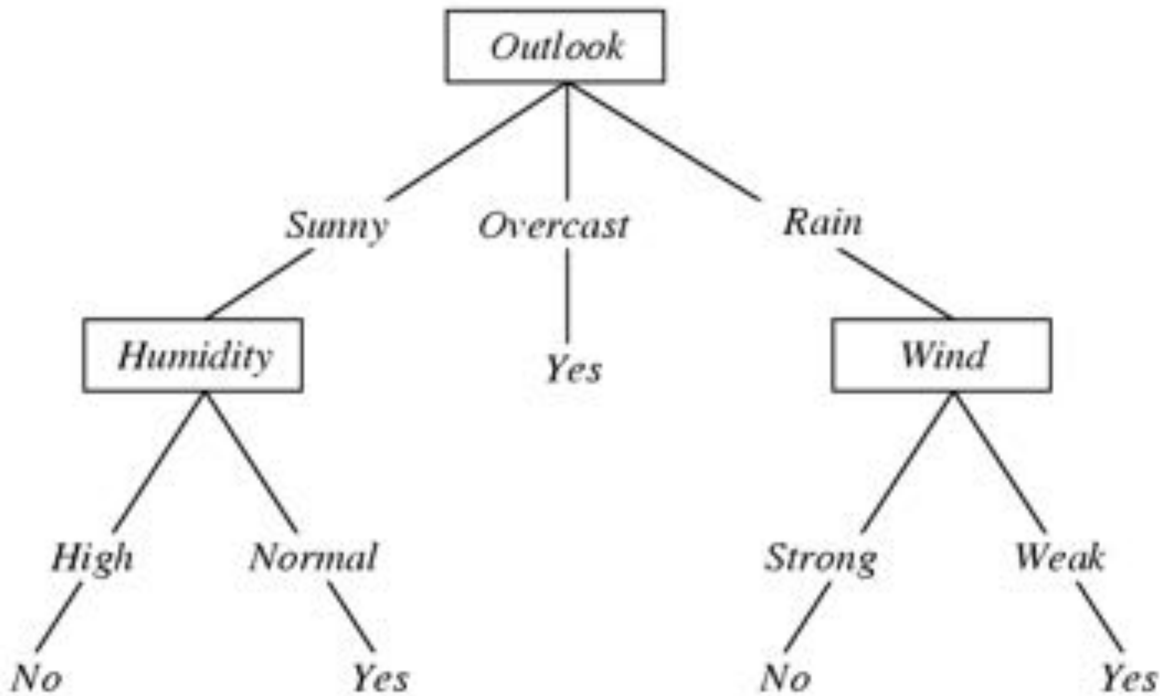
S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\,(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain\,(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain\,(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Final Decision Tree for
f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?

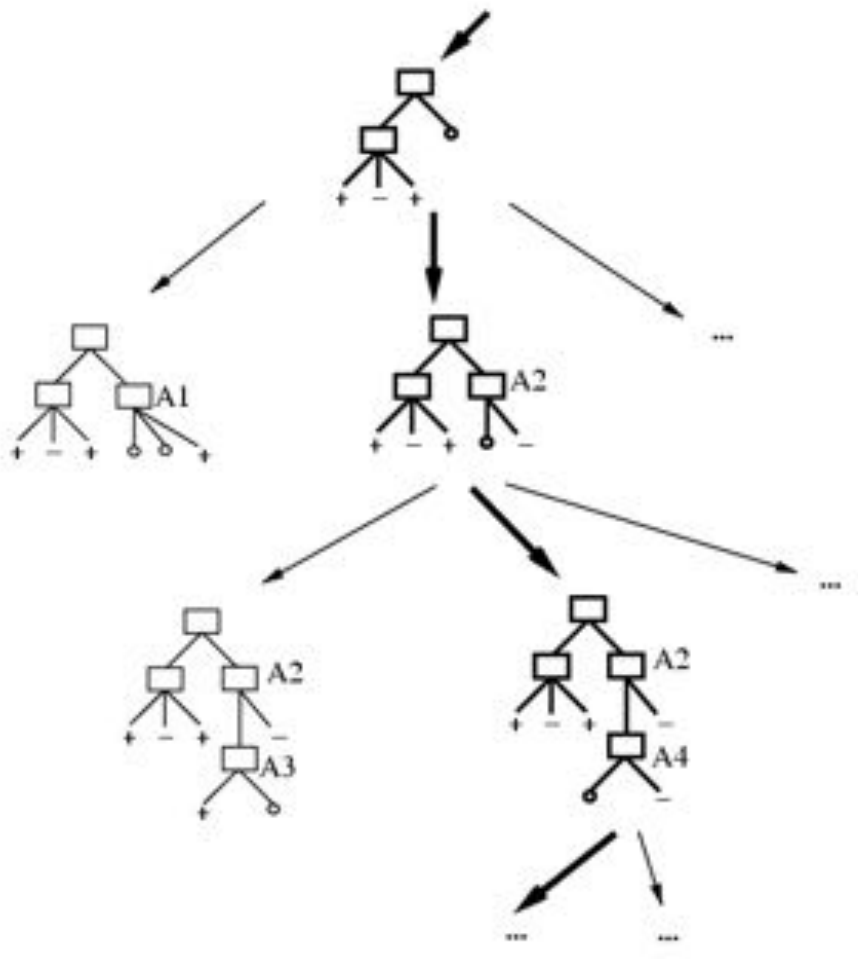


Each internal node: test one discrete-valued attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y

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

Why Prefer Short Hypotheses? (Occam's Razor)

Arguments in favor:

Arguments opposed:

Why Prefer Short Hypotheses? (Occam's Razor)

Argument in favor:

- Fewer short hypotheses than long ones

→ a short hypothesis that fits the data is less likely to be a statistical coincidence

→ highly probable that *<u>some</u>* sufficiently complex hypothesis will fit the data

Argument opposed:

- Also fewer hypotheses with prime number of nodes and attributes beginning with "Z"
- What's so special about "short" hypotheses?

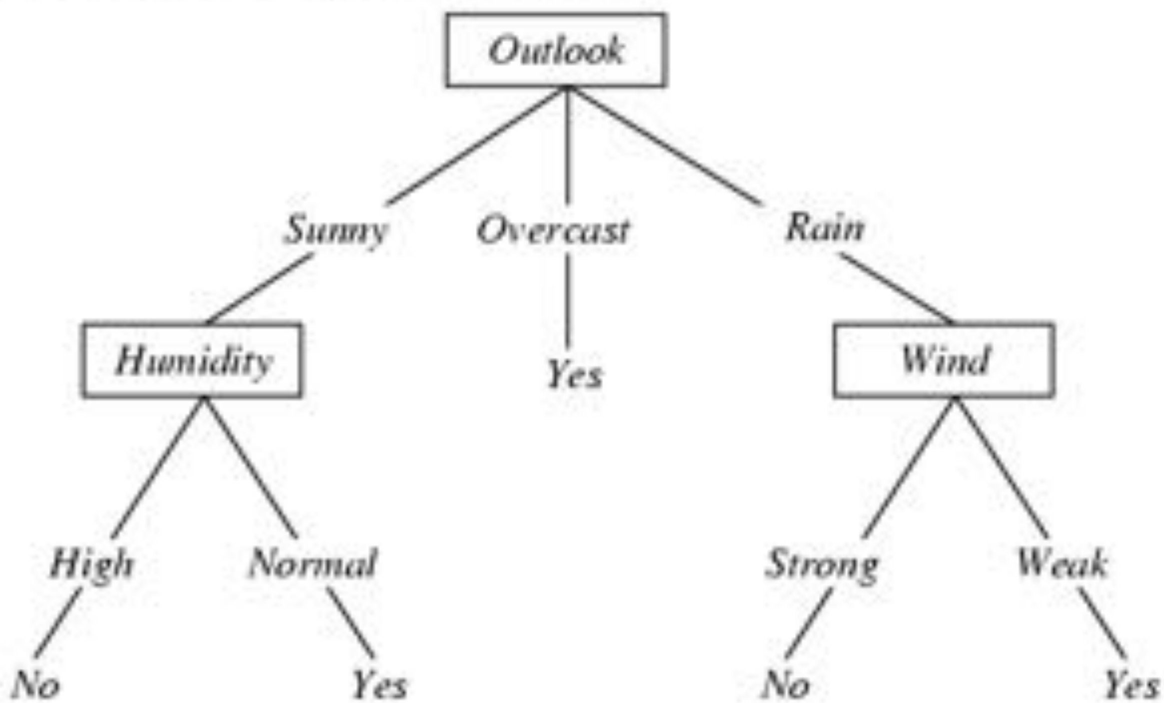
Overfitting in Decision Trees

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

Consider adding noisy training example #15:

$$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$$

What effect on earlier tree?



Overfitting

Consider a hypothesis h and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

Overfitting

Consider a hypothesis h and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

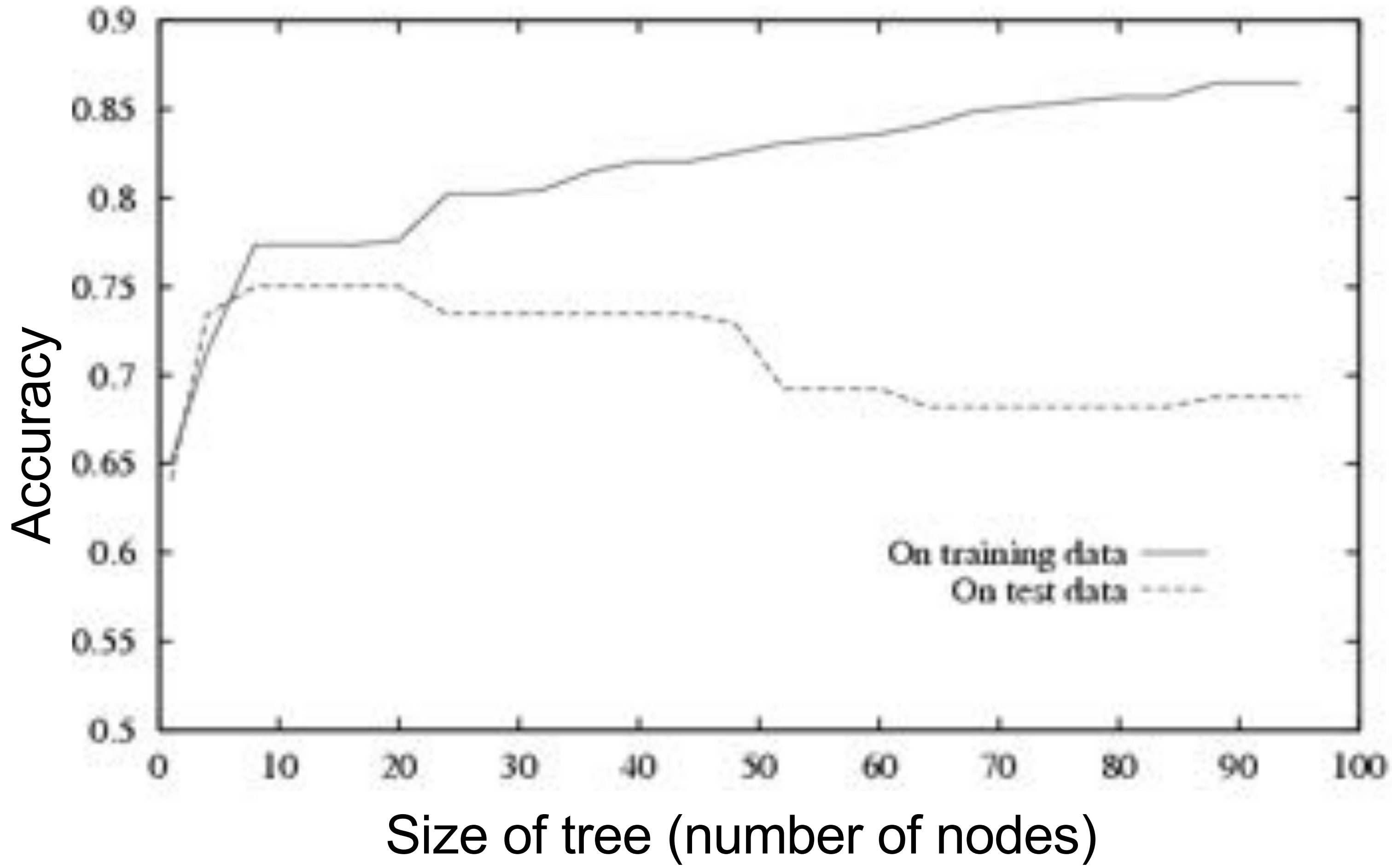
We say h overfits the training data if

$$error_{true}(h) > error_{train}(h)$$

Amount of overfitting =

$$error_{true}(h) - error_{train}(h)$$

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

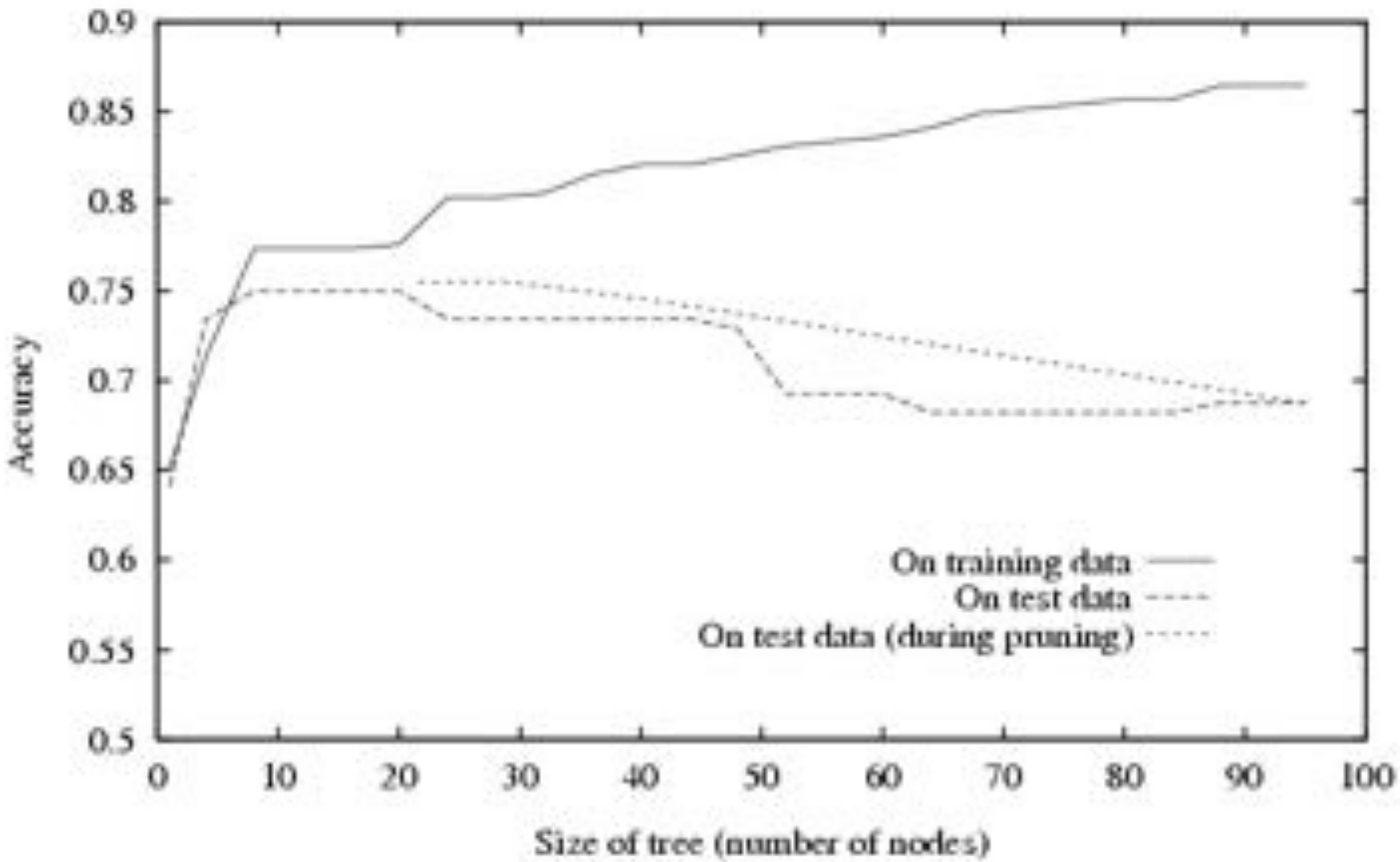
Reduced Error Pruning

- Split data into *training set* and *validation set*
- Train a tree to classify *training set* as well as possible

- Do until further pruning reduces *validation set* accuracy:
  1. For each internal tree node, consider making it a leaf node (pruning the tree below it)
  2. Greedily chose the above pruning step that best improves error over *validation set*

Produces smallest version of the most accurate pruned tree

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)

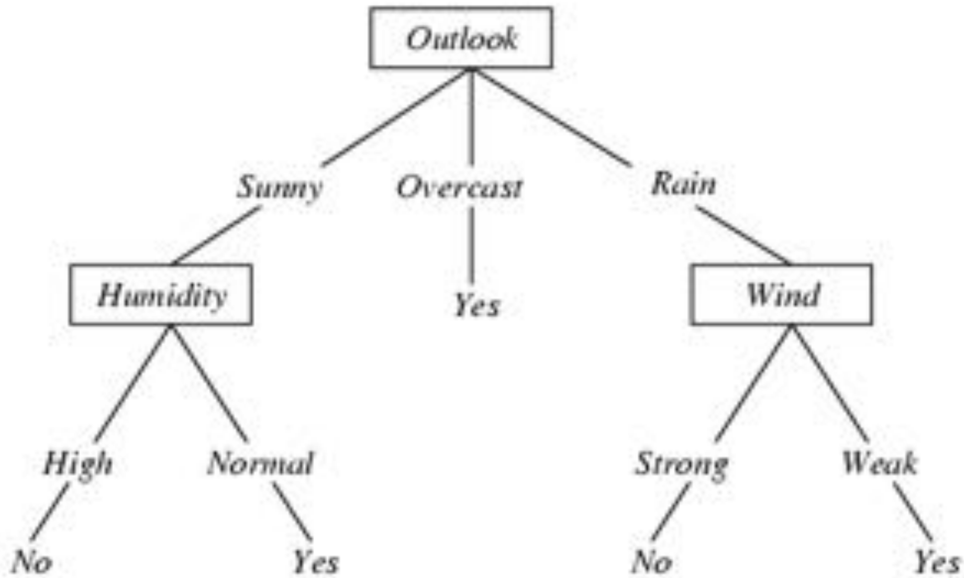
Continuous Valued Attributes

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

Converting A Tree to Rules



Decision Forests

Key idea:

1. learn a collection of many trees
2. classify by taking a weighted vote of the trees

Empirically successful. Widely used in industry.

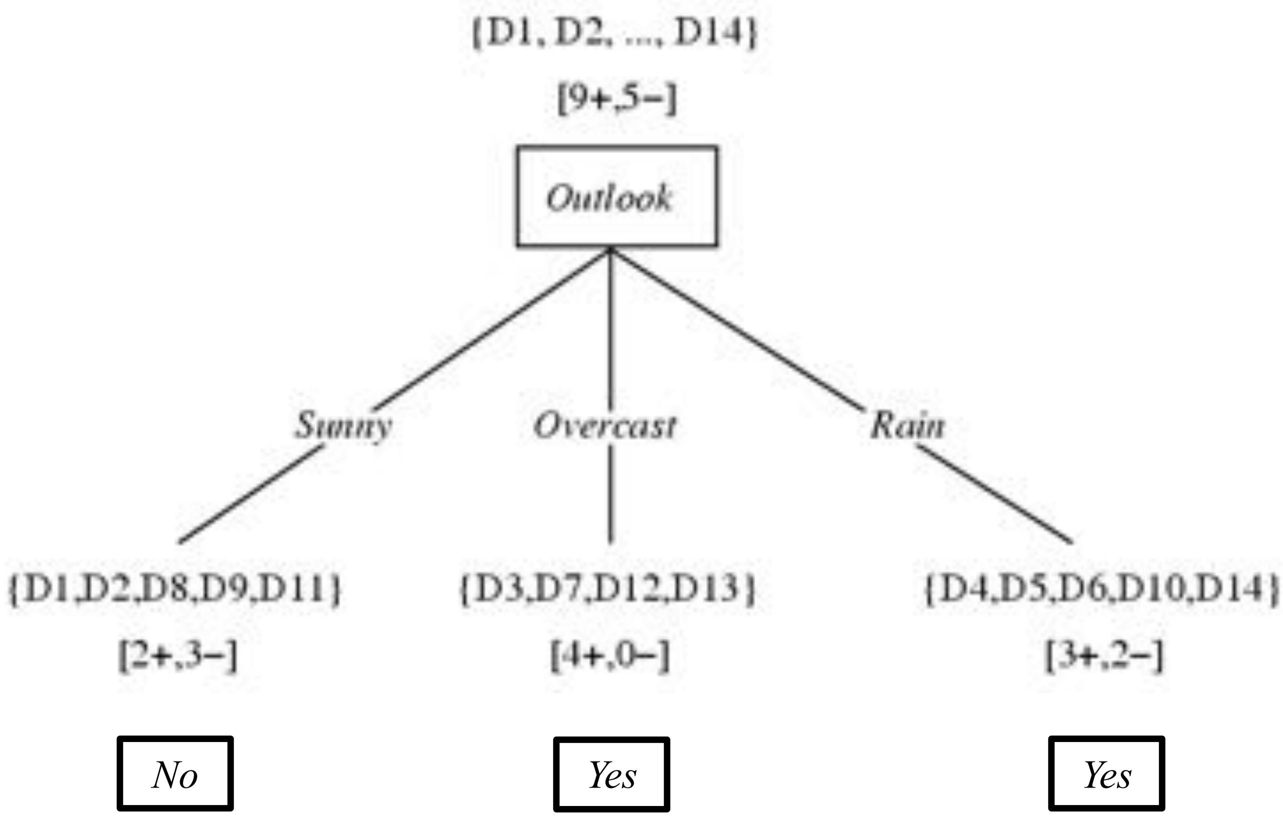
- human pose recognition in Microsoft kinect
- medical imaging – cortical parcellation
- classify disease from gene expression data

How to train different trees

1. Train on different random subsets of data
2. Randomize the choice of decision nodes

Decision Forests often use ensemble of Decision Stumps

A decision stump is simply a decision tree with depth 1:



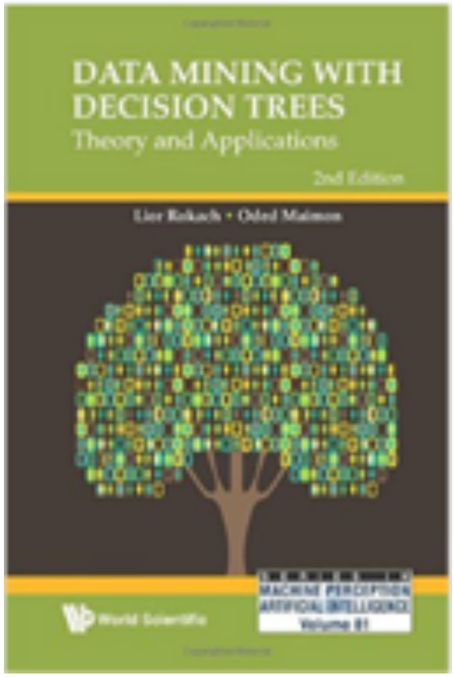
You should know:

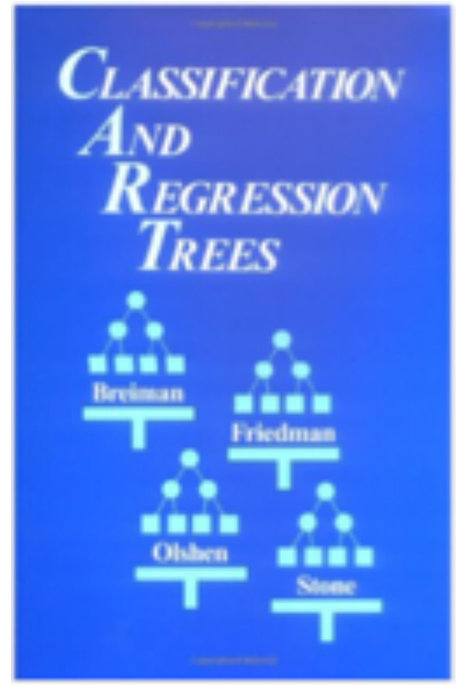
- Well posed function approximation problems:
  - Instance space, X
  - Sample of labeled training data { <$x^{(i)}$, $y^{(i)}$>}
  - Hypothesis space, H = { f: X→Y }

- Learning is a search/optimization problem over H
  - Various objective functions to define the goal
    - minimize training error (0-1 loss)
    - minimize validation error (0-1 loss)
    - among hypotheses that minimize error, select smallest (?)

- Decision tree learning
  - Greedy top-down learning of decision trees (ID3, C4.5, ...)
  - Overfitting and post-pruning
  - Extensions… to continuous values, probabilistic classification
  - Widely used commercially: decision *forests*

Further Reading…





Questions to think about (0)

- How can we use decision trees to make probabilistic predictions (ie., $P(Y=1|X)$ instead of simply predict that $Y=1$ or $Y=0$?

[Hint: go back and look at the tree for predicting C-section birth risk]

Questions to think about

- Consider target function f: $<x_1,x_2> \rightarrow y$, where x_1 and x_2 are real-valued, y is Boolean (0 or 1)
  - What is the set of decision surfaces describable with decision trees that use each attribute at most once?

Questions to think about (2)

- ID3 and C4.5 are heuristic algorithms that search through the space of decision trees. Why not just do an exhaustive search over all possible trees?

Questions to think about (3)

- Why use Information Gain to select attributes in decision trees? What other criteria seem reasonable, and what are the tradeoffs in making this choice?