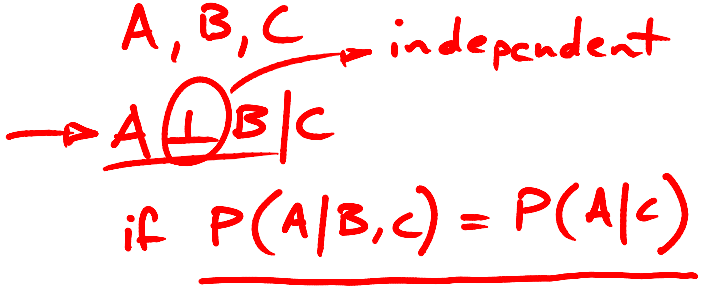
Decision Trees

A divide-and-conquer approach to classification problems

Matt Eicholtz
Department of Mechanical Engineering
Carnegie Mellon University
meicholt@andrew.cmu.edu



Can you spot the pattern?

PATTERNS:

✓ IF $X_3 = 1, Y = 1$ ①

✗ $(X_1 \text{ XOR } X_2) \text{ OR } X_3$

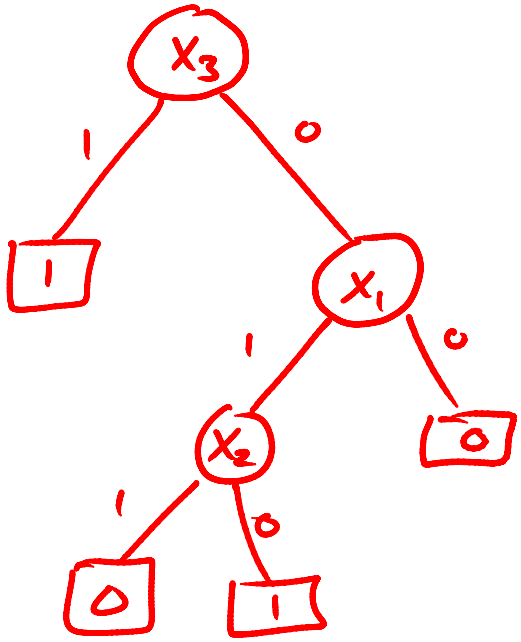
✓ $X_3 \text{ OR } (X_1, \sim X_2)$

$\sim X_3, \sim X_2 \rightarrow Y = 1$

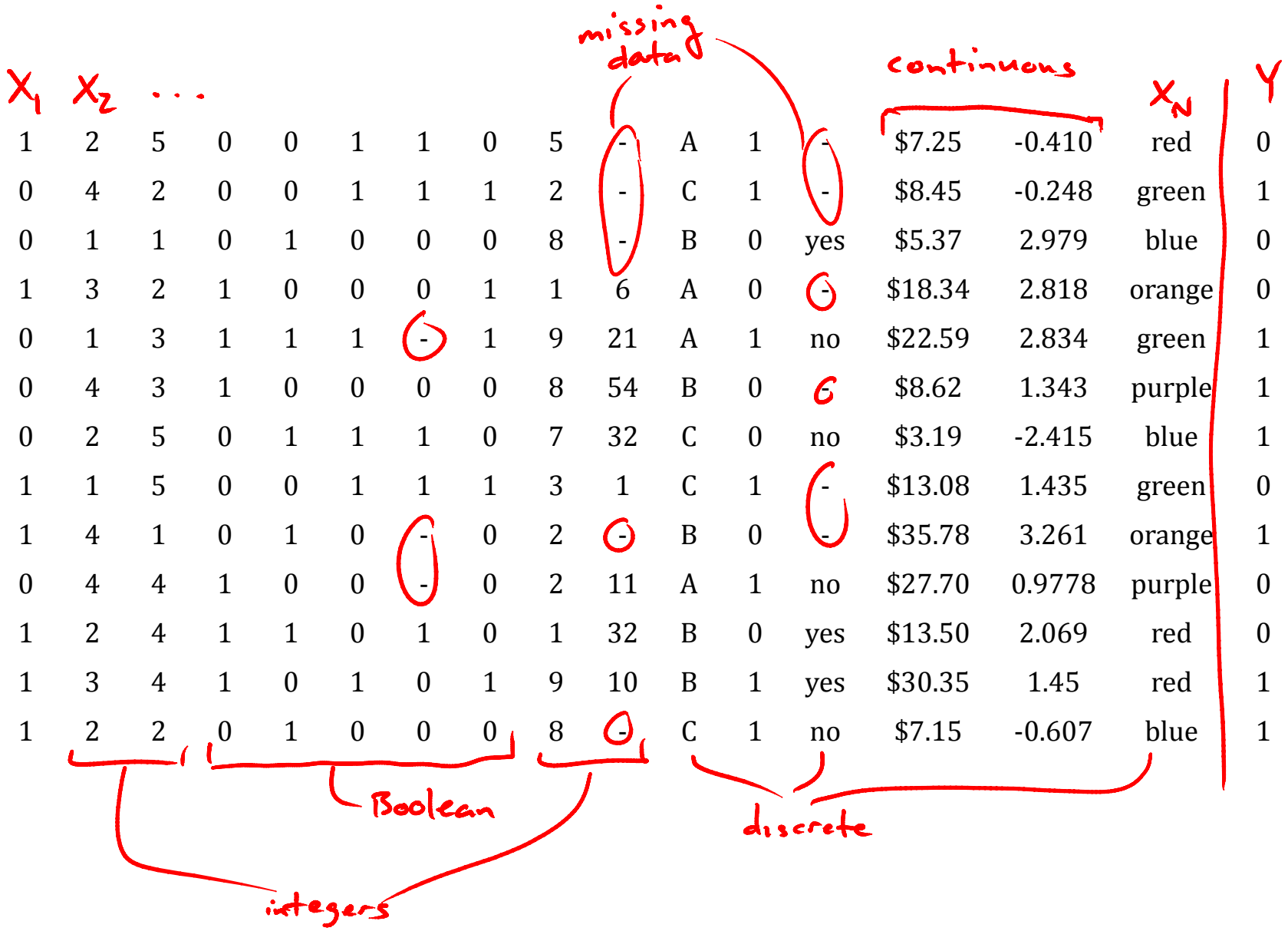
Attributes

output (class)

X_1	X_2	X_3	Y	
0	0	0	0	
0	0	1	1	✓
0	1	0	0	← instance
0	1	1	1	✓
1	0	0	1	
1	0	1	1	✓
1	1	0	0	
1	1	1	1	✓



How about now?



X_1	X_2													X_N	Y
1	2	5	0	0	1	1	0	5	-	A	1	-	$7.25	-0.410	red	0
0	4	2	0	0	1	1	1	2	-	C	1	-	$8.45	-0.248	green	1
0	1	1	0	1	0	0	0	8	-	B	0	yes	$5.37	2.979	blue	0
1	3	2	1	0	0	0	1	1	6	A	0	-	$18.34	2.818	orange	0
0	1	3	1	1	1	-	1	9	21	A	1	no	$22.59	2.834	green	1
0	4	3	1	0	0	0	0	8	54	B	0	C	$8.62	1.343	purple	1
0	2	5	0	1	1	1	0	7	32	C	0	no	$3.19	-2.415	blue	1
1	1	5	0	0	1	1	1	3	1	C	1	-	$13.08	1.435	green	0
1	4	1	0	1	0	-	0	2	-	B	0	-	$35.78	3.261	orange	1
0	4	4	1	0	0	-	0	2	11	A	1	no	$27.70	0.9778	purple	0
1	2	4	1	1	0	1	0	1	32	B	0	yes	$13.50	2.069	red	0
1	3	4	1	0	1	0	1	9	10	B	1	yes	$30.35	1.45	red	1
1	2	2	0	1	0	0	0	8	-	C	1	no	$7.15	-0.607	blue	1

Definitions

The task of **supervised learning** is this:

Given a training set of N example input-output pairs

$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$

where each y_j was generated by an unknown function $y = f(x)$,

discover a function h that approximates the true function f.

A hypothesis is **consistent** if it agrees with the data.

A hypothesis **generalizes** well if it predicts the correct output on novel examples.

A learning problem is **realizable** if the hypothesis space contains the true function.

Learning problems: classification v. regression

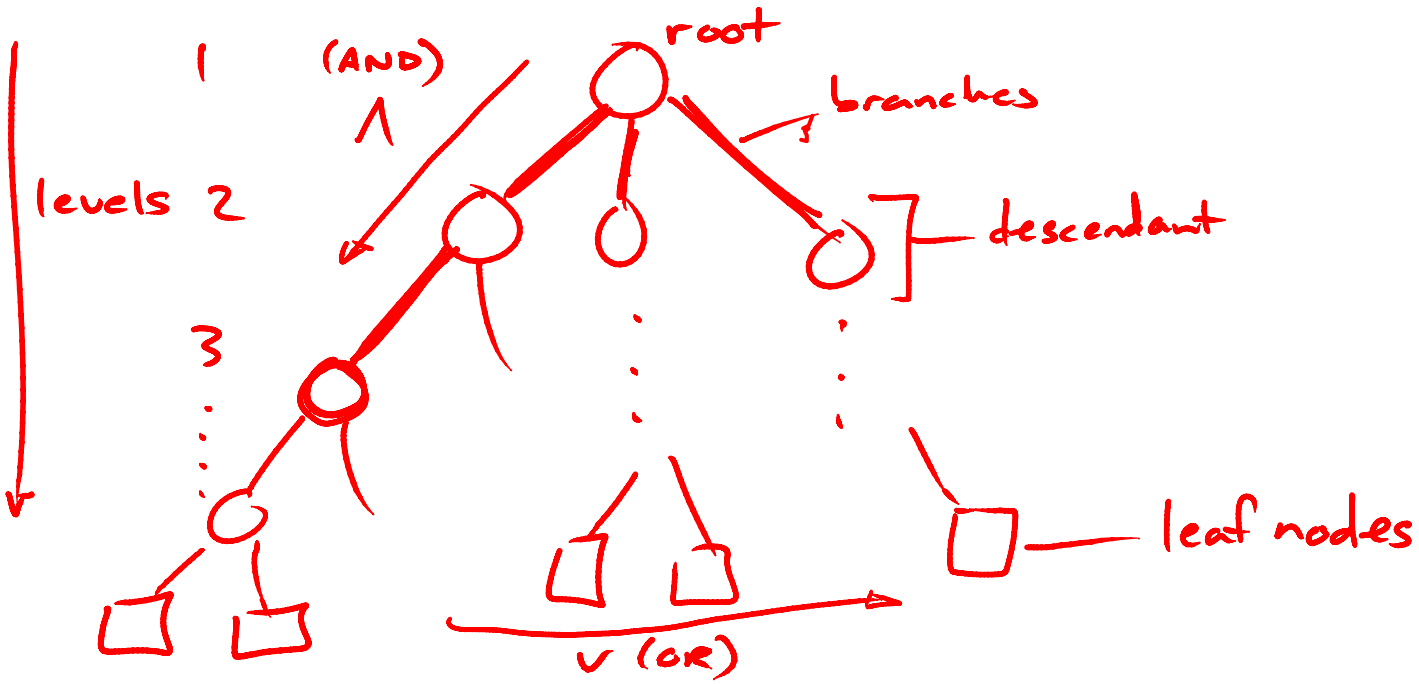
(Russell and Norvig pp. 695-697)

Definitions

A **decision tree** is a tree-structured plan of a set of attributes to test in order to predict the output.

The tree contains **nodes** connected by **branches**.

The top node is the **root**, which is linked to **descendants**, until **leaf nodes** are reached.



(Andrew Moore)

Questions to consider…

How do you learn a decision tree using example input-output pairs?

How do you test a decision tree on a new example?

What if some attributes have multiple values? What if they are continuous?

What if the output has multiple values? What if it is continuous?

What if data is missing?

How do you choose which attributes to split on?

When should you stop splitting the data?

How can you tell if a decision tree is good?

What if a decision tree performs well for the data it was trained on, but does not perform well on new data?

Some examples…

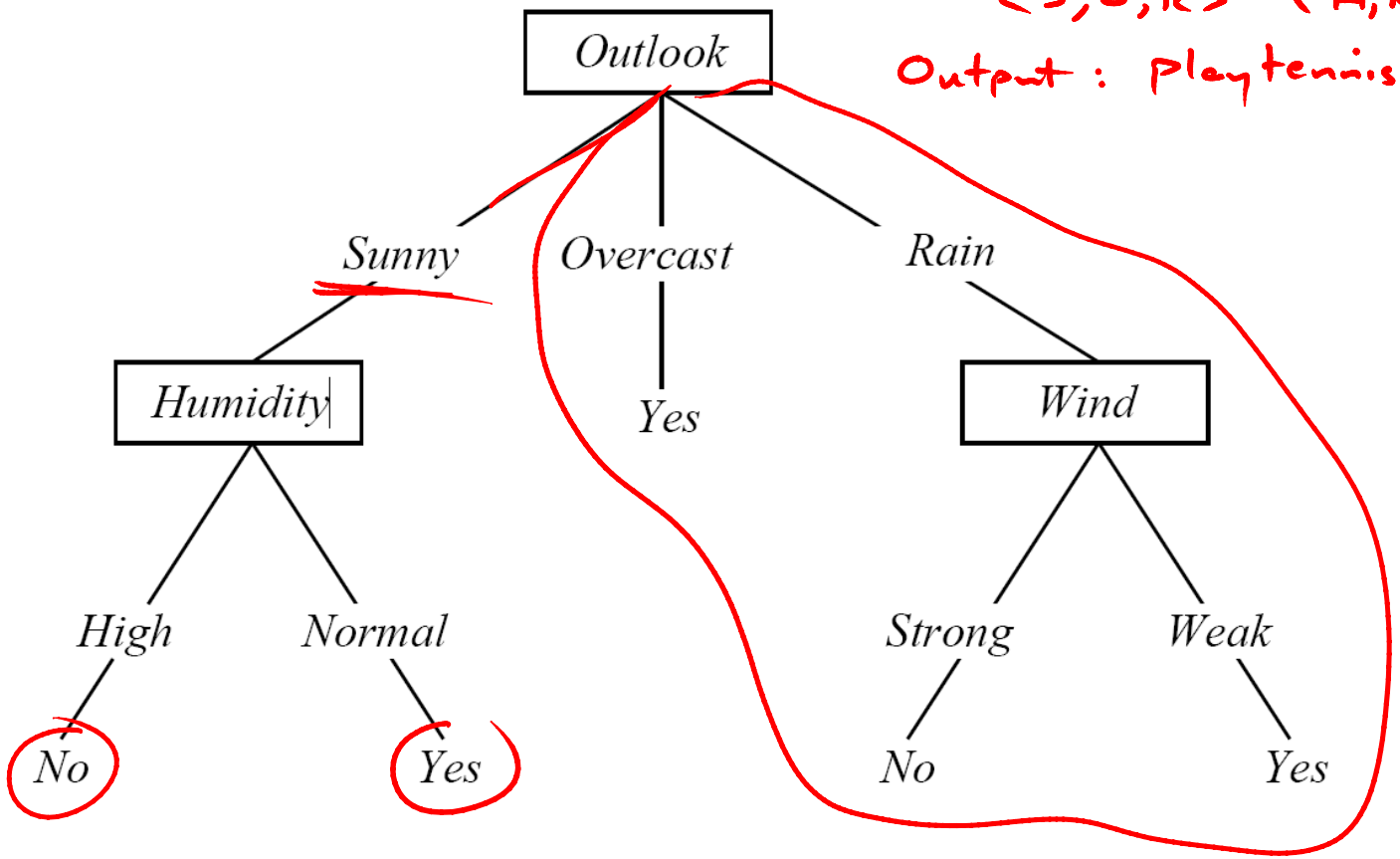
Some examples…

Should I play tennis?

Attributes: [Outlook, Humidity, Wind]

Values: <S,O,R> <H,N> <S,W>

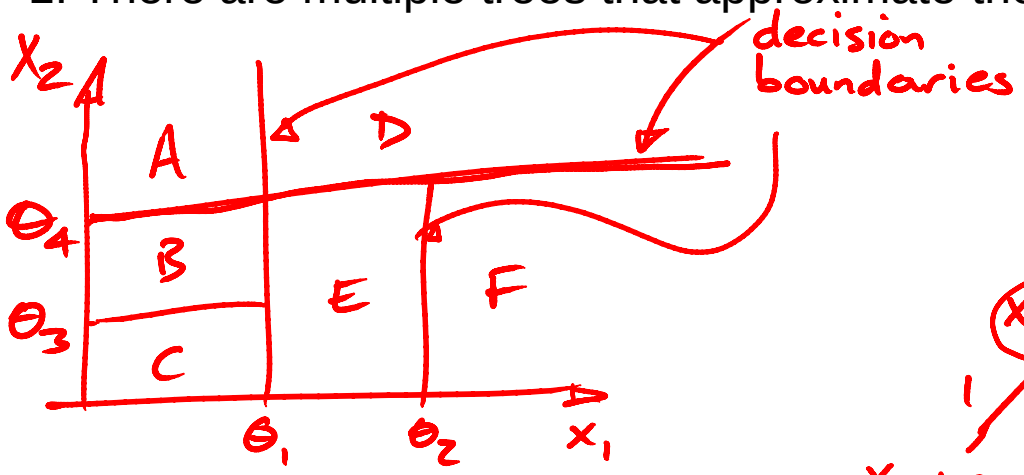
Output: playtennis?

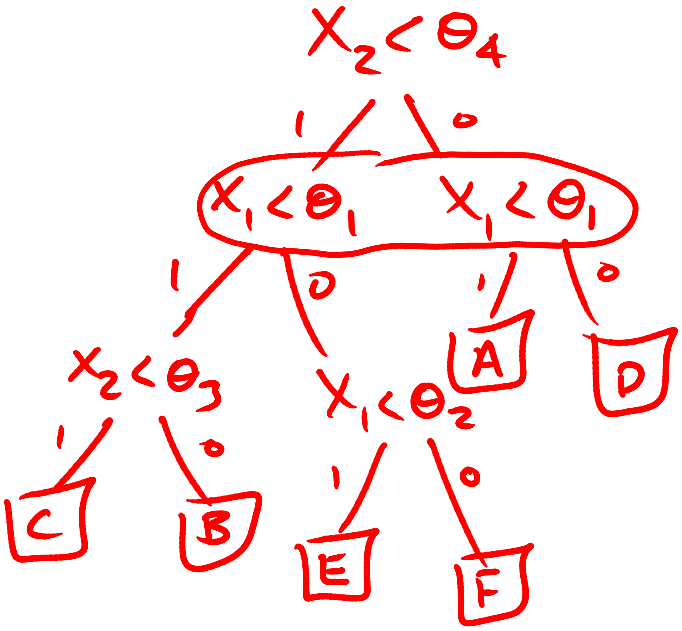


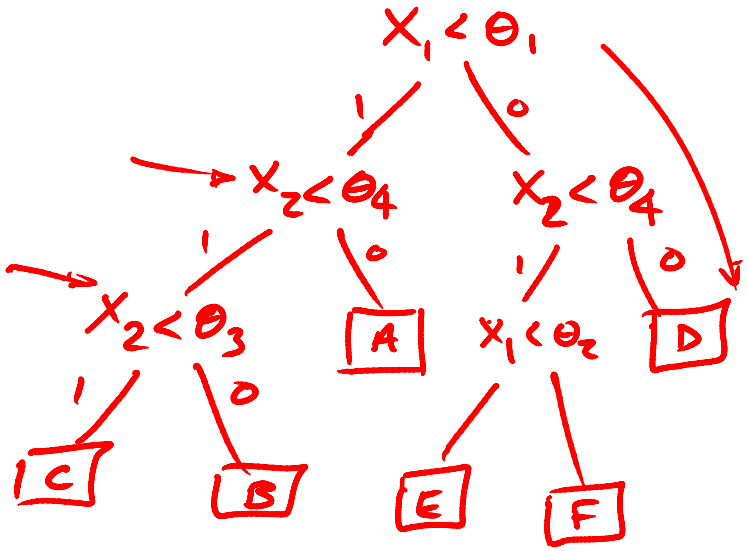
Some examples…

Interesting properties

1. There are multiple trees that approximate the same function.





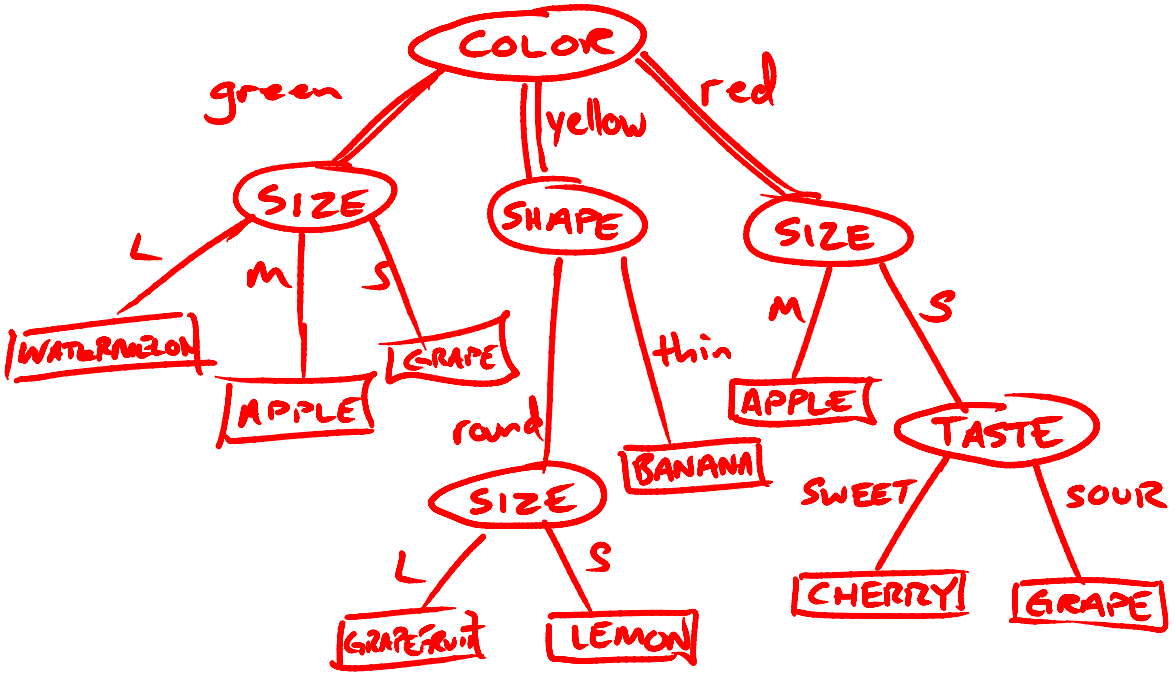


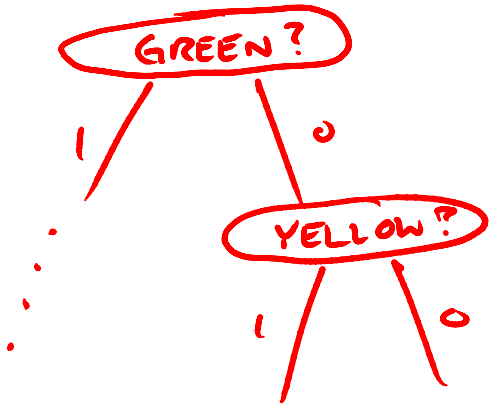
Interesting properties

2. Any decision tree has an equivalent binary decision tree.

BRANCHING FACTOR, B : # of discrete values it can take

when B=2 always ⇒ binary decision tree





Learning decision trees from data

Top-down induction

1. Determine the "best" decision attribute from those available
2. Assign that attribute to the next node
3. For each value of that attribute, create a descendant node
4. Sort training examples for each descendant
5. Recursively build tree from each descendant until stopping criteria is met
6. At each leaf node, assign the appropriate classification

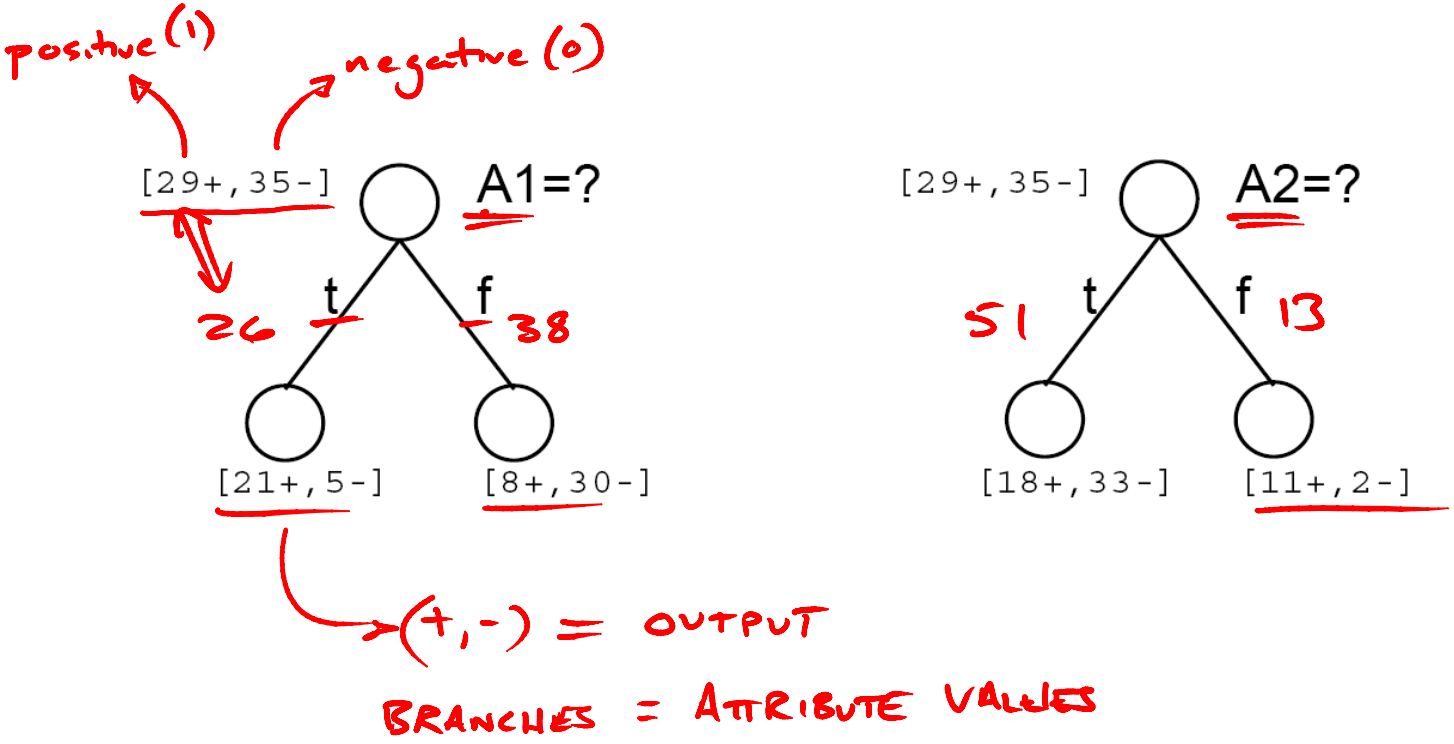
Stopping criteria

- The subset of training examples have the same output
  → assign as output @ leaf
- The subset of training examples have the same values for all input attributes
  → NOISE ; assign the plurality values
  $(x_1, x_2, \ldots x_N)_i = (-)_j$
  $y_i \neq y_j$
- There are no training examples for a particular leaf node

assign majority @ previous node

How to determine which attribute is best?

Use the **information gain**!



Entropy

Entropy is a measure of the uncertainty associated with a random variable

- Suppose you are receiving a set of independent random samples of X
- You see that X has four possible values with equal probabilities

$$P(X = A) = \frac{1}{4},\ \ P(X = B) = \frac{1}{4},\ \ P(X = C) = \frac{1}{4},\ \ P(X = D) = \frac{1}{4}$$

$$BAACBADCDADDDA \ldots$$

- You need to transmit the data over a binary serial link. You can encode each reading with two bits:

A-00 C-10
B-01 D-11

$$0100001001001110110011111100 \ldots$$

B A A C B ...

Entropy

- Now suppose someone tells you the probabilities are not equal

$$P(X = A) = \frac{1}{2},\ P(X = B) = \frac{1}{4},\ P(X = C) = \frac{1}{8},\ P(X = D) = \frac{1}{8}$$

- It is possible to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

A – 0	1
B – 10	01
C – 110	001
D – 111	000

not unique

$\frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{8}(3) + \frac{1}{8}(3) = 1.75$ bits

0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 ...

A B A B A D A A B

1 1 1 1 1 0 1 1 1 1

D D A

Entropy

- What happens if there are three equally likely values for X?

$$P(X = A) = \frac{1}{3}, \quad P(X = B) = \frac{1}{3}, \quad P(X = C) = \frac{1}{3}$$

- A naïve coding (2 bits per symbol)

A 00
B 01
C 10

- Can you think of an encoding scheme that would need only 1.6 bits per symbol on average?

A 0
B 10
C 11

$$\left(\frac{1}{3}\right)1 + \left(\frac{1}{3}\right)2 + \left(\frac{1}{3}\right)2 = 1.6$$

- In theory, it can actually be done with 1.58496 bits per symbol.

Entropy

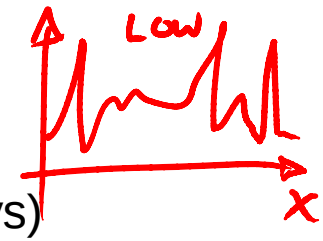
The **entropy** (H) of a discrete random variable X with k possible values represents the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from the distribution of X.

$$H(X) = -\sum_{j=1}^{k} p_j \log_2 p_j$$

$p_j = P(X = x_j)$

$= -p_1 \log_2 p_1 - p_2 \log_2 p_2 \cdots - p_k \log_2 p_k$

Boolean $\Rightarrow P(X) = q \Rightarrow H(X) = -q \log_2 q - (1-q)\log_2(1-q)$



High entropy means X is from a uniform distribution

Low entropy means X is from a varied distribution (peaks and valleys)

Entropy

$$H(X) = -\sum_{j=1}^{k} p_j \log_2 p_j$$

Back to our previous examples…

1. $P(X = A) = \frac{1}{4}$, $P(X = B) = \frac{1}{4}$, $P(X = C) = \frac{1}{4}$, $P(X = D) = \frac{1}{4}$

$H(X) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} = 2$ bits

2. $P(X = A) = \frac{1}{2}$, $P(X = B) = \frac{1}{4}$, $P(X = C) = \frac{1}{8}$, $P(X = D) = \frac{1}{8}$

$H(X) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{8}\log_2\frac{1}{8} = 1.75$

3. $P(X = A) = \frac{1}{3}$, $P(X = B) = \frac{1}{3}$, $P(X = C) = \frac{1}{3}$

$= 1.58496$

Specific Conditional Entropy

The **specific conditional entropy**, $H(Y|X = x_i)$, assumes you already know the value of X and want to measure the uncertainty of Y for that subset

$$H(Y|X = x_i) = -\sum_{j=1}^{k} P(Y = y_j|X = x_i) \log_2 P(Y = y_j|X = x_i)$$

$$H(Y) = -\sum_{j}^{k} P(Y=y_j) \log_2 P(Y=y_j)$$

Conditional Entropy

The **conditional entropy**, $H(Y|X)$, is the average specific conditional entropy of Y

$$H(Y|X) = \sum_{j=1}^{k} P(X = x_j) H(Y|X = x_j)$$

In other words, it is the entropy of Y given a randomly chosen example of X

Information Gain

The **information gain**, $\mathrm{IG}(Y|X)$, represents the expected reduction in entropy for a test on attribute X:

$$IG(Y|X) = H(Y) - H(Y|X)$$

entropy of Y

conditional entropy of Y given X

We want IG to be high!

How many bits would be saved by knowing X?

A simple example

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Suppose you want to predict Y and you know X:

$IG(Y|X)$

$P(Y) = 0.5 \quad P(\sim Y) = 1 - P(Y) = 0.5$

$P(X = \text{MATH}) = 0.5$

$P(X = \text{HISTORY}) = 0.25$

$P(X = CS) = 0.25$

$P(Y | X = \text{MATH}) = 0.5$

$P(Y | X = \text{HISTORY}) = 0$

$P(Y | X = CS) = 1$

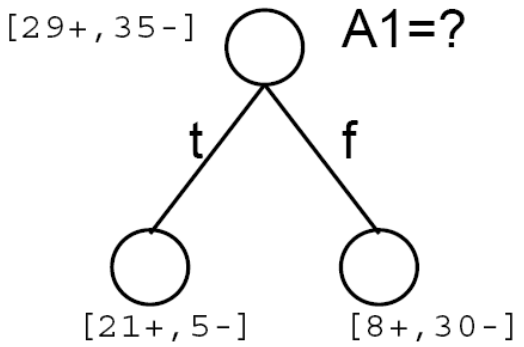
$$H(X) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} = 1.5$$

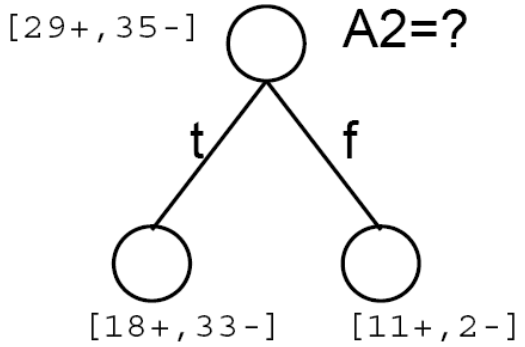
$$H(Y) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

Back to learning decision trees…

How do you pick the best attribute to split on?

Use the one with the highest information gain!





Learning decision trees from data

Top-down induction

1. Determine the “best” decision attribute from those available
2. Assign that attribute to the next node
3. For each value of that attribute, create a descendant node
4. Sort training examples for each descendant
5. Recursively build tree from each descendant until stopping criteria is met
6. At each leaf node, assign the appropriate classification

Stopping criteria

- The subset of training examples have the same output
- The subset of training examples have the same values for all input attributes
- There are no training examples for a particular leaf node

- The information gain is zero for all attributes???

Zero information gain for all attributes

Is this a good idea?

Let's build a decision tree

Day	Outlook	Temp	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Let's build a decision tree

Day	Outlook	Temp	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Let's build a decision tree

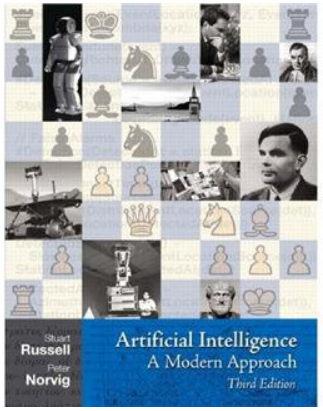
Day	Outlook	Temp	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Continuous-valued attributes

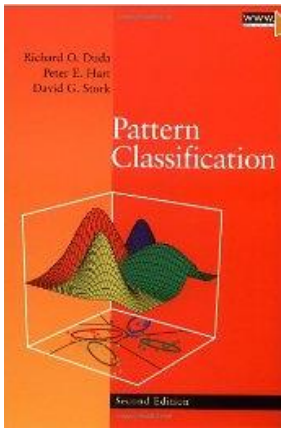
Dynamically create attributes by choosing a threshold that maximizes information gain.

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

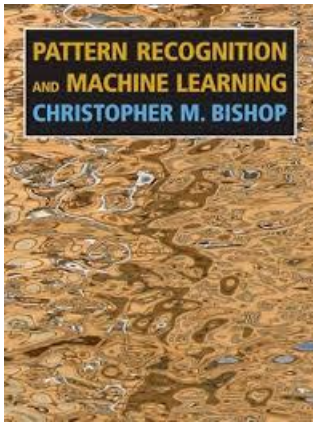
References



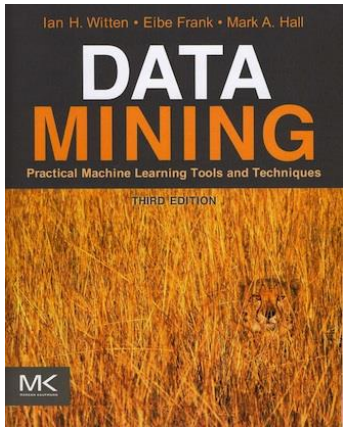
18.1 – 18.3



8.1 – 8.4, Appendix A.7



14.4



3.3, 4.3, 6.1

*Some slides have been adapted from Andrew Moore (http://www.autonlab.org/tutorials/) and Levent Burak Kara

Project Idea

http://espn.go.com/mens-college-basketball/story/_/id/10328805/1-billion-offered-perfect-tournament-bracket&ex_cid=sportscenter