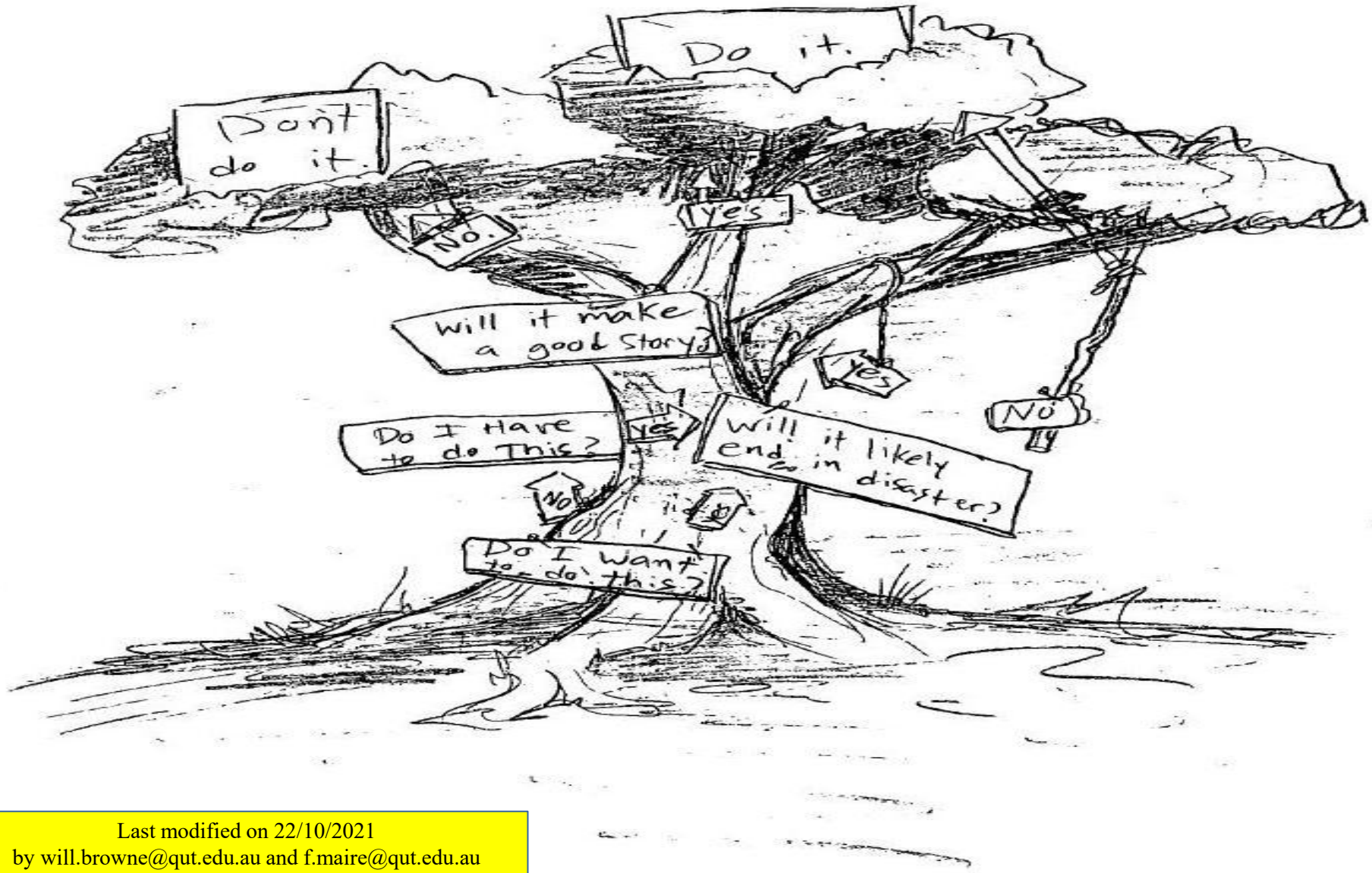
Decision Trees (DT)



Last modified on 22/10/2021
by will.browne@qut.edu.au and f.maire@qut.edu.au

Reading for this week

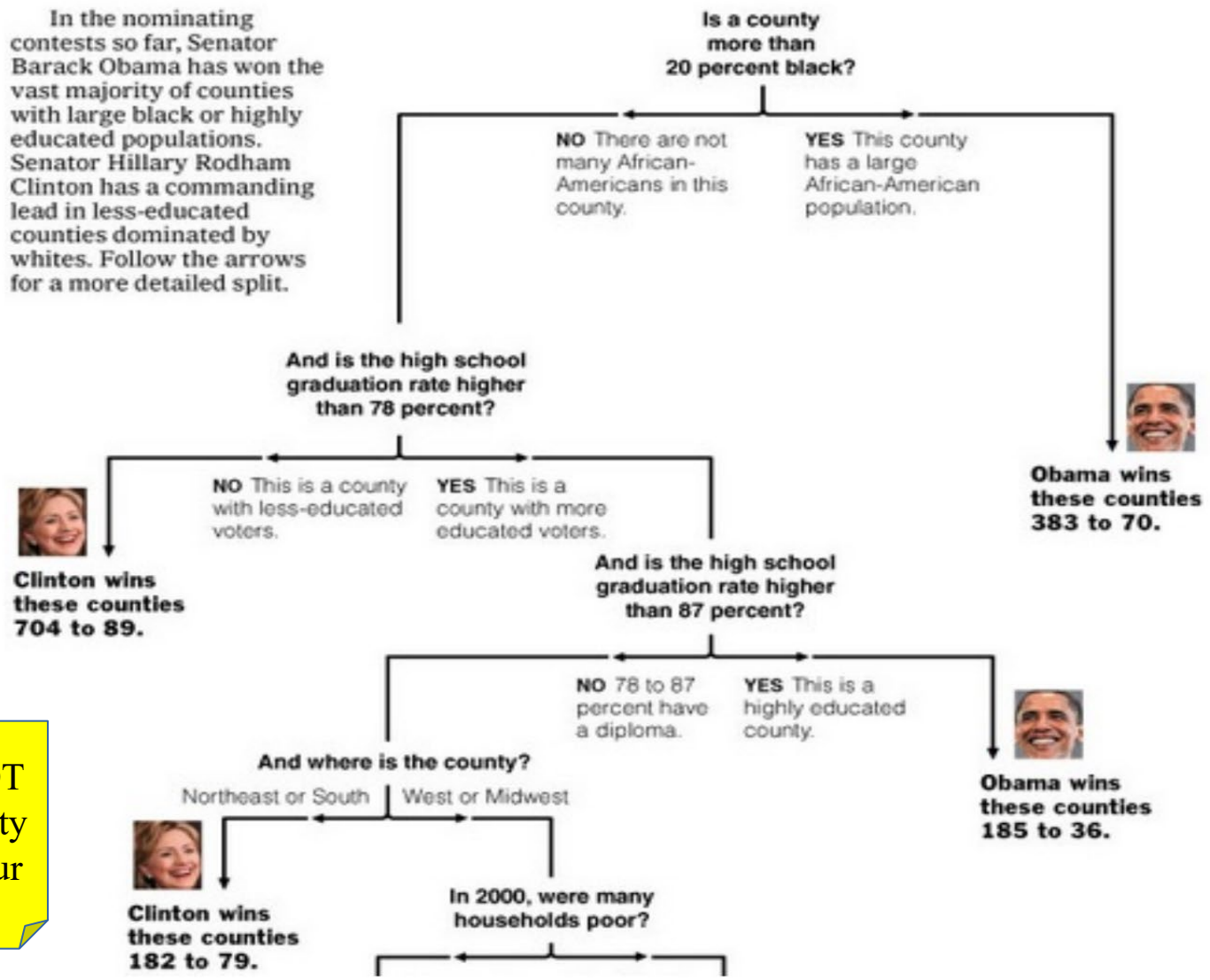
- Sections 18.3 and 18.4 of AIMA

Russell and Norvig Textbook:
Artificial Intelligence, a Modern Approach
3rd edition

Today's Menu

- Decision Tree (DT) hypothesis space
- Building DTs
  - choosing the best variable
  - entropy
  - information gain
- Avoiding overfitting when building DTs

Decision Tree: The Obama-Clinton Divide

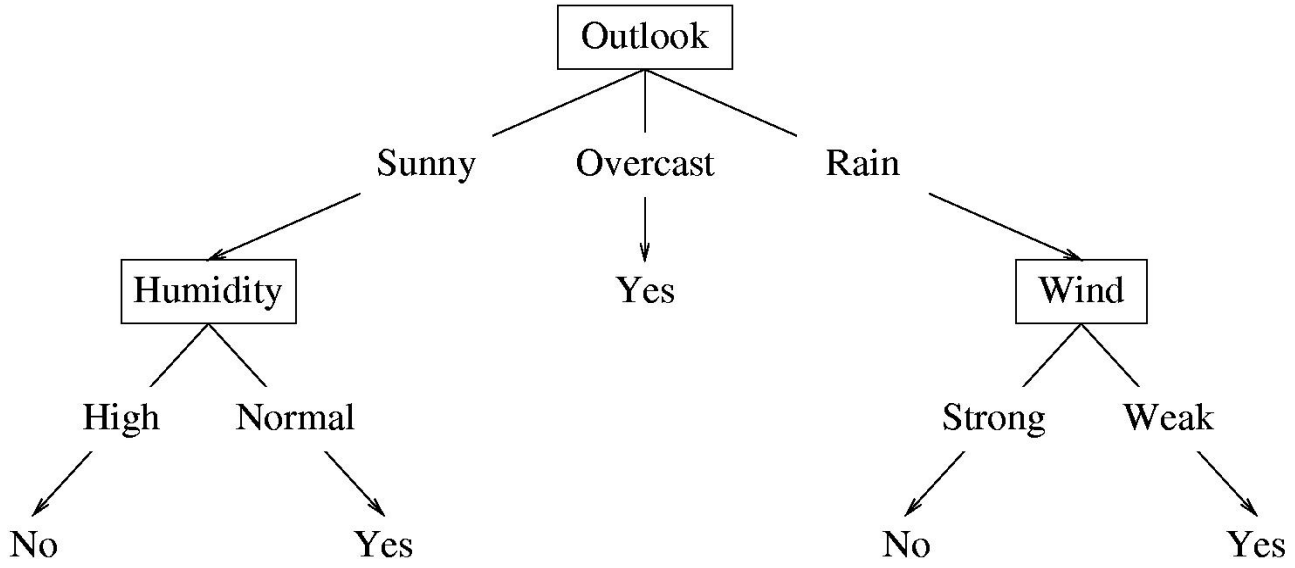


Example of a DT to classify county voting behaviour

http://graphics8.nytimes.com/images/2008/04/16/us/0416-nat-subOBAMA.jpg

Decision Tree Hypothesis Space

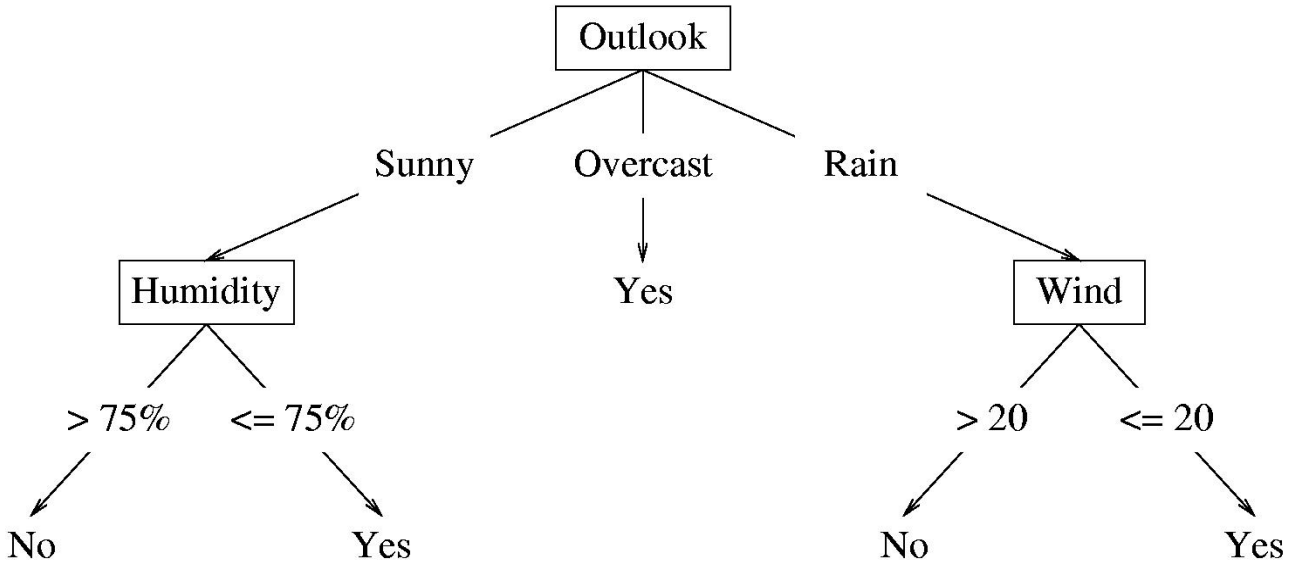
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



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.



A Tree to Predict C-Section Risk

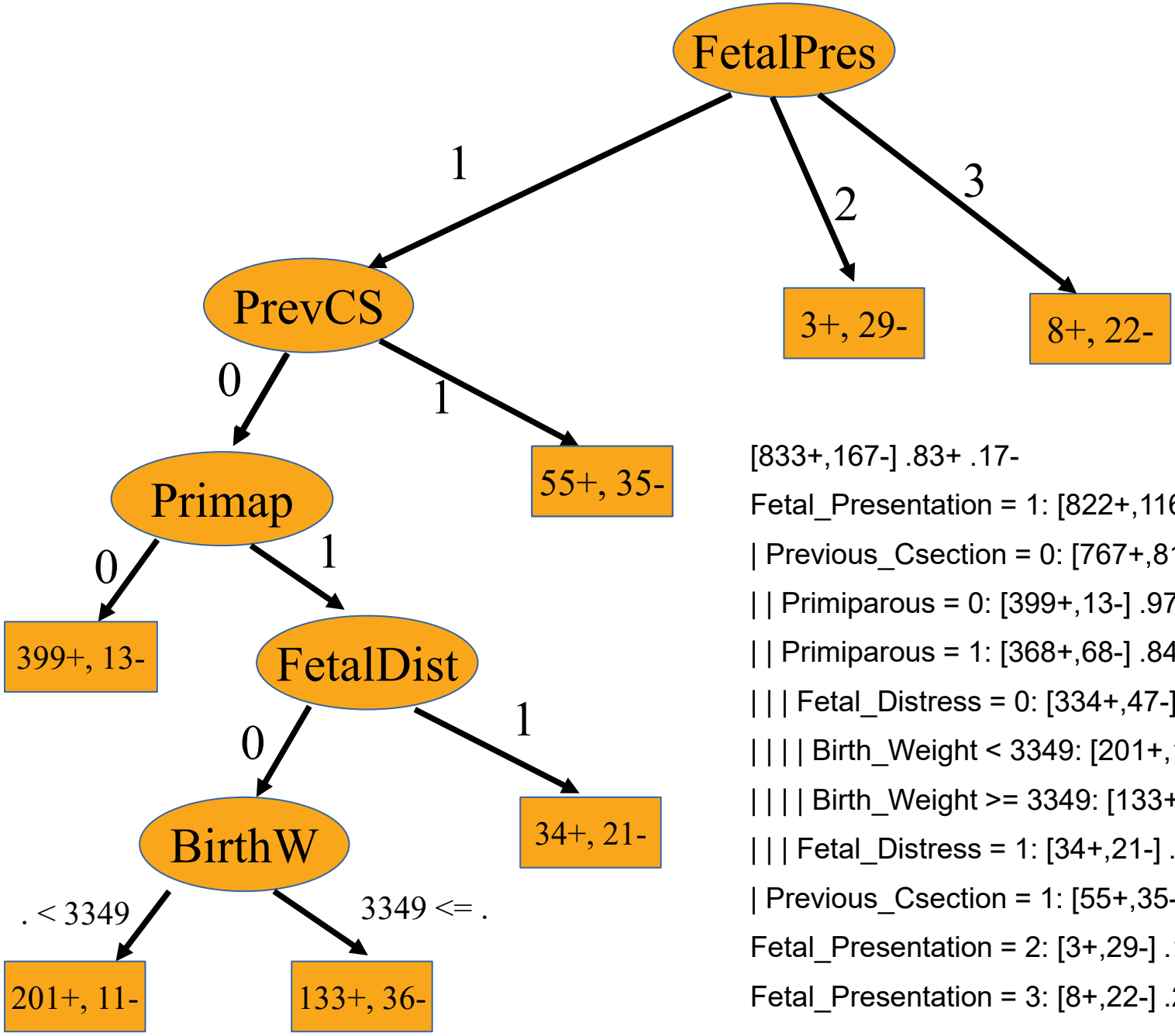
Learned from medical records of 1000 women

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+,11-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36-] .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-

Example of a DT to help doctors decide whether to perform a C-section

- What are the variables?
- Outline of the tree?
- Depth of the tree?



[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+,11-] .95+ .05-

| | | | Birth_Weight >= 3349: [133+,36-] .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-

Decision Trees

- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to an attribute value
  - Each leaf node assigns a classication

- How would you represent the following functions with a DT?
  - ***or*** function ; ***and*** function ; ***xor*** function
  - (*A* ***and*** *B*) ***or*** (C ***and*** ***not*** *D* ***and*** *E*)
  - *M* ***of*** *N*

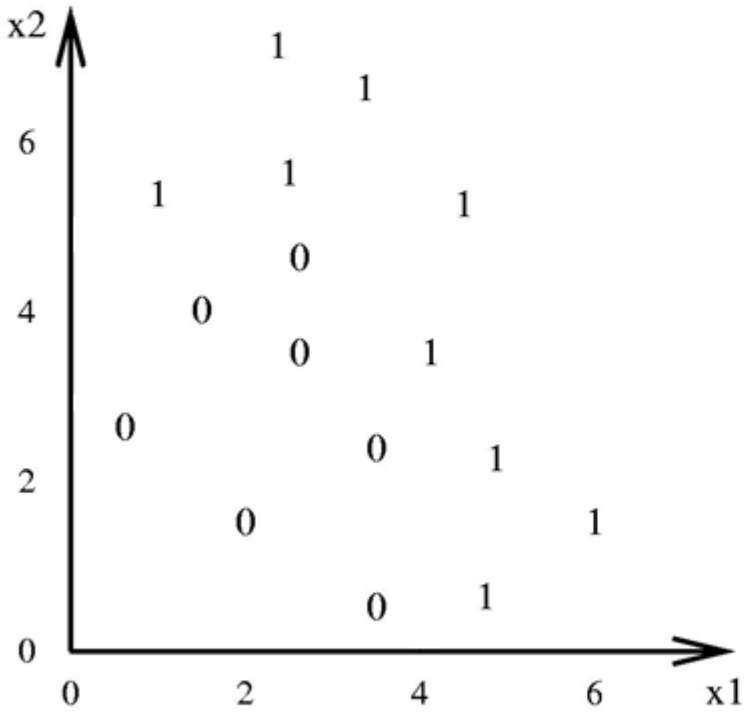
When to Consider Decision Trees

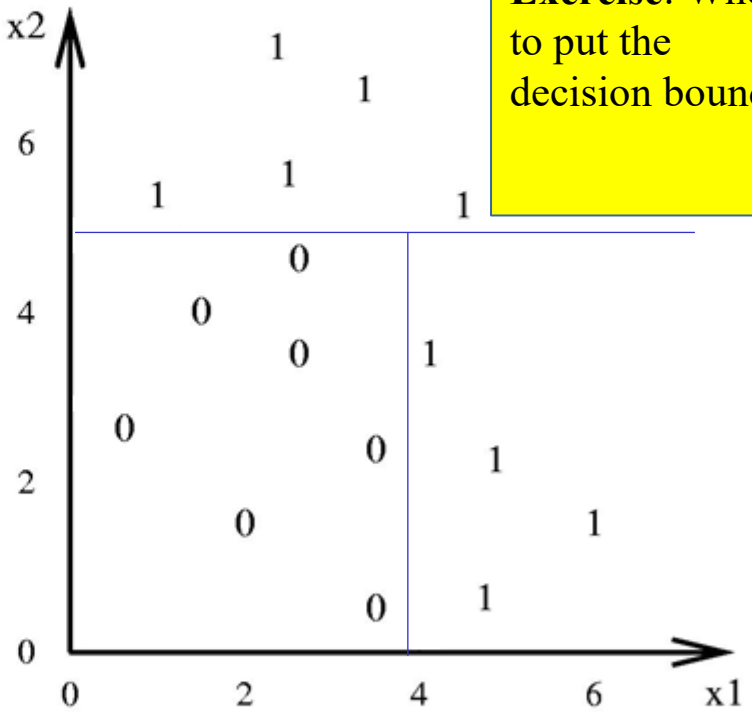
- Instances describable by **attribute-value** pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

- Examples:
  - Equipment or medical diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences

Decision Tree Decision Boundaries

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

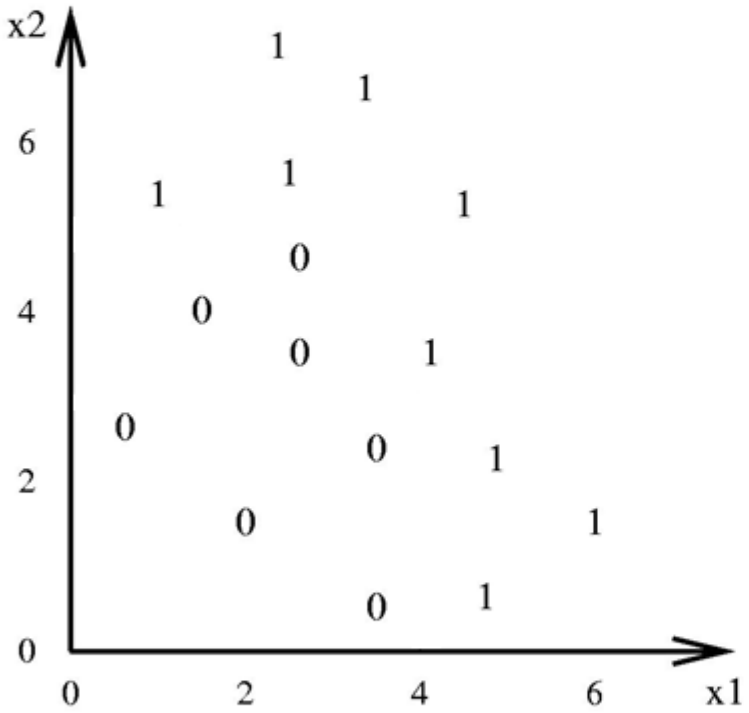


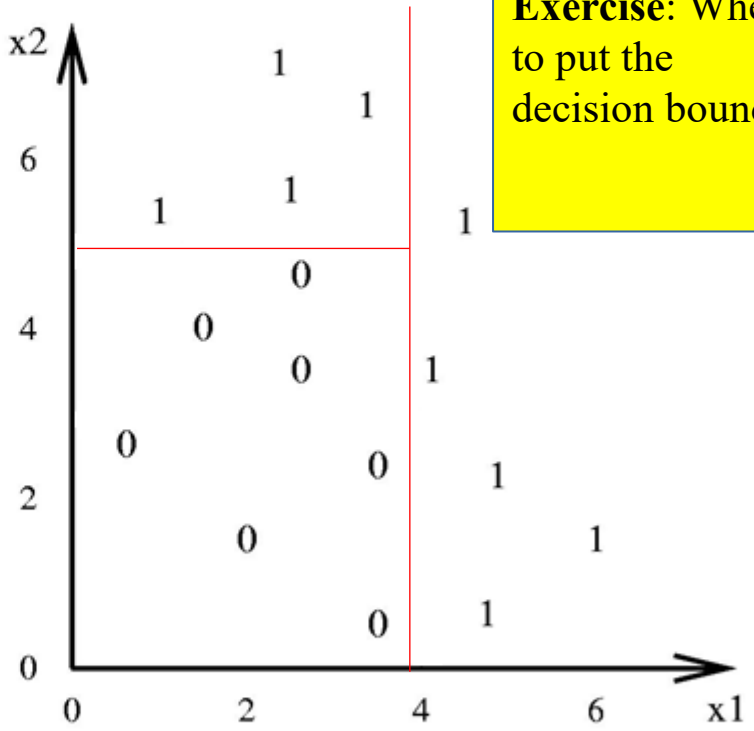


Exercise: Where to put the decision boundaries?

Decision Tree Decision Boundaries

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

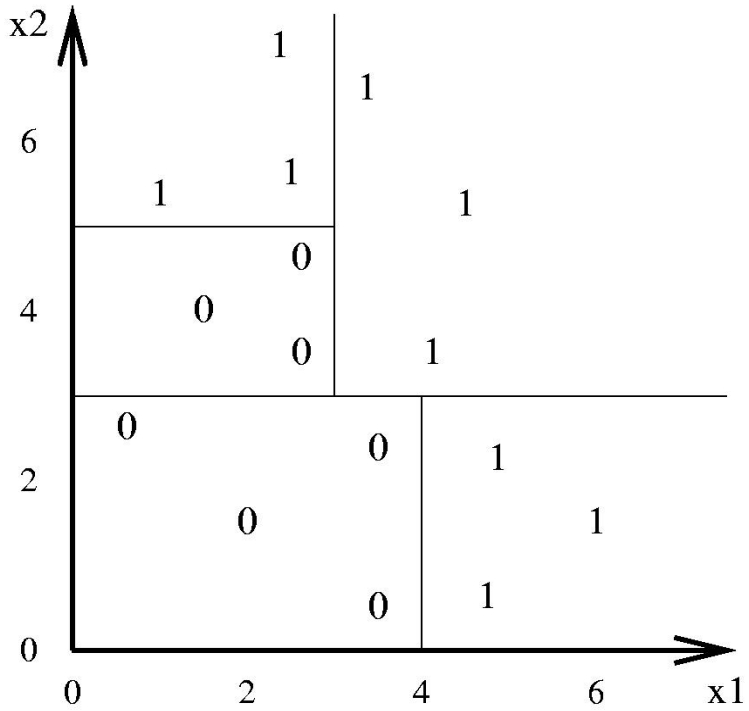


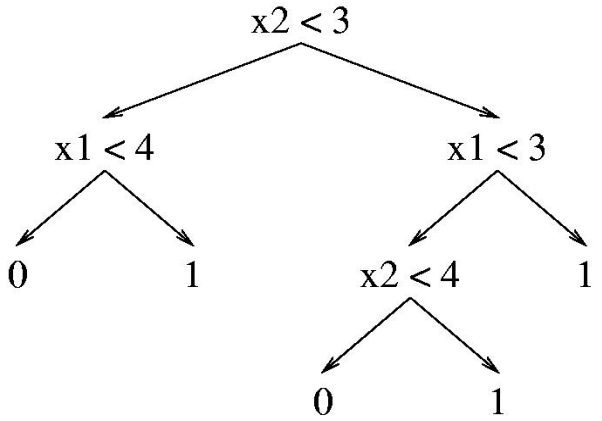


Exercise: Where to put the decision boundaries?

Decision Tree Decision Boundaries

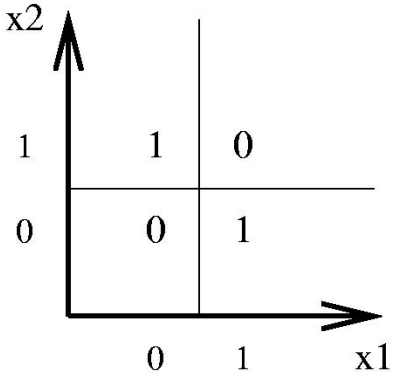
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

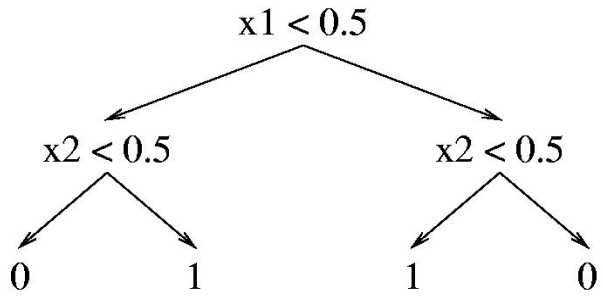




Exercise: express the subset classified with label 0 as a disjunction of conjunctions

Decision Trees Can Represent Any Boolean Function





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

Exercise: can you think of a boolean function that requires exponentially many nodes?

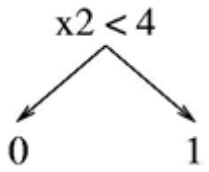
Answer: parity function

Decision Trees Provide Variable-Size Hypothesis Space

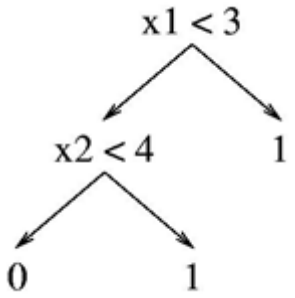
As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **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.**

depth 1



depth 2



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))

How?!

S is the training set.
S gets filtered as
the tree grows.

We want to find boundary that separates the most instances recursively. Note S changes after each split (node / depth)

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.

A natural/obvious idea...

ChooseBestAttribute(S)

Loss function

choose j to minimize J_j, computed as follows:

$S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;

Filtering of S

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

$y_0 =$ the most common value of y in S_0

Majority rule

$y_1 =$ the most common value of y in S_1

$J_0 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$

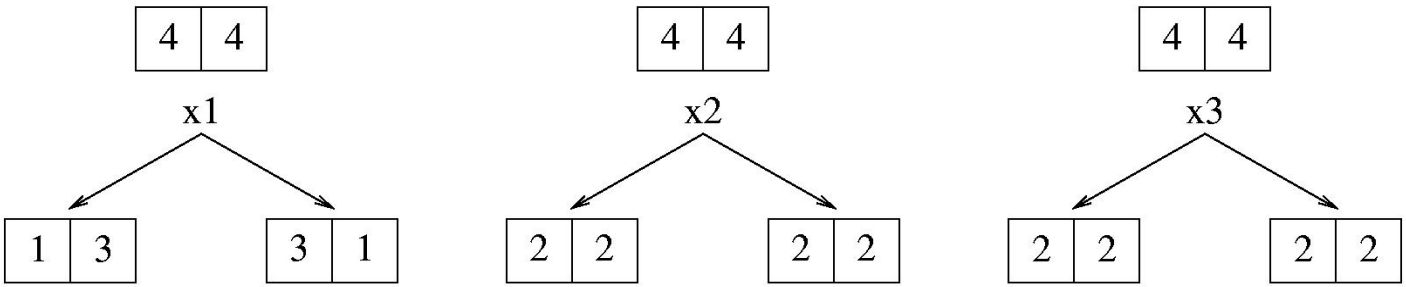
$J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$

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

return j

Choosing the Best Attribute—An Example

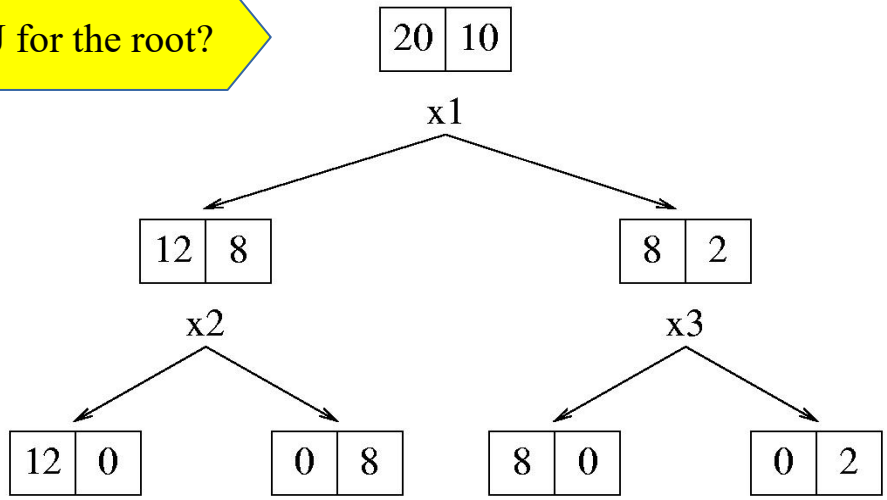
x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



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.

What's J for the root?



Exercise: What's J for the root?

Answer: 10 "1" will be misclassified "0"

Information Gain Information Theory

- Goal: Select the attribute that will result in the smallest expected tree size
- How: Use information theory

How many yes/no questions would you expect to ask to determine a number someone picks in the range 1 to 100?

- With each yes/no question at most 1/2 of the elements remaining can be eliminated

$$\log_2 100 = 6.64$$

Given a set S of size $|S|$, the expected work required to determine a specific element is $\log_2 |S|$

We can use this to determine what is the number of questions on average to discovery something v that has a probability of occurring $p(V = v)$:

A Better Heuristic From Information Theory

Let V be a random variable with the following probability distribution:

$P(V = 0)$	$P(V = 1)$
0.2	0.8

The *surprise*, $S(V = v)$ of each value of V is defined to be

$$S(V = v) = -\lg P(V = v).$$

Surprise of
the measurement
V = v
given the prior

An event with probability 1 gives us zero surprise.

An event with probability 0 gives us infinite surprise!

It turns out that the surprise is equal to the number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results.

This is also called the *description length* of $V = v$.

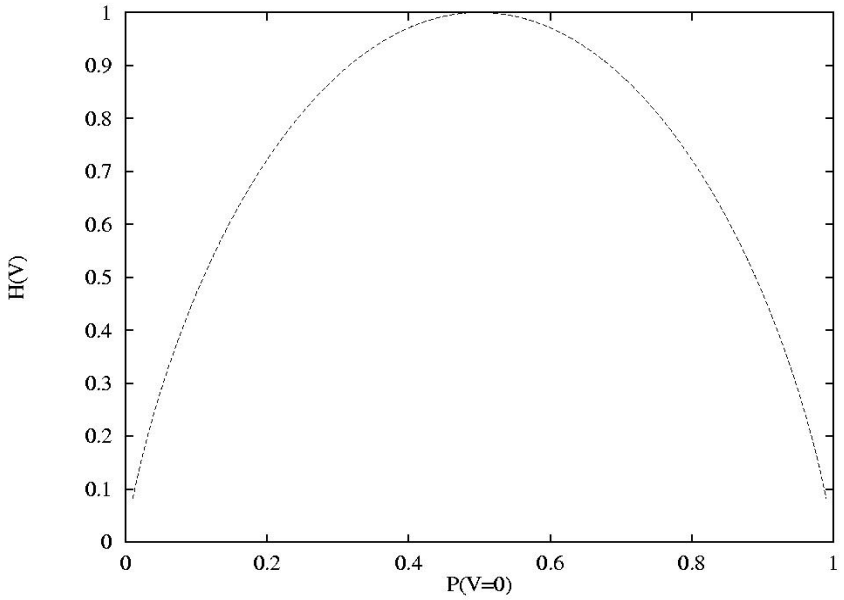
Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).

Entropy

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

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

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.

Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of S:

$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

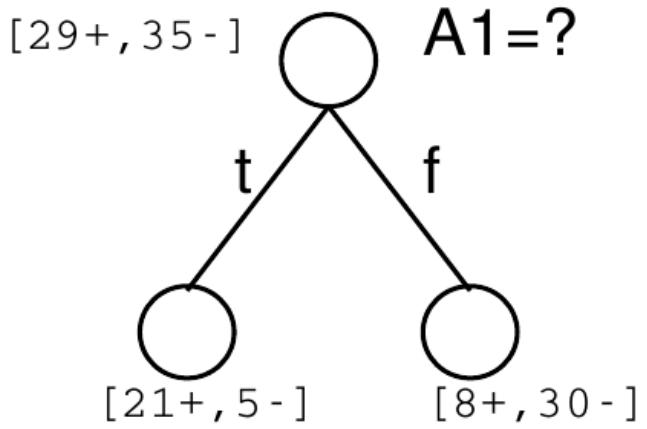
$$Entropy(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

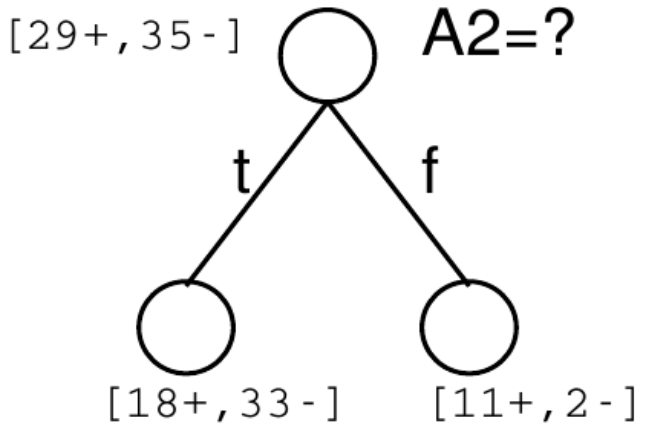
See also
http://en.wikipedia.org/wiki/Huffman_coding

Information Gain

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

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



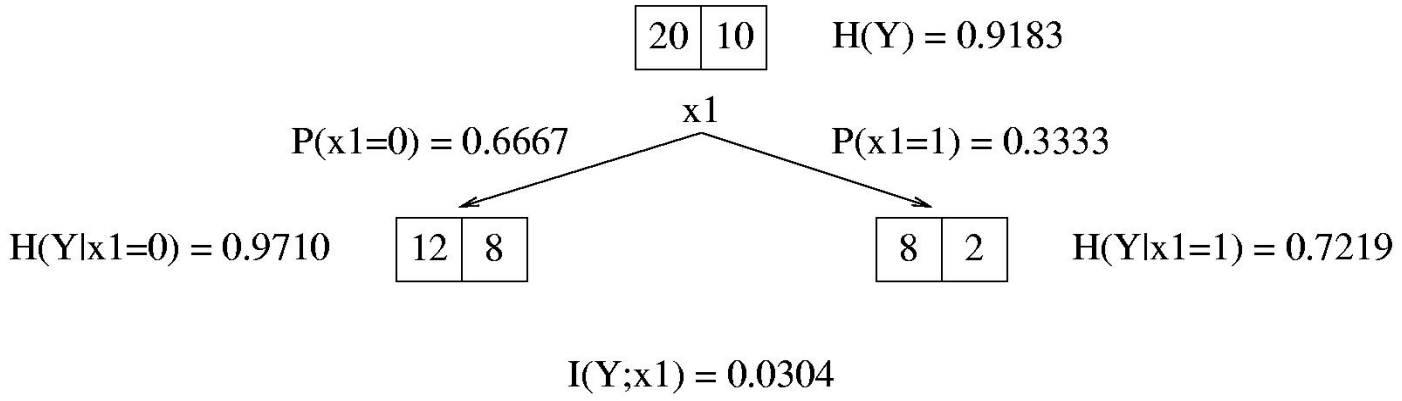


Mutual Information

Now consider two random variables A and B that are not necessarily independent. The *mutual information* between A and B is the amount of information we learn about B by knowning the value of A (and vice versa—it is symmetric). It is computed as follows:

$$I(A;B) = H(A) - \sum_b H(A|B=b)\,P(B=b)$$

In particular, consider the class Y of each training example and the value of feature x_1 to be random variables. Then the mutual information quantifies how much x_1 tells us about the value of the class Y.



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

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

how much information do we need to tell which branch an instance belongs to

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

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

Unknown Attribute Values

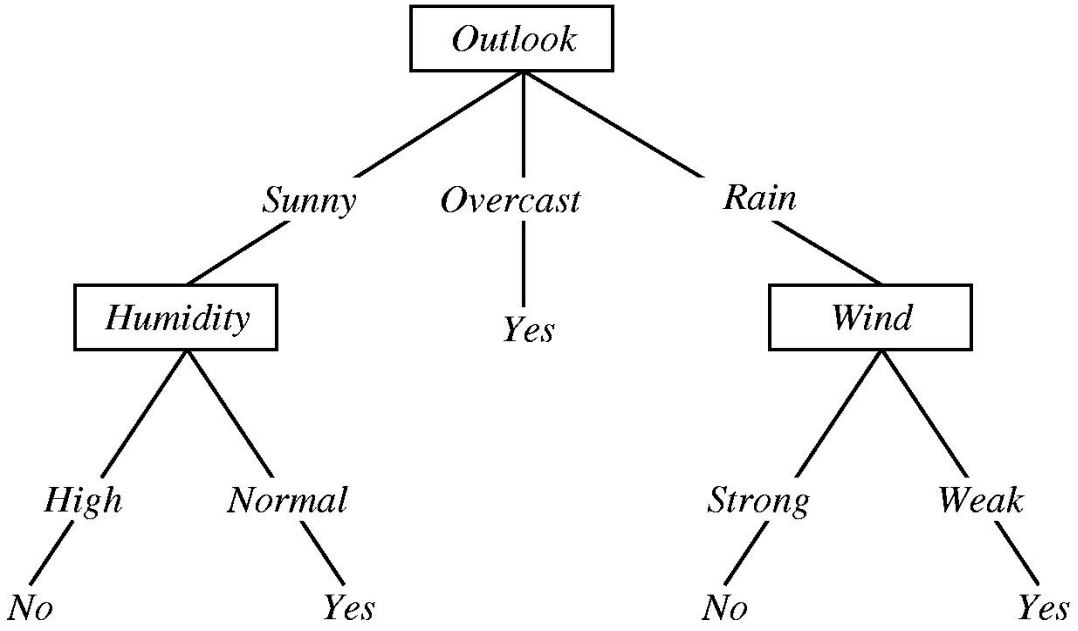
What if some examples are missing values of A?

Use training example anyway, sort through tree

- If node n tests A, assign most common value of A among other examples sorted to node n
- Assign most common value of A among other examples with same target value
- Assign probability p_i to each possible value v_i of A
  Assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion

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: $error_{\mathcal{D}}(h)$

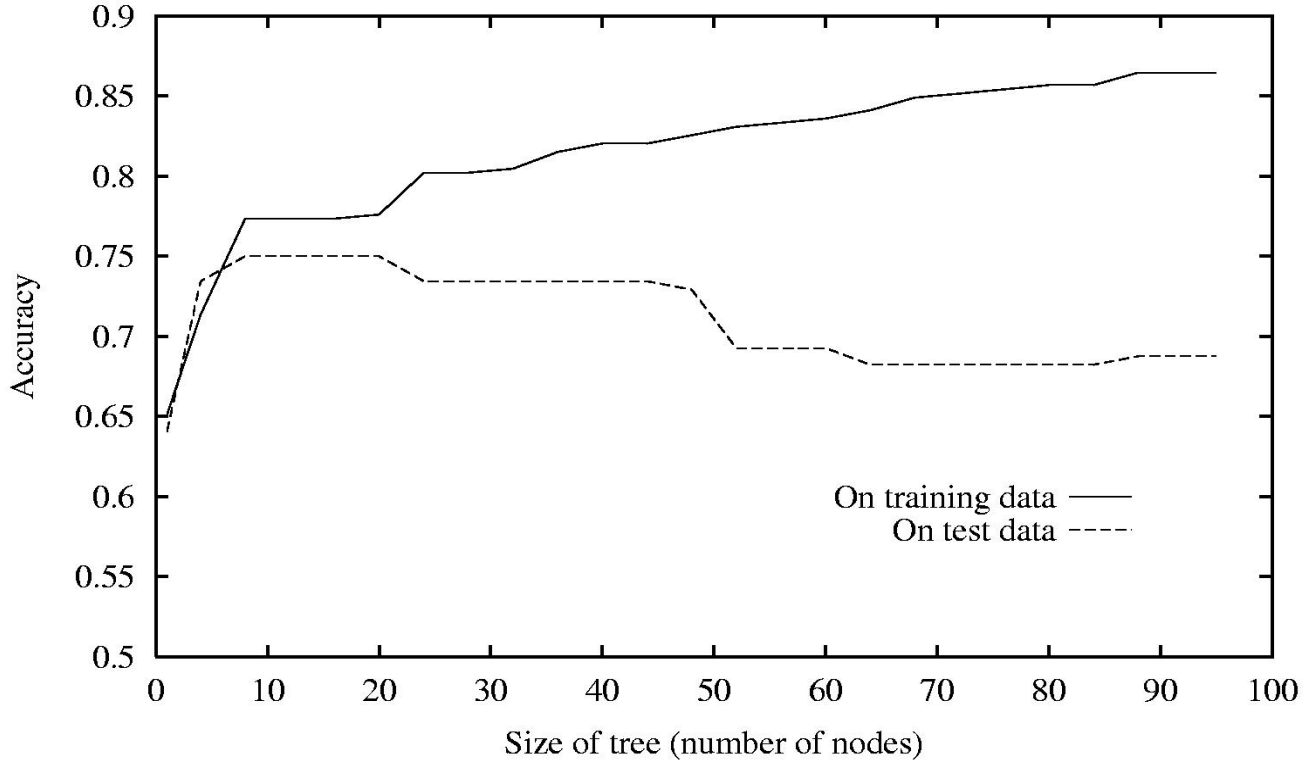
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

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

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

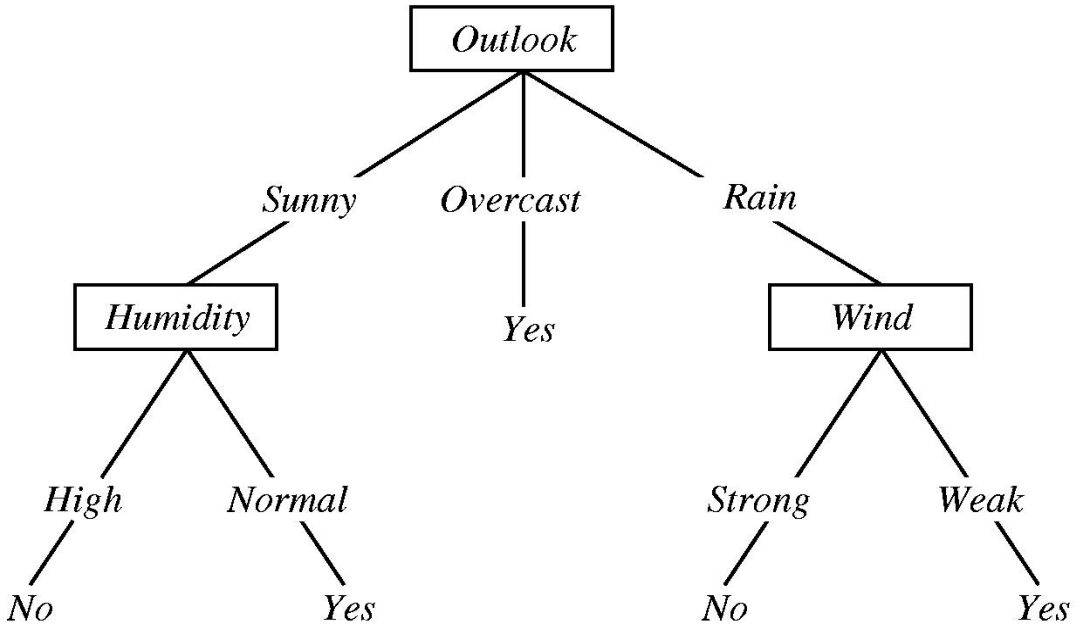
Reduced-Error Pruning

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

Converting A Tree to Rules



IF $(Outlook = Sunny)\ AND\ (Humidity = High)$
THEN $PlayTennis = No$

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

...

See sklearn documentation

http://scikit-learn.org/stable/modules/tree.html

Summary: Decision Trees

- Efficient learning algorithm
- Handle both discrete and continuous inputs and outputs
- Robust against any monotonic input transformation, also against outliers
- Automatically ignore irrelevant features: no need for feature selection
- Decision trees are usually interpretable