CS 349:Machine Learning

Bryan Pardo & Bongjun Kim

Topic: Decision Trees

(Includes content provided by: Tom Mitchell, Russel & Norvig, D. Downie, P. Domingos)

General Learning Task

There is a set of possible examples $X = \{\vec{x}_1, \ldots \vec{x}_n\}$

Each example is an n-tuple of attribute values

$$\vec{x}_1 = < a_1, \ldots, a_k >$$

There is a target function that maps *X* onto some finite set *Y*

$$f : X \rightarrow Y$$

The DATA is a set of duples <example, target function values>

$$D = \{< \vec{x}_1, f(\vec{x}_1) >, \ldots < \vec{x}_m, f(\vec{x}_m) >\}$$

Find a hypothesis ***h*** such that...

$$\forall \vec{x}, h(\vec{x}) \approx f(\vec{x})$$

Attribute-based representations

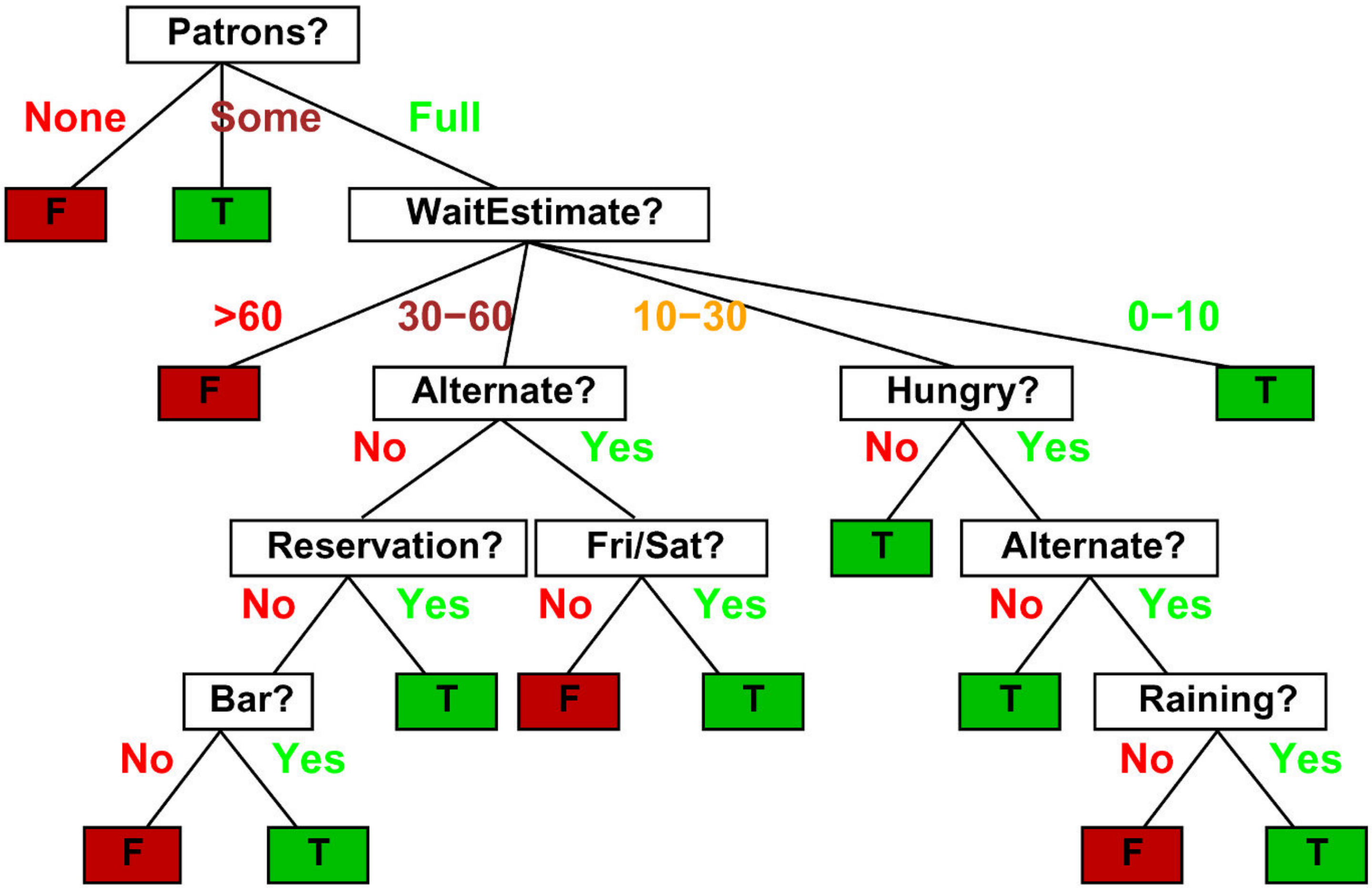
Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Classification of examples is positive (T) or negative (F)

Decision Tree

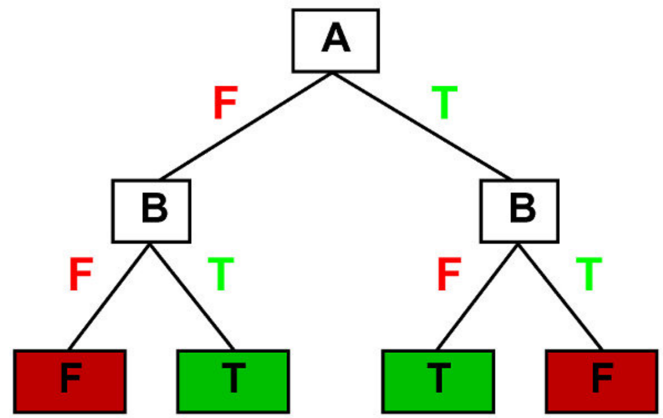
One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



Expressiveness of D-Trees

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

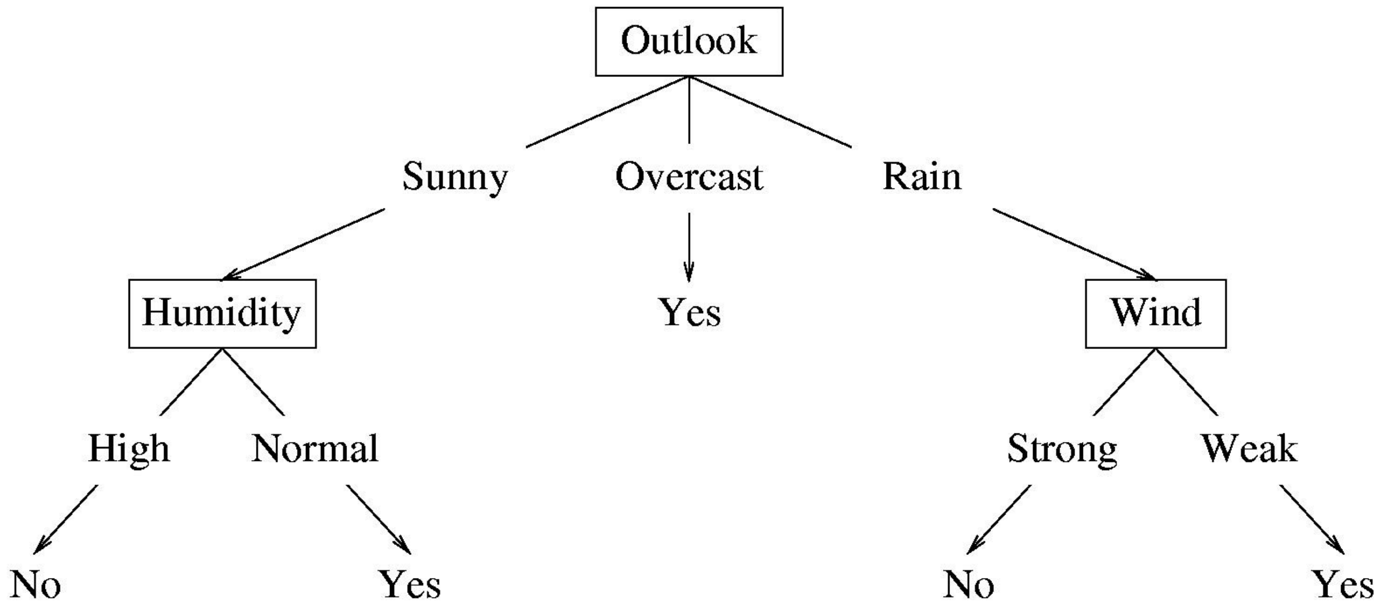
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set
w/ one path to leaf for each example (unless f nondeterministic in x)
but it probably won't generalize to new examples

Prefer to find more **compact** decision trees

Decision Trees represent *disjunctions* of *conjunctions*



$$f(x) = yes \text{ iff...}$$

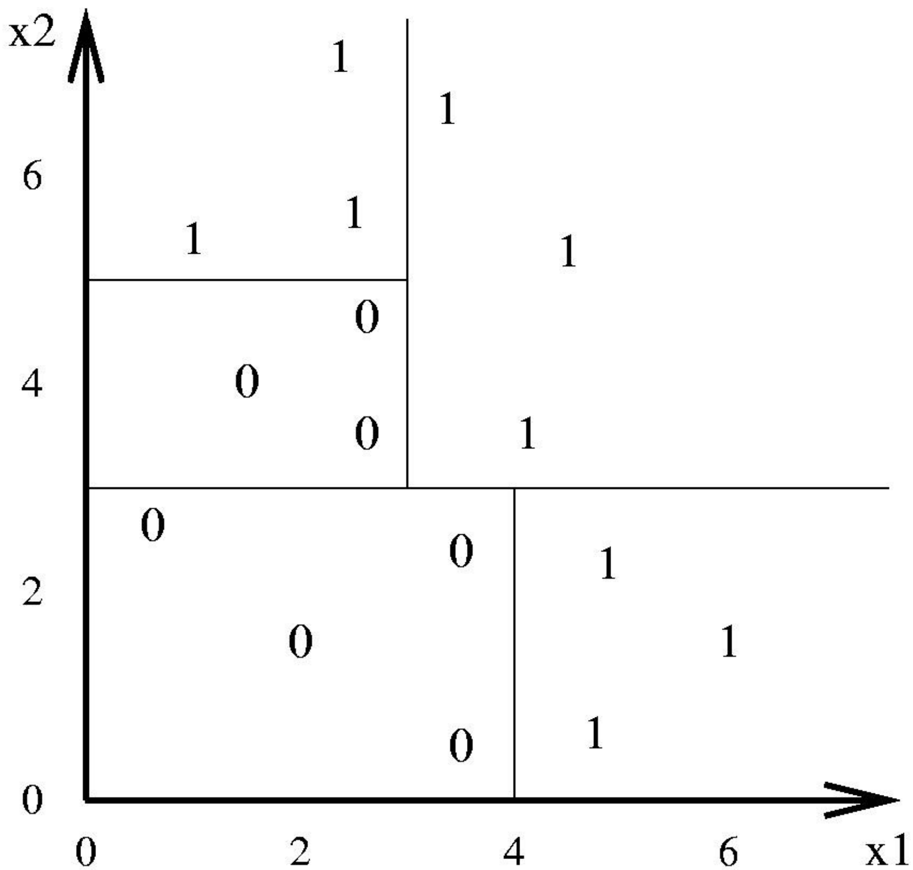
$$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal}) \vee$$

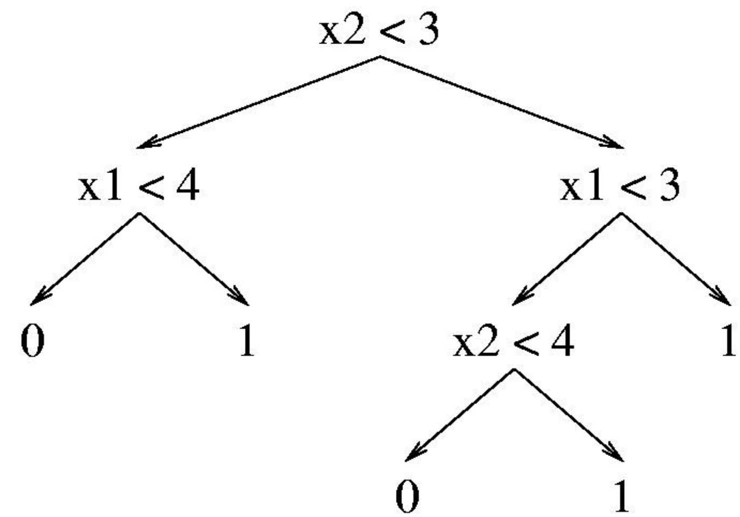
$$(\text{Outlook} = \text{overcast}) \vee$$

$$(\text{Outlook} = \text{rain} \wedge \text{Wind} = \text{weak})$$

Decision Tree Boundaries

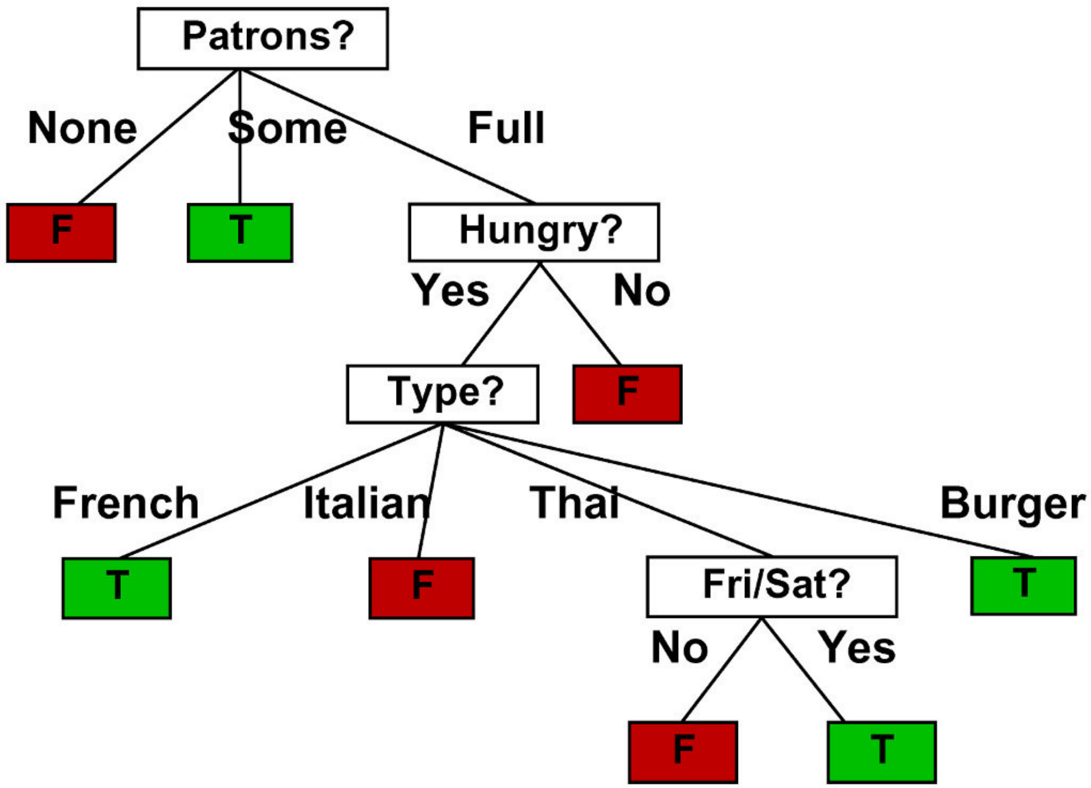
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





A learned decision tree

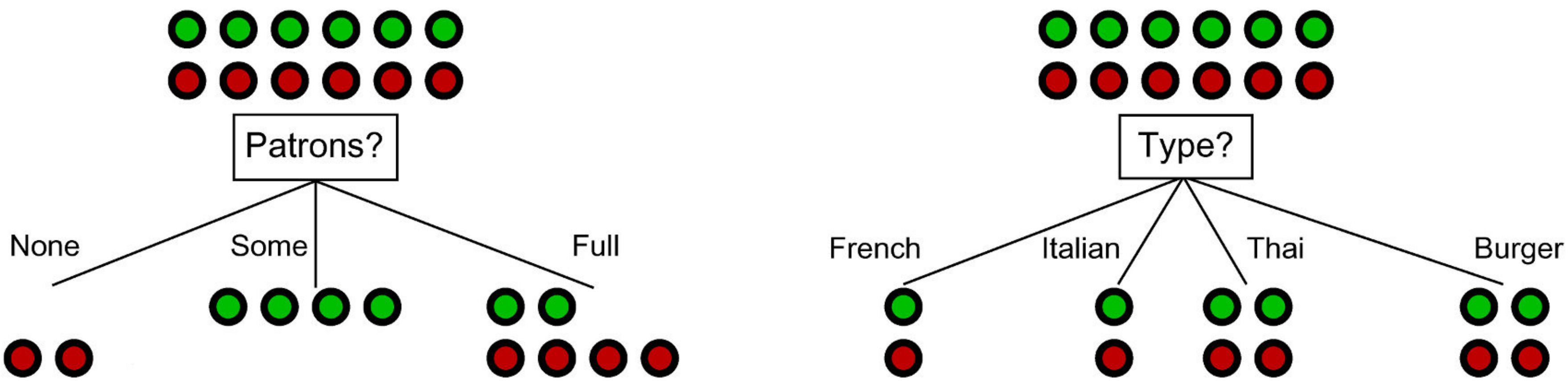
Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

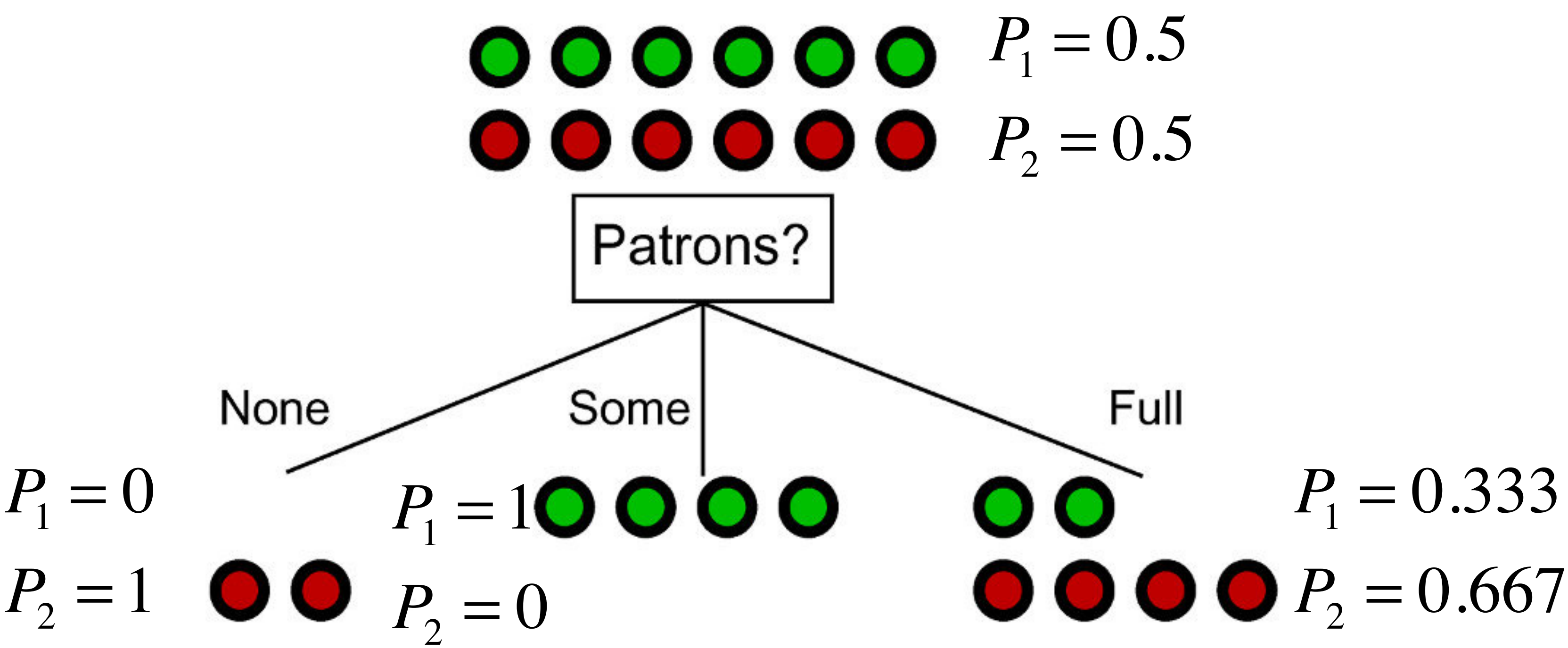
The more skewed the examples in a bin, the better.

We're going to use ENTROPY to as a measure of how skewed each bin is.

Counts as probabilities

$P_1 =$ probability I will wait for a table

$P_2 =$ probability I will NOT wait for a table



Information

Information answers questions

The more clueless I am about the answer initially, the more information contained in the answer

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \Sigma_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)

About ID3

- A recursive, greedy algorithm to build a decision tree
- At each step it picks the best variable to split the data on, and then moves on
- It is "greedy" because it makes the optimal choice at the current step, without considering anything beyond the current step.
- This can lead to trouble, if one needs to consider things beyond a single variable (e.g. multiple variables) when making a choice. (Try it on XOR)

Decision Tree Learning (ID3)

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Choosing an attribute in ID3

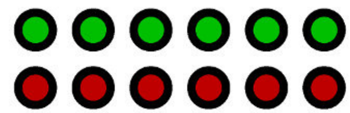
- For each attribute, find the entropy *H* of the example set AFTER splitting on that example

  *note, this means taking the entropy of each subset created by splitting on the attribute, and then combining these entropies…weighted by the size of each subset.

- Pick the attribute that creates the lowest overall entropy.

Entropy prior to splitting

Instances where I waited
Instances where I didn’t



$$P_1 = \text{probability I will wait for a table}$$

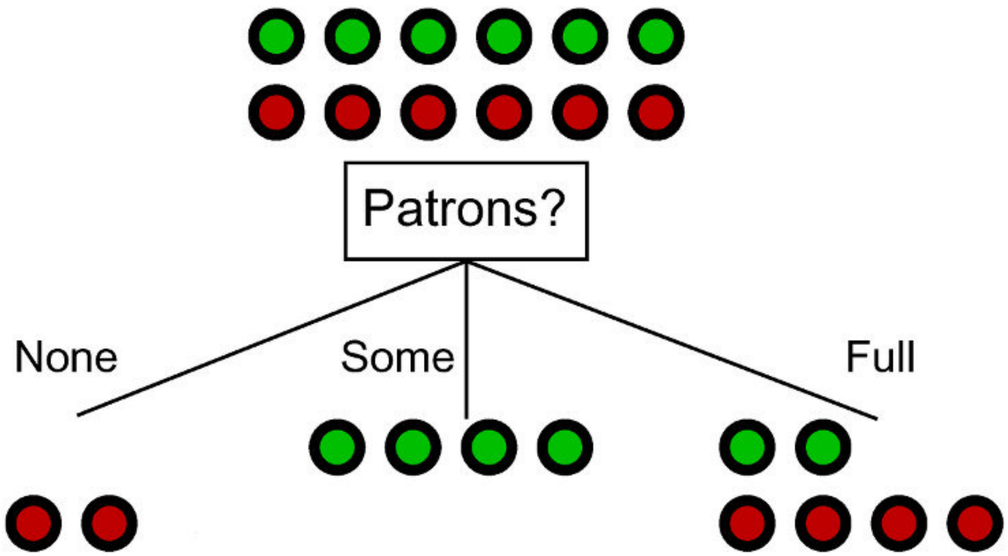
$$P_2 = \text{probability I will NOT wait for a table}$$

$$H_0\langle P_1, P_2\rangle = \sum_j -P_j \log_2 P_j$$

$$= -P_1 \log_2 P_1 - P_2 \log_2 P_2$$

$$= 1$$

If we split on Patrons

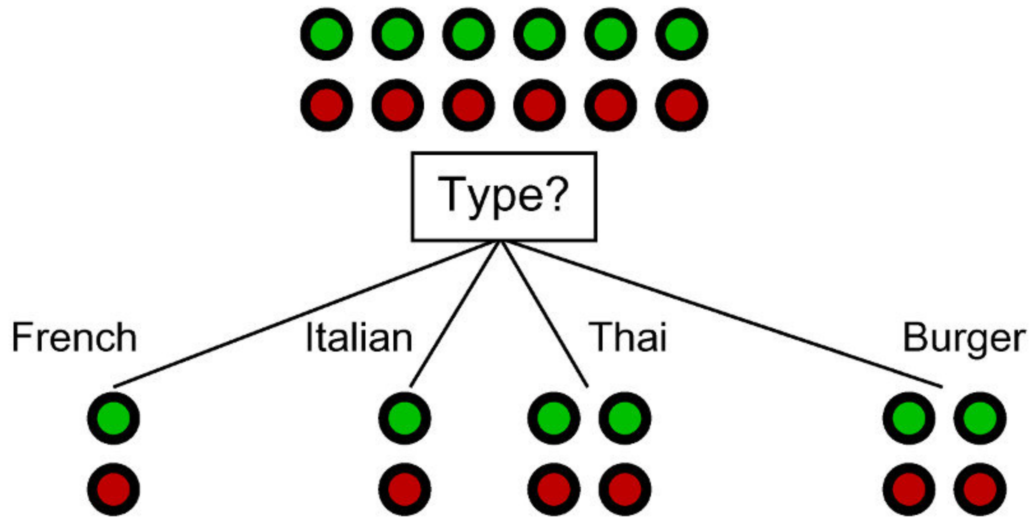


$$H_{none} \qquad H_{some} \qquad H_{full}$$

$$H_{Patrons} = W_{none} H_{none} + W_{some} H_{some} + W_{full} H_{full}$$

$$= \frac{2}{12} 0 + \frac{4}{12} 0 + \frac{6}{12} \left(-\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} \right) = .459$$

If we split on Type



$$H_{Type} = W_{french} H_{french} + W_{italian} H_{italian} + W_{thai} H_{thai} + W_{burger} H_{burger}$$

$$= \frac{2}{12}1 + \frac{2}{12}1 + \frac{4}{12}1 + \frac{4}{12}1 = 1$$

Decision Tree Learning (ID3)

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

A fully worked-out example

- Let's build a tree with ID3 to decide whether, on a particular day, I will play tennis.

Training examples

Someone followed me around for two weeks to see if I played tennis each day and collected the following data to train a model.

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

Information Gain

- The expected reduction in Entropy
- Entropy (before split) – Entropy (after split)

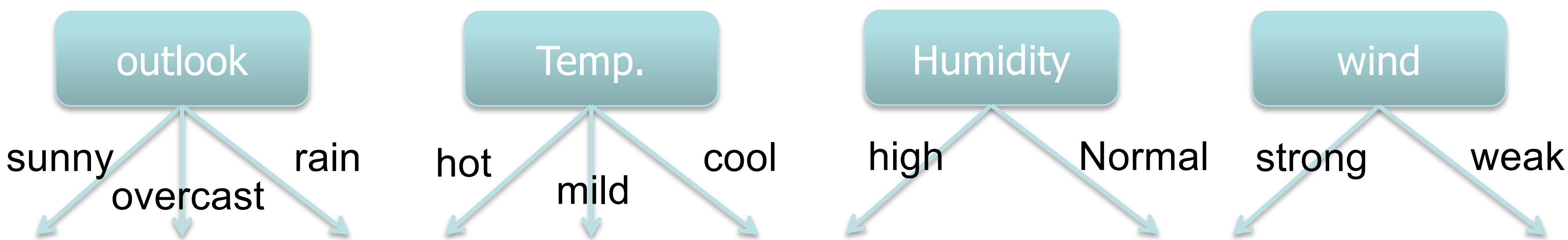
S = the set of examples
|S| = the number of examples in the set S
A = the attribute you're splitting the data on
v = one of the values attribute A can take
S_v = the subset of S where the examples take value v on attribute A

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

How to choose the best attribute

- Pick the one with the most information gain
  - The expected reduction in Entropy
  - Entropy (before split) – Entropy (after split)

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



Entropy before & after

P_1 = probability I will play tennis

P_2 = probability I will NOT play tennis

$$Entropy(S) = \sum_j -P_j \log_2 P_j$$

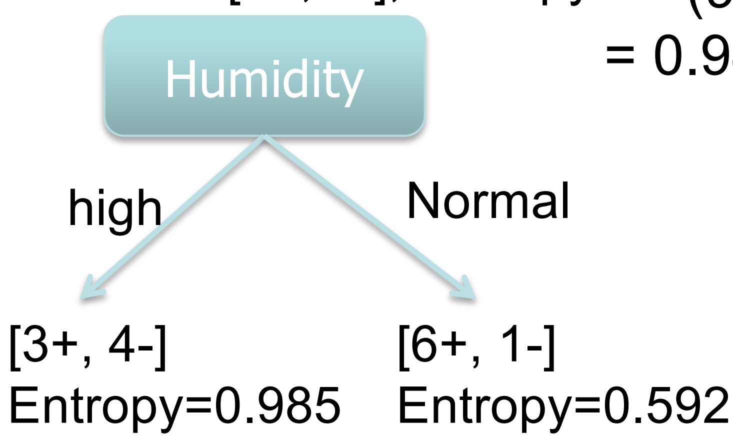
$$= -P_1 \log_2 P_1 - P_2 \log_2 P_2$$

9 'yes' examples

5 'no' examples

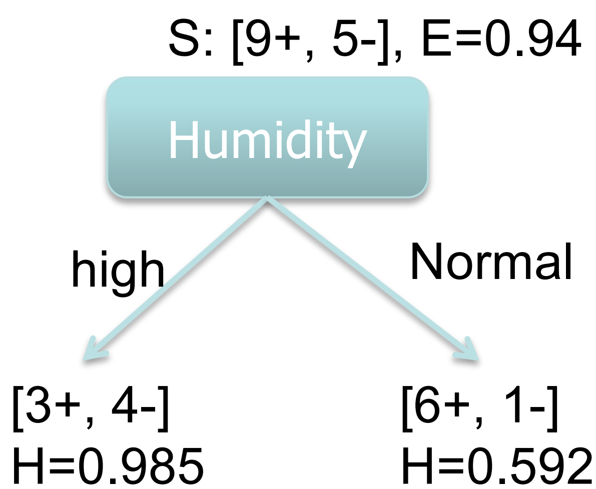
S: [9+, 5-], Entropy = −(9/14)log2(9/14) −(5/14)log2(5/14)

= 0.94



Information gain for Humidity

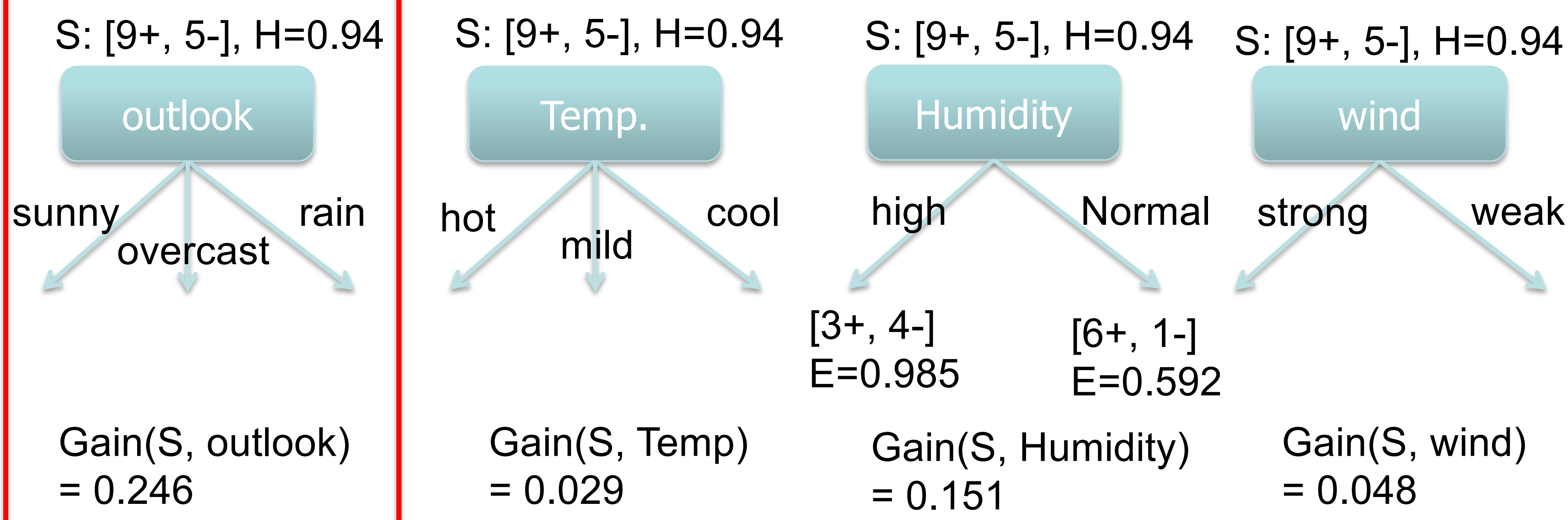
$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



Gain(S, Humidity)
= 0.94 – ((7/14)*0.985+(7/14)*0.592
= 0.151

(remember, we're using H to mean 'entropy')

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

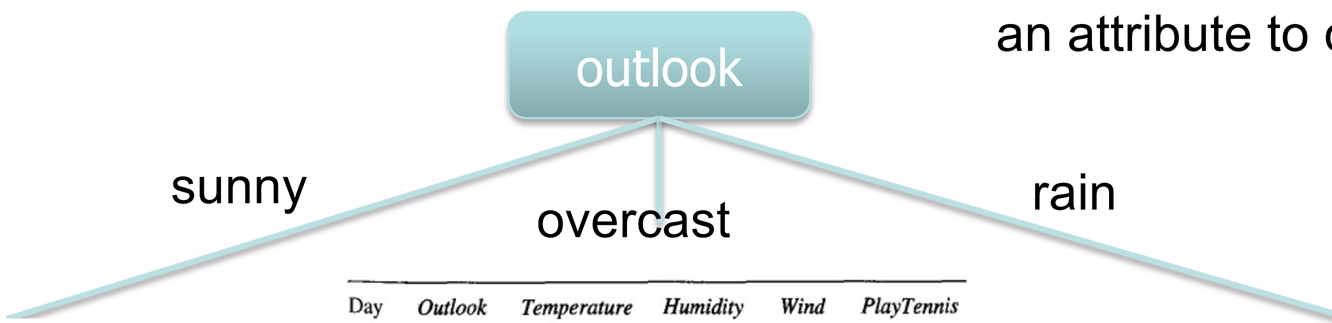


Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool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

Since ‘outlook’ has the most information gain, this becomes the root of the decision tree.

We split the data into subsets, based on the outlook.

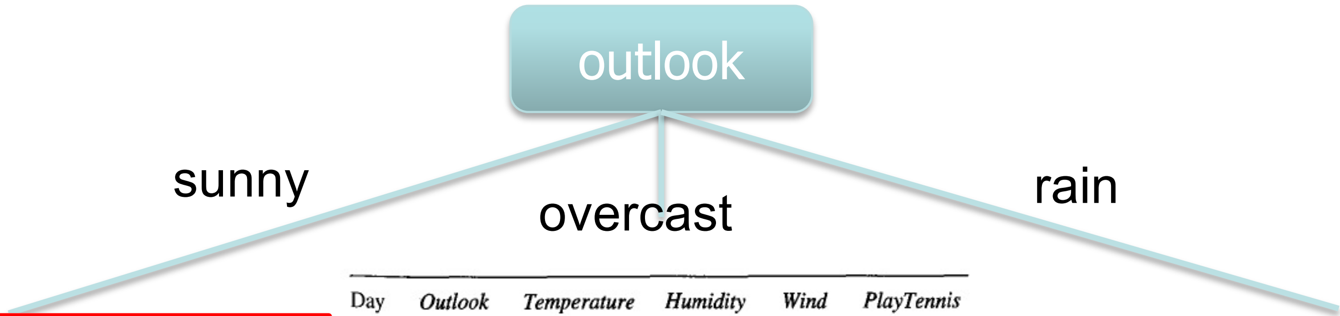
…and we remove outlook as an attribute to consider.



Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

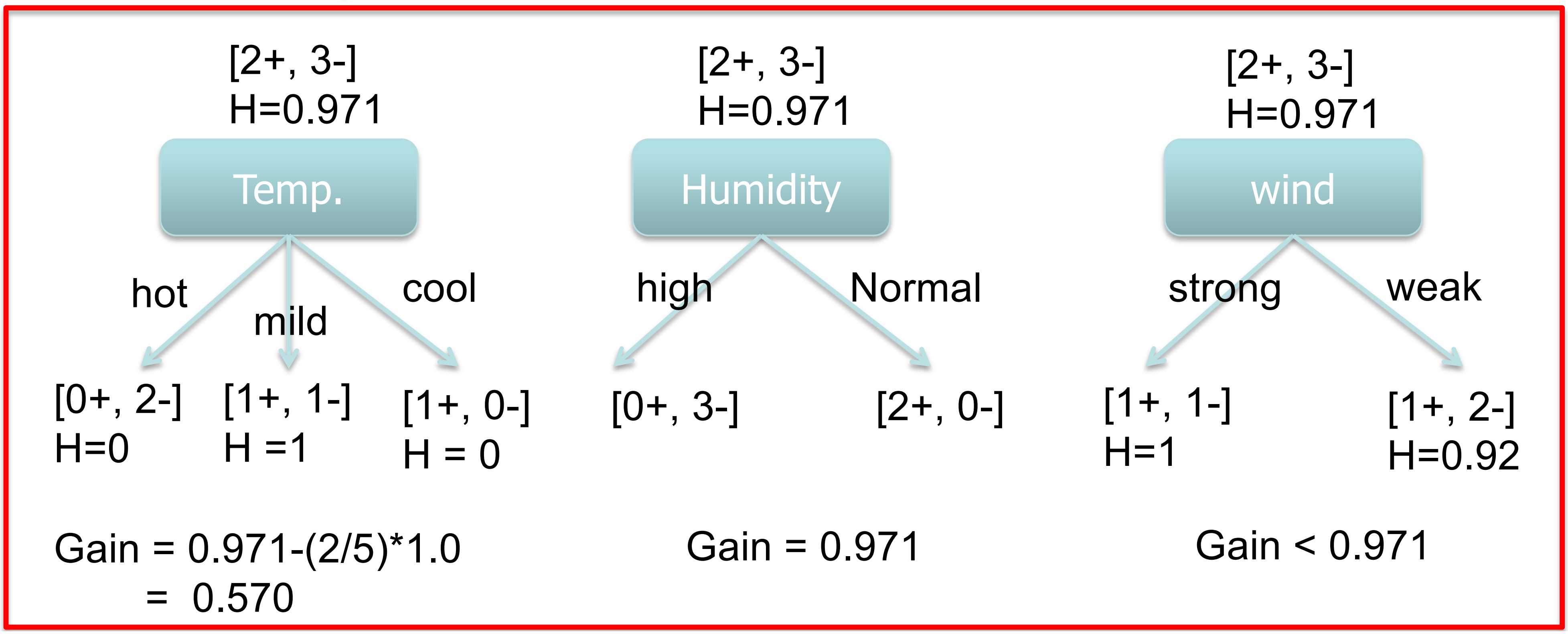
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



outlook

sunny

overcast

rain

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

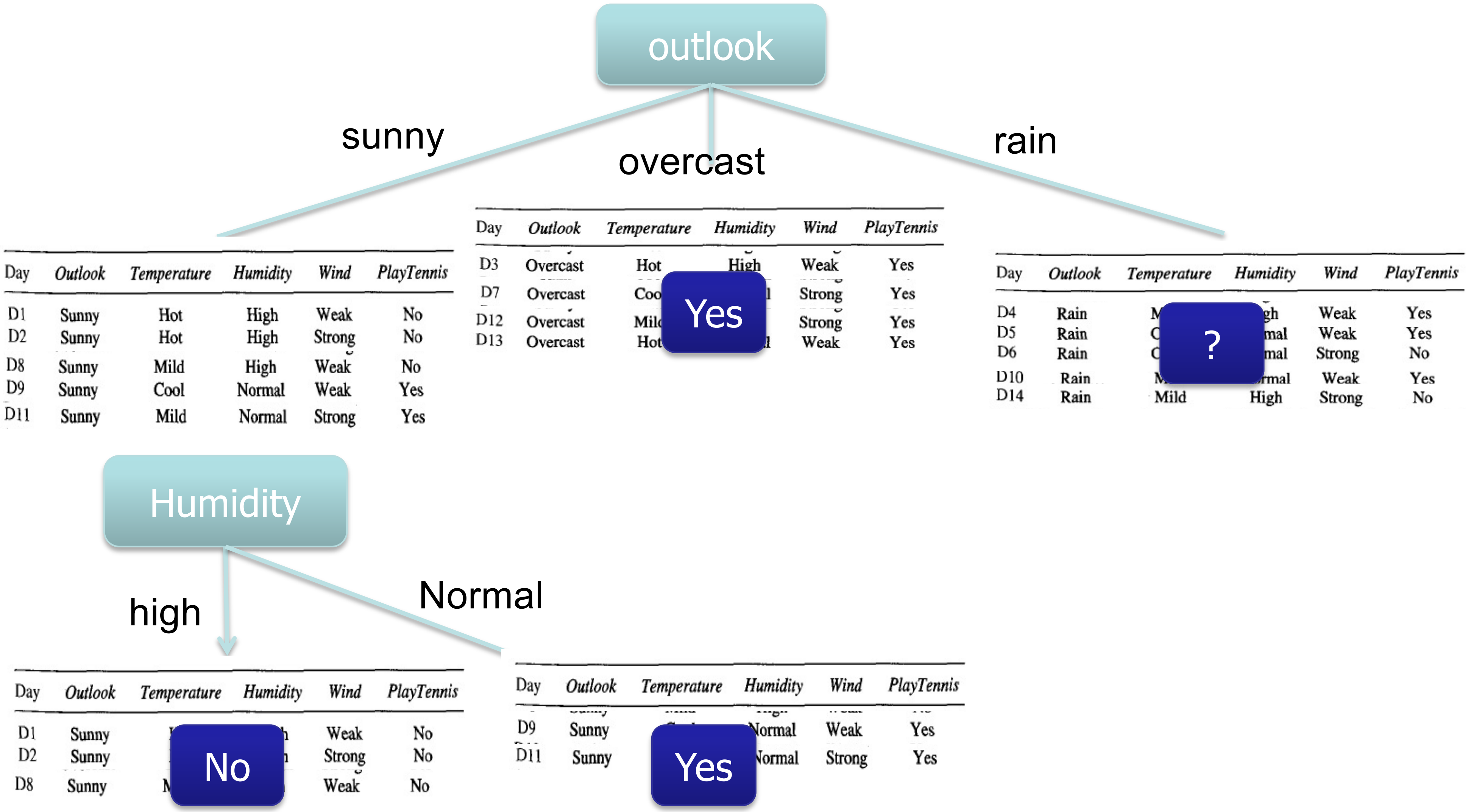
Humidity

high

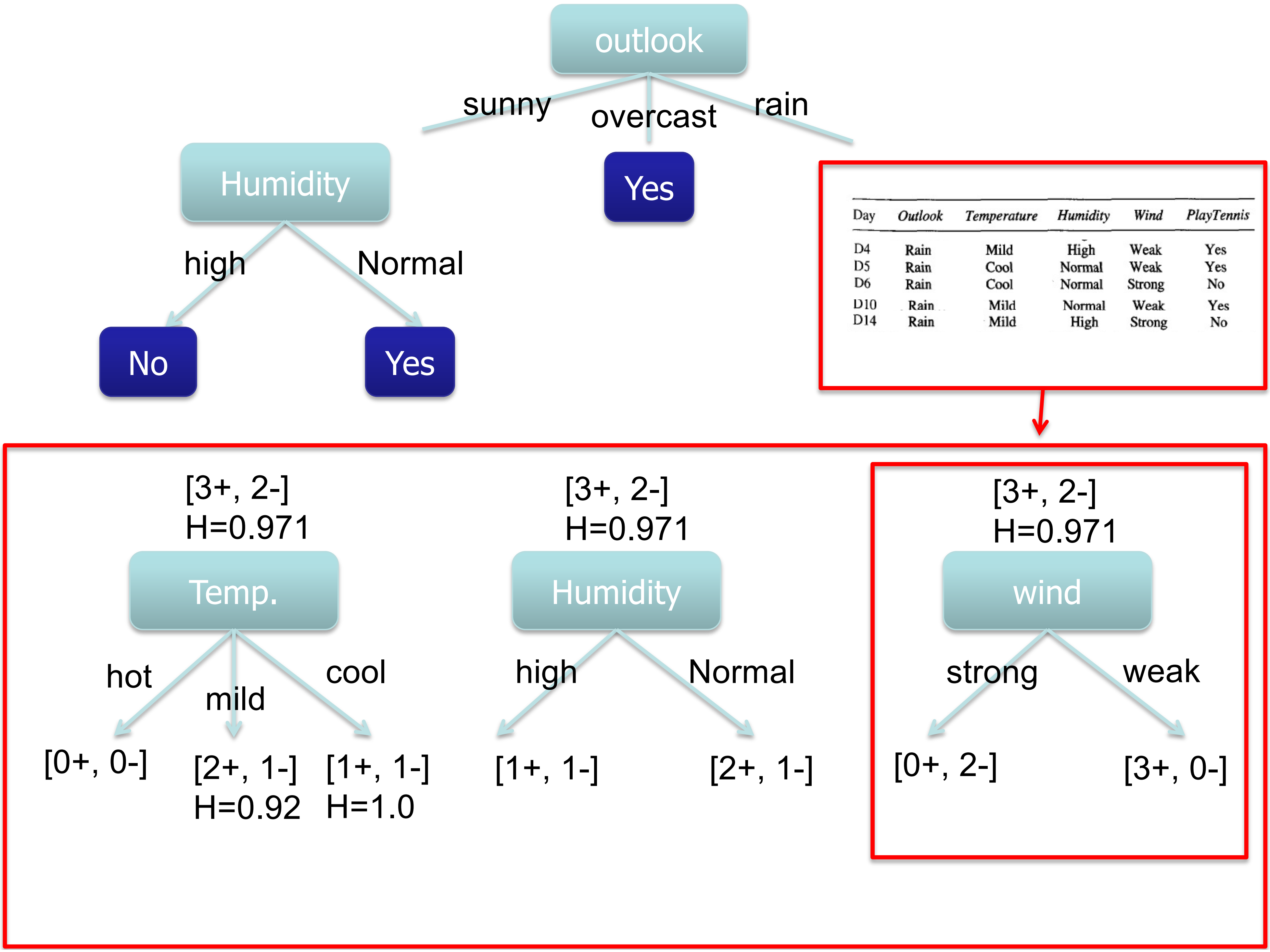
Normal

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

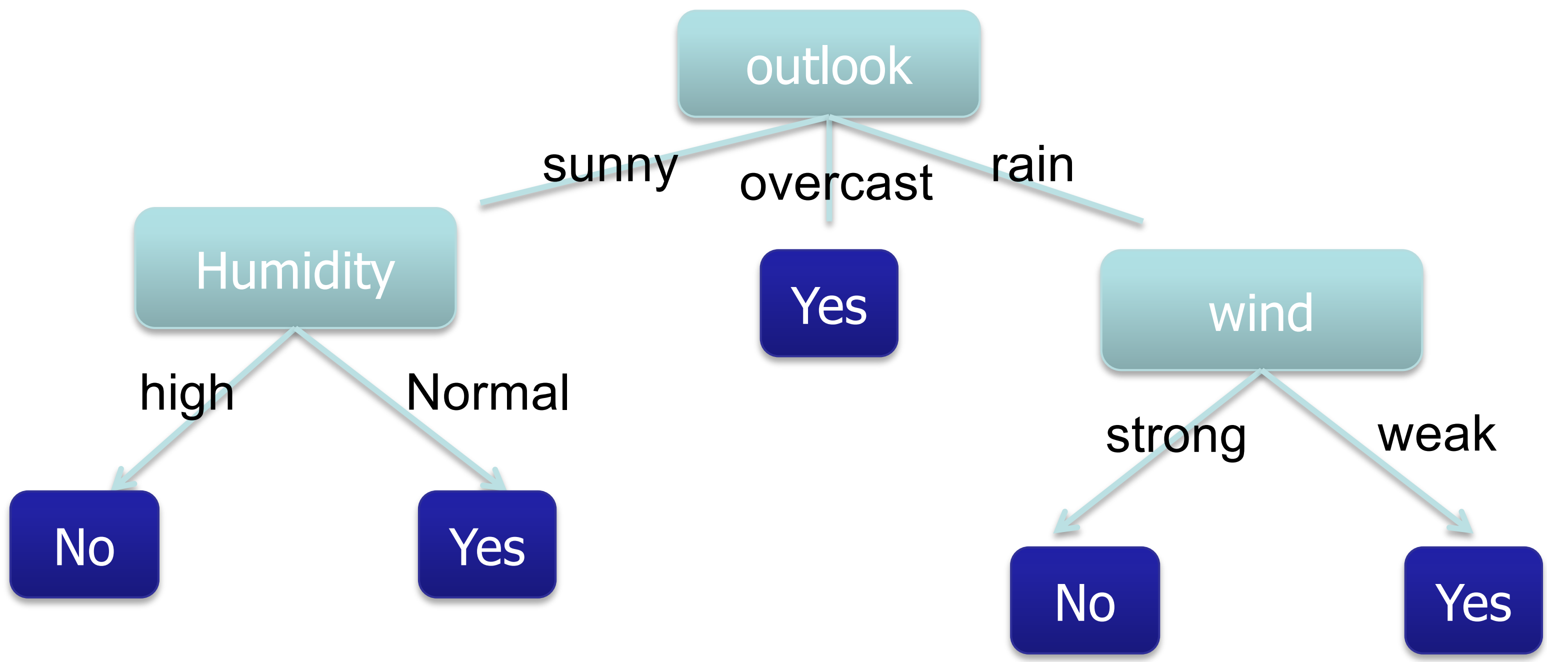


If the data subset at a branch is all one category, you can stop and return that category as the answer.



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

The final tree



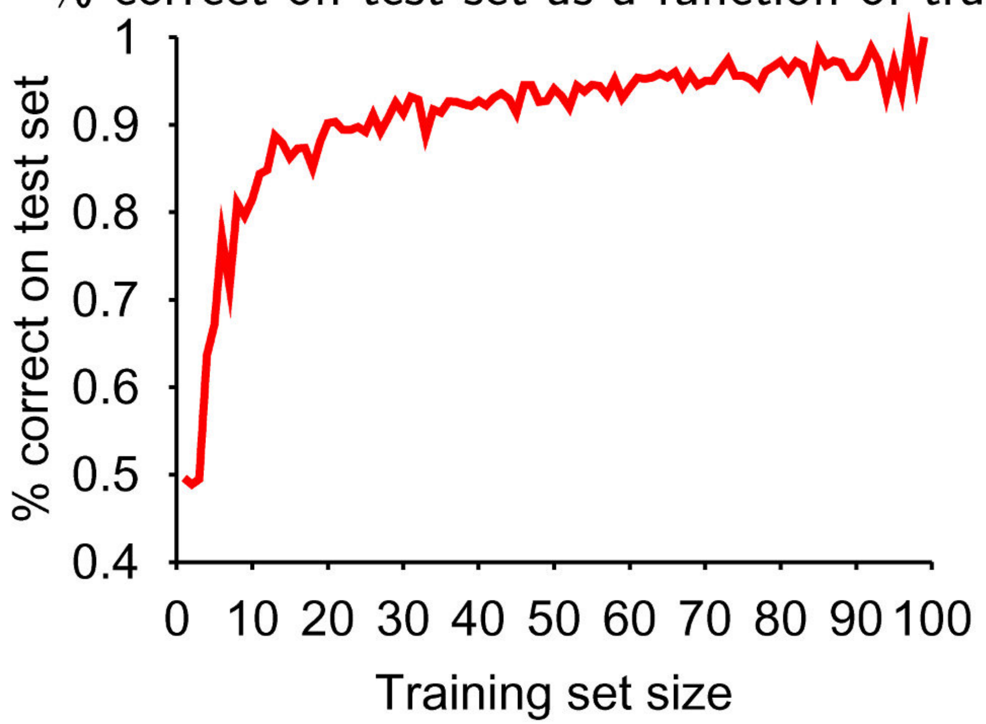
Measuring Performance

How do we know that $h \approx f$? (Hume's **Problem of Induction**)

1) Use theorems of computational/statistical learning theory

2) Try h on a new test set of examples
   (use **same distribution over example space** as training set)

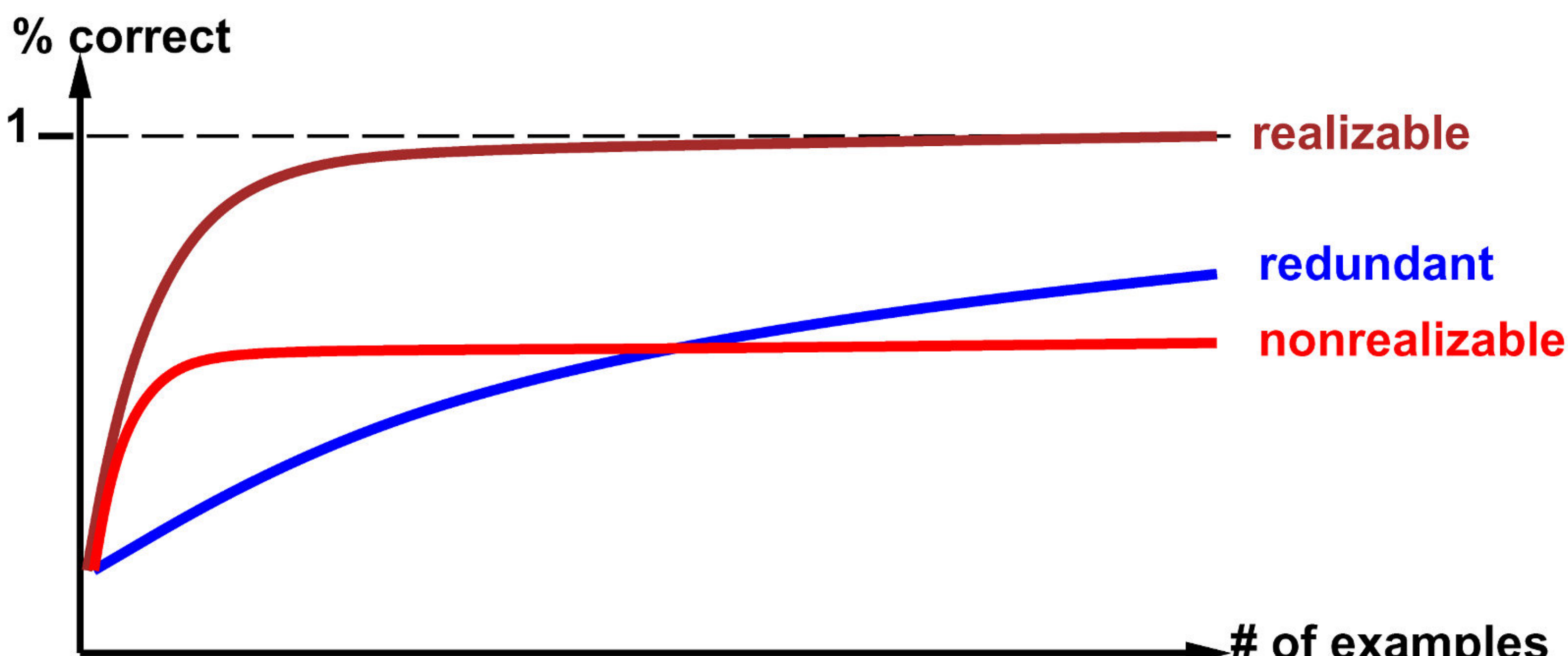
Learning curve = % correct on test set as a function of training set size



What the learning curve tells us

Learning curve depends on

- realizable (can express target function) vs. non-realizable
  non-realizability can be due to missing attributes
  or restricted hypothesis class (e.g., thresholded linear function)
- redundant expressiveness (e.g., loads of irrelevant attributes)



Rule #2 of Machine Learning

The *best* (i.e. the one that generalizes well) hypothesis almost never achieves 100% accuracy on the training data.

(Rule #1 was: you can't learn anything without inductive bias)

Avoiding Overfitting

- Approaches
  - Stop splitting when information gain is low or when split is not statistically significant.
  - Grow full tree and then **prune** it when done

- How to pick the “best” tree?
  - Performance on training data?
  - Performance on validation data?
  - Complexity penalty?

Reduced Error Pruning

- Split data into a training and a validation set
- Repeat until pruning hurts performance measure
  1. Try removing each leaf node (one by one) and measure the resulting performance on the validation set
  2. Remove the leaf that most improves performance

C4.5 Algorithm

- Builds a decision tree from labeled training data
- Also by Ross Quinlan
- Generalizes ID3 by
  - Allowing continuous value attributes
  - Allows missing attributes in examples
  - Prunes tree after building to improve generality

Rule post pruning

- Used in C4.5
- Steps
  1. Build the decision tree
  2. Convert it to a set of logical rules
  3. Prune each rule independently
  4. Sort rules into desired sequence for use

Take away about decision trees

- Used as classifiers
- Supervised learning algorithms (ID3, C4.5)
- (mostly) Batch processing
- Good for situations where
  - The classification categories are finite
  - The data can be represented as vectors of attributes
  - You want to be able to UNDERSTAND how the classifier makes its choices