Supervised Learning (contd)
Decision Trees

Mausam

(based on slides by UW-AI faculty)

Decision Trees

To play or not to play?

http://www.sfgate.com/blogs/images/sfgate/sgreen/2007/09/05/2240773250x321.jpg

Example data for learning the concept "Good day for tennis"

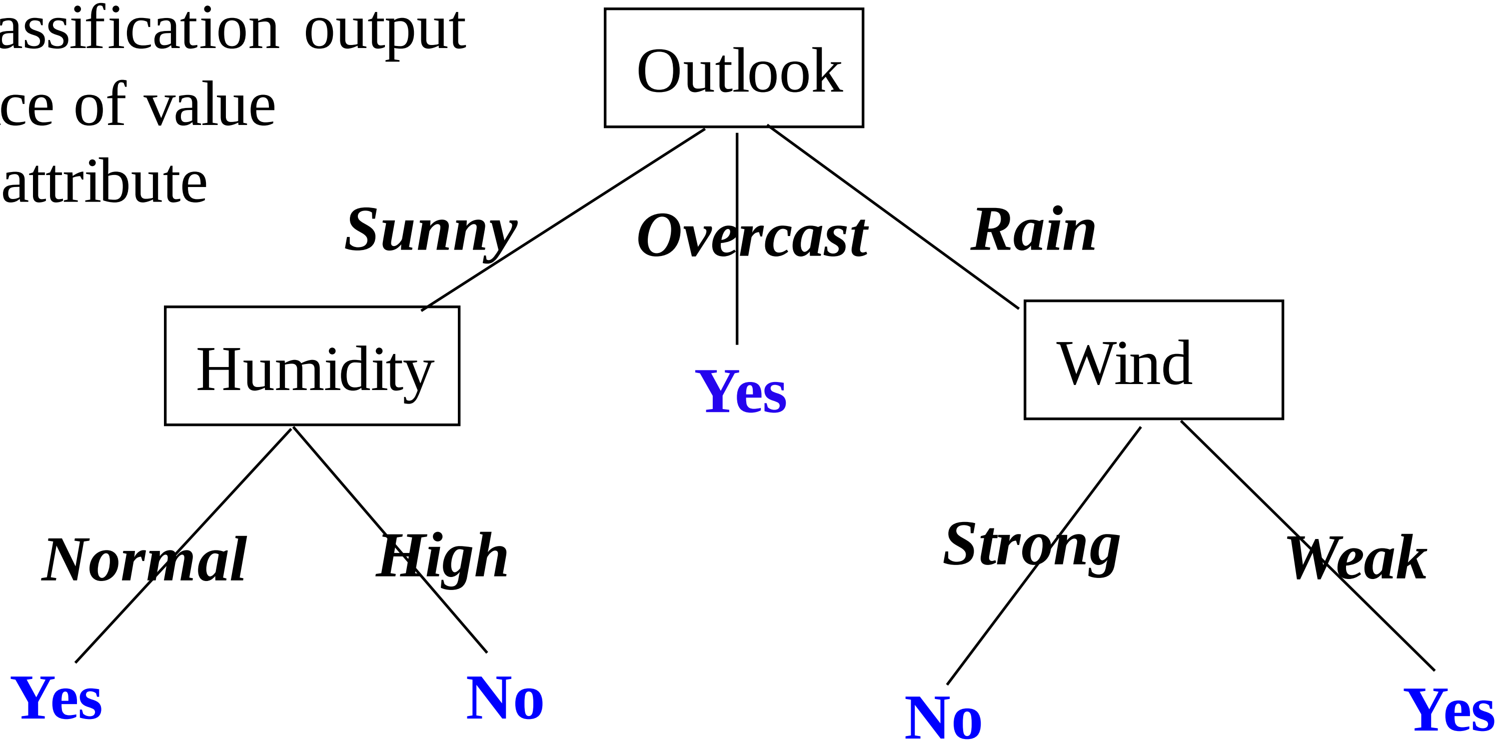
Day	Outlook	Humid	Wind	**PlayTennis**?
d1	s	h	w	n
d2	s	h	s	n
d3	o	h	w	y
d4	r	h	w	y
d5	r	n	w	y
d6	r	n	s	y
d7	o	n	s	y
d8	s	h	w	n
d9	s	n	w	y
d10	r	n	w	y
d11	s	n	s	y
d12	o	h	s	y
d13	o	n	w	y
d14	r	h	s	n

- Outlook = sunny, overcast, rain
- Humidity = high, normal
- Wind = weak, strong

A Decision Tree for the Same Data

Decision Tree for "PlayTennis?"

Leaves = classification output
Arcs = choice of value
for parent attribute



Decision tree is equivalent to logic in disjunctive normal form

PlayTennis $\Leftrightarrow$ (Sunny $\wedge$ Normal) $\vee$ Overcast $\vee$ (Rain $\wedge$ Weak)

Decision Trees

Input: Description of an object or a situation through a set of **attributes**

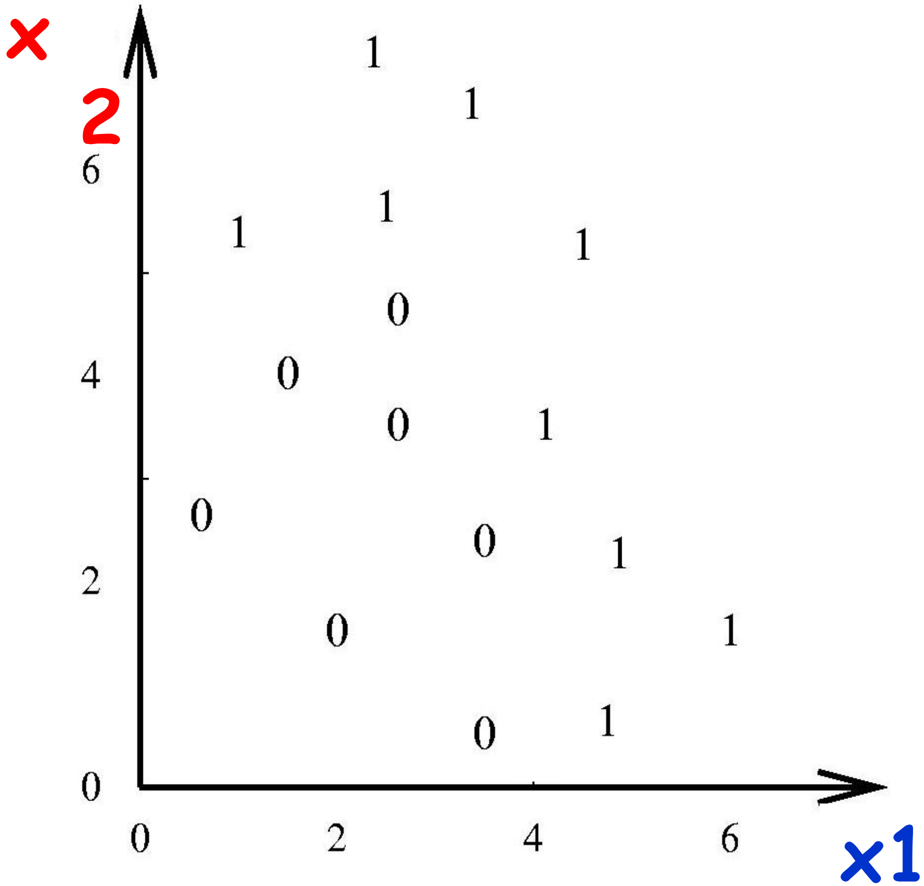
Output: a **decision** that is the predicted output value for the input

Both **input and output can be discrete or continuous**

Discrete-valued functions lead to **classification problems**

Example: Decision Tree for Continuous Valued Features and Discrete Output

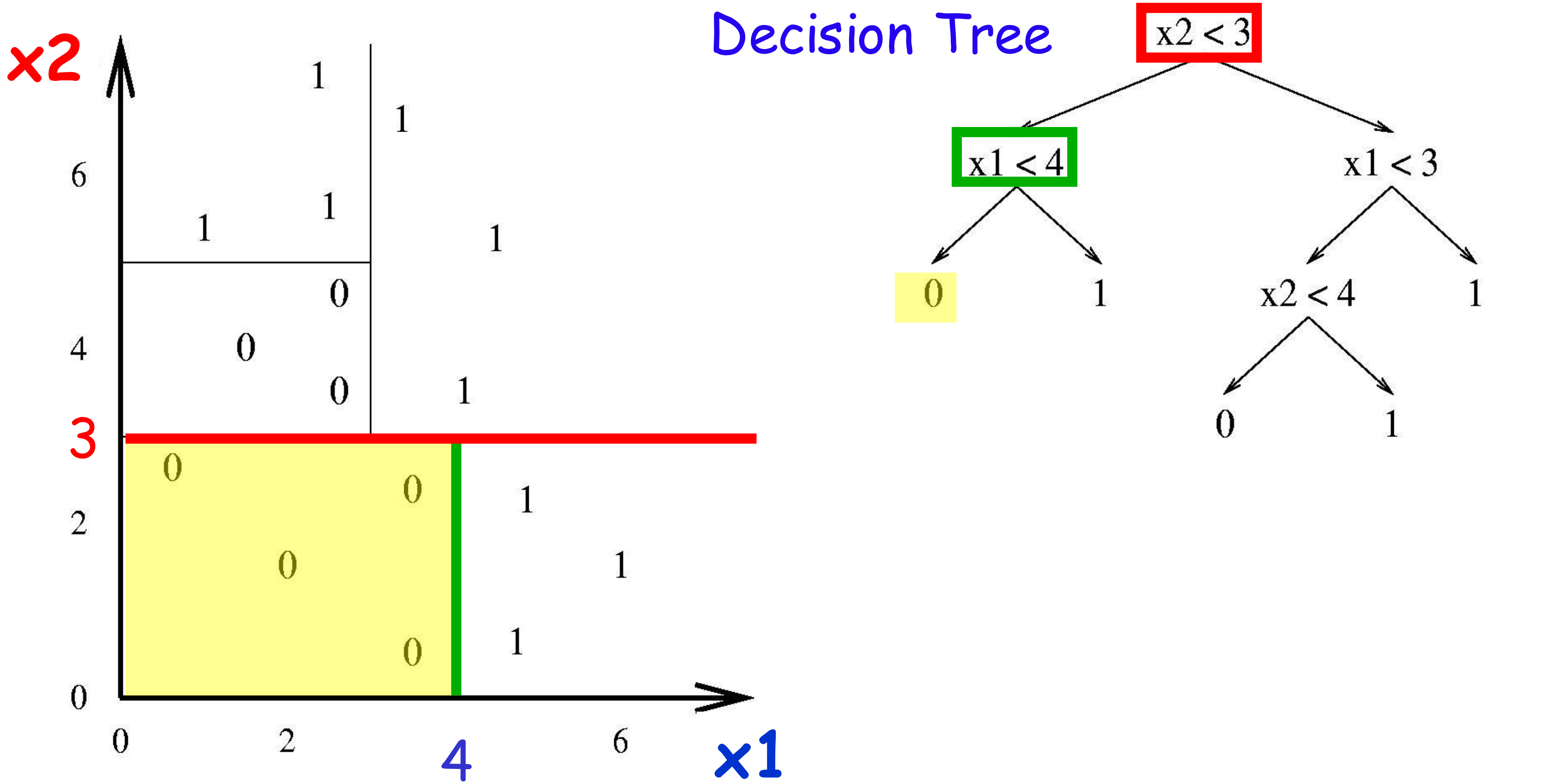
Input real number attributes (x1,x2), Classification output: 0 or 1



How do we branch using attribute values x1 and x2 to partition the space correctly?

Example: Classification of Continuous Valued Inputs

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

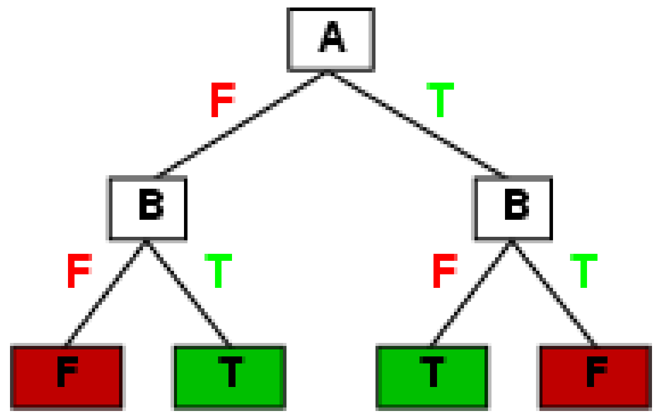


Expressiveness of Decision Trees

Decision trees can express any function of the input attributes.

E.g., for Boolean functions, truth table row = 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 with one path to leaf for each example

- But most likely won't generalize to new examples

Prefer to find more compact decision trees

Learning Decision Trees

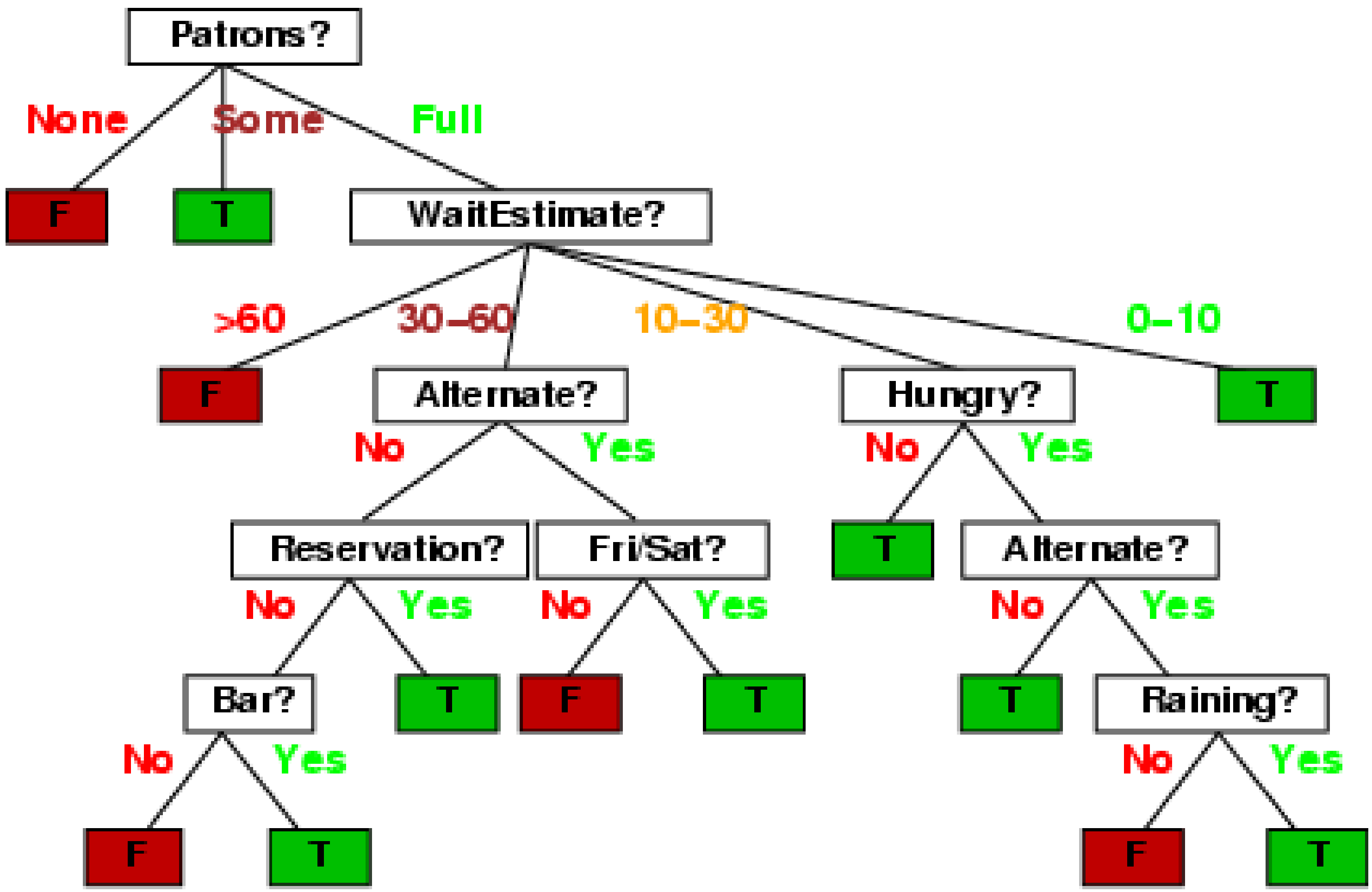
Example: When should I wait for a table at a restaurant?

Attributes (features) relevant to *Wait?* decision:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Example Decision tree

A decision tree for *Wait?* based on personal "rules of thumb":



Input Data for Learning

Past examples when I did/did not wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
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 Learning

Aim: find a small tree consistent with training examples

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

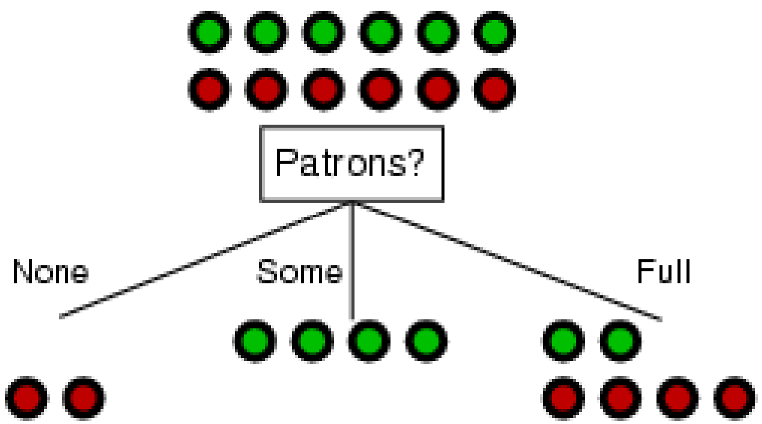
function DTL(*examples*, *attributes*, *default*) **returns** a decision tree

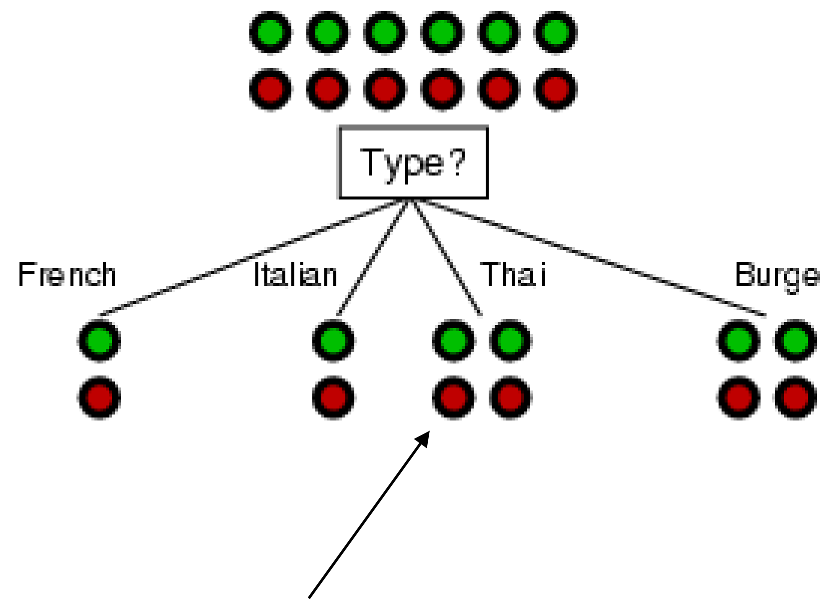
    if *examples* is empty **then return** *default*
    else if all *examples* have the same classification **then return** the classification
    else if *attributes* is empty **then return** MODE(*examples*)
    else
        $best \leftarrow$ CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        $tree \leftarrow$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

Choosing an attribute to split on

Idea: a good attribute should reduce uncertainty

- E.g., splits the examples into subsets that are (ideally) "all positive" or "all negative"





Patrons? is a better choice

For *Type?*, to wait or not to wait is still at 50%

How do we quantify uncertainty?

http://a.espncdn.com/media/ten/2006/0306/photo/g_mcenroe_195.jpg

Using information theory to quantify uncertainty

Entropy measures the amount of uncertainty in a probability distribution

Entropy (or Information Content) of an answer to a question with possible answers $v_1, \ldots, v_n$:

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

Using information theory

Imagine we have p examples with Wait = True (positive) and n examples with Wait = false (negative).

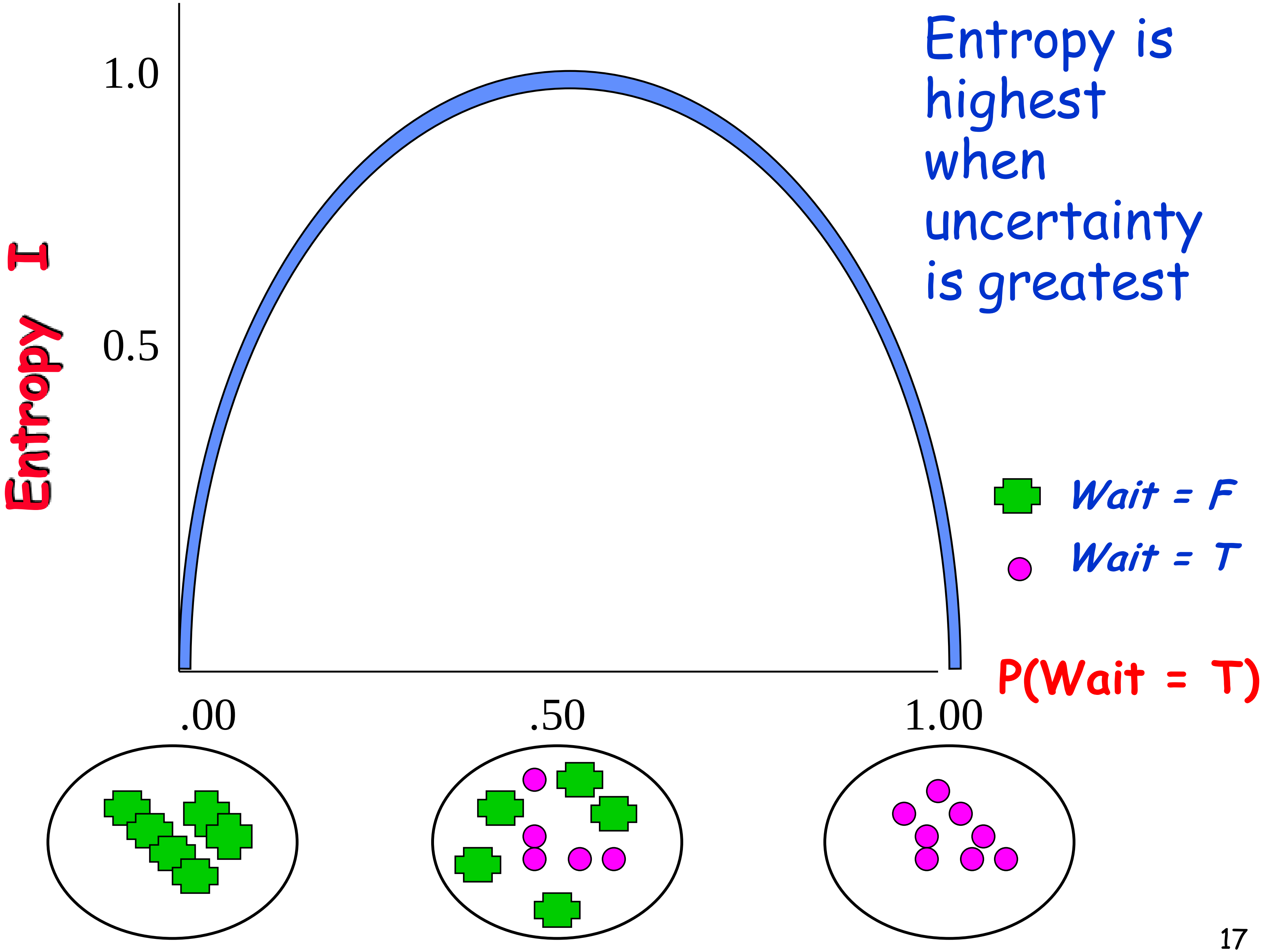
Our best estimate of the probabilities of Wait = true or false is given by:

$$P(true) \approx p / p + n$$

$$p(false) \approx n / p + n$$

Hence the entropy of Wait is given by:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2 \frac{p}{p+n} - \frac{n}{p+n}\log_2 \frac{n}{p+n}$$



Entropy is highest when uncertainty is greatest

Choosing an attribute to split on

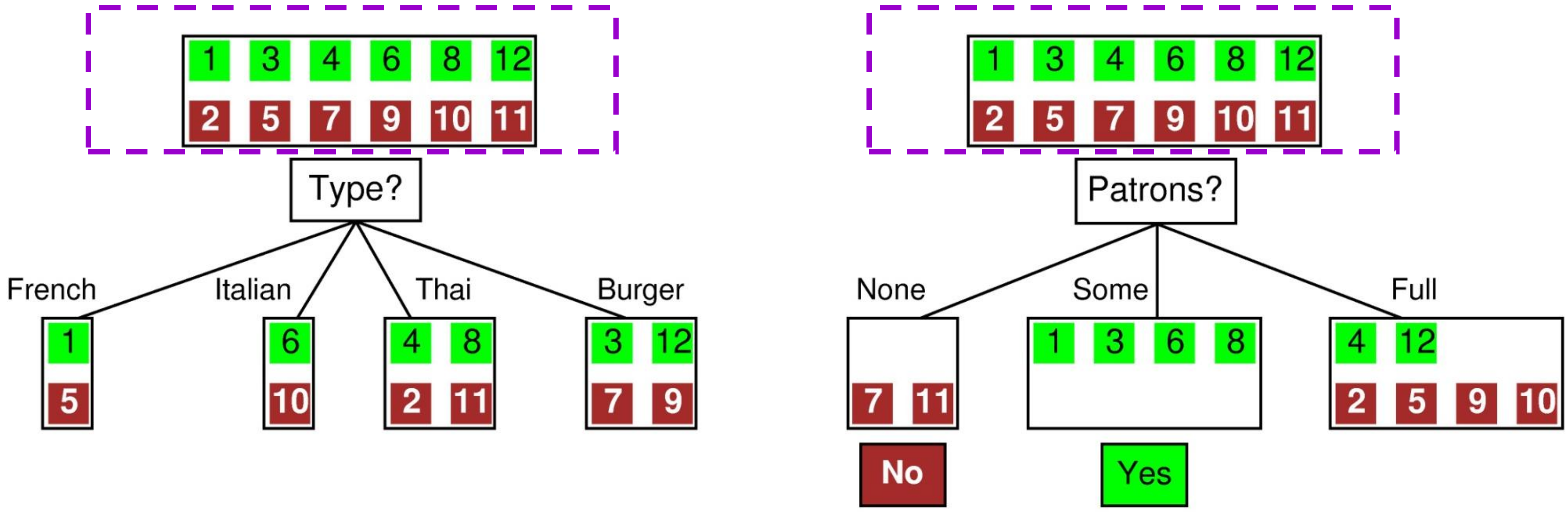
Idea: a good attribute should reduce uncertainty and result in "gain in information"

How much information do we gain if we disclose the value of some attribute?

Answer:

uncertainty before – uncertainty after

Back at the Restaurant



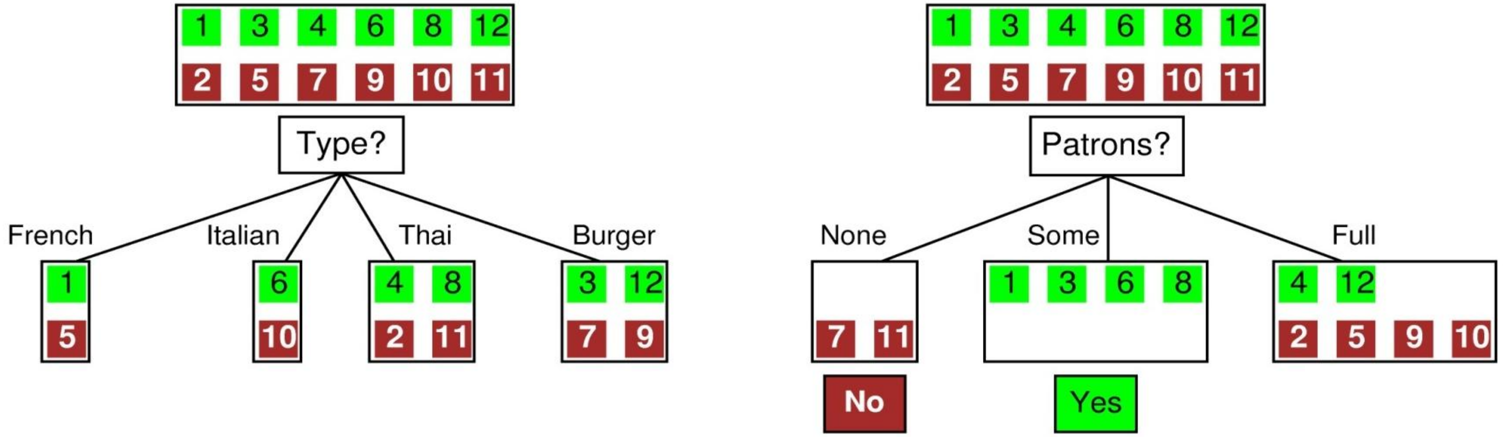
Before choosing an attribute:

Entropy = - 6/12 log(6/12) – 6/12 log(6/12)

= - log(1/2) = log(2) = 1 bit

There is "1 bit of information to be discovered"

Back at the Restaurant



If we choose Type: Go along branch "French": we have entropy = 1 bit; similarly for the others.

Information gain = 1-1 = 0 along any branch

If we choose Patrons:

In branch "None" and "Some", entropy = 0

For "Full", entropy = -2/6 log(2/6)-4/6 log(4/6) = 0.92

Info gain = (1-0) or (1-0.92) bits > 0 in both cases

So choosing Patrons gains more information!

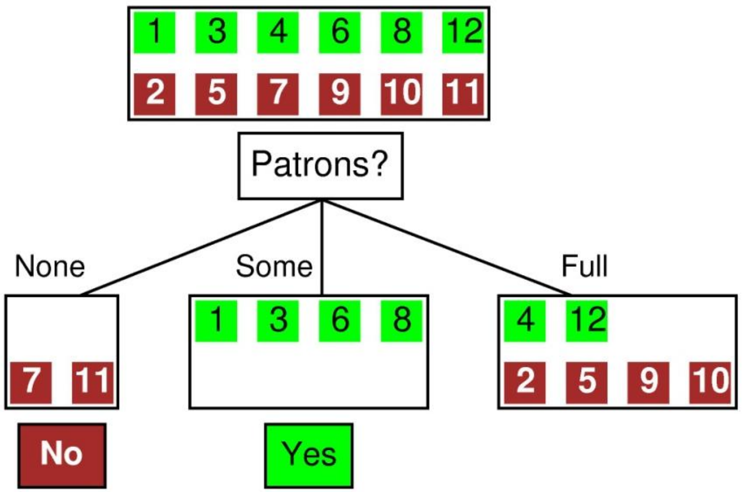
Entropy across branches

- How do we combine entropy of different branches?
- Answer: Compute average entropy
- Weight entropies according to probabilities of branches

  2/12 times we enter "None", so weight for "None" = 1/6

  "Some" has weight: 4/12 = 1/3

  "Full" has weight 6/12 = ½



$$AvgEntropy = \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} Entropy\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

weight for each branch

entropy for each branch

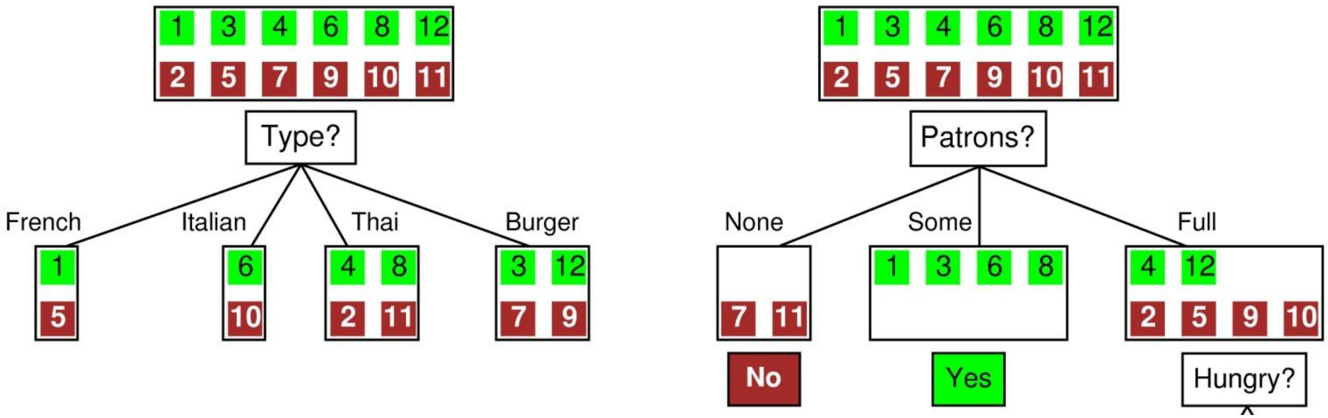
Information gain

Information Gain (IG) or reduction in entropy from using attribute A:

IG(A) = Entropy before – AvgEntropy after choosing A

Choose the attribute with the largest IG

Information gain in our example



$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .541 \text{ bits}$$

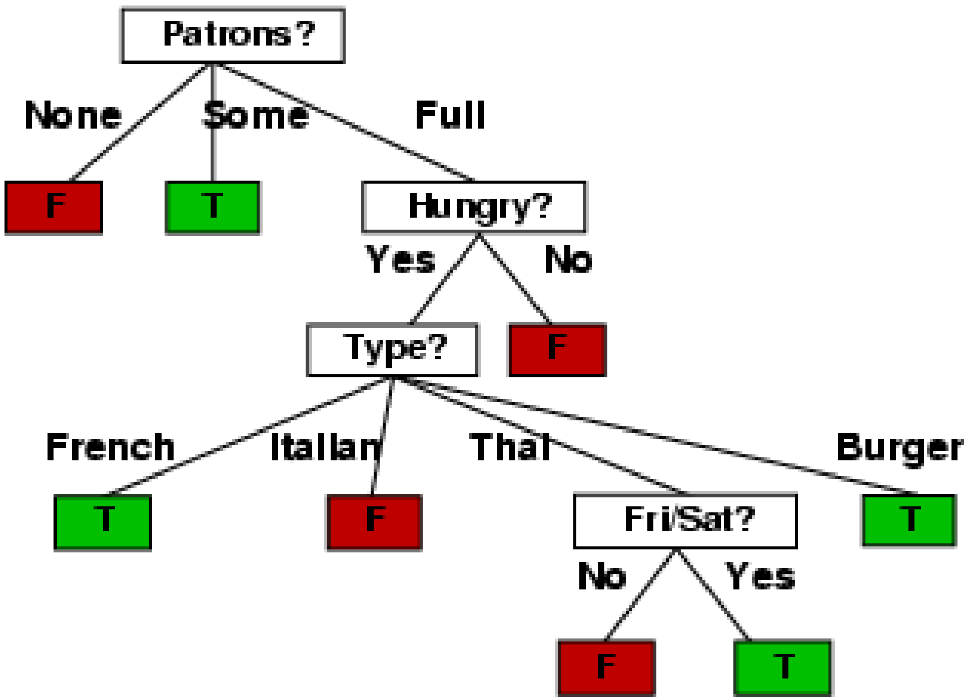
$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes

$\Rightarrow$ DTL algorithm chooses *Patrons* as the root

Should I stay or should I go? Learned Decision Tree

Decision tree learned from the 12 examples:



Substantially simpler than "rules-of-thumb" tree

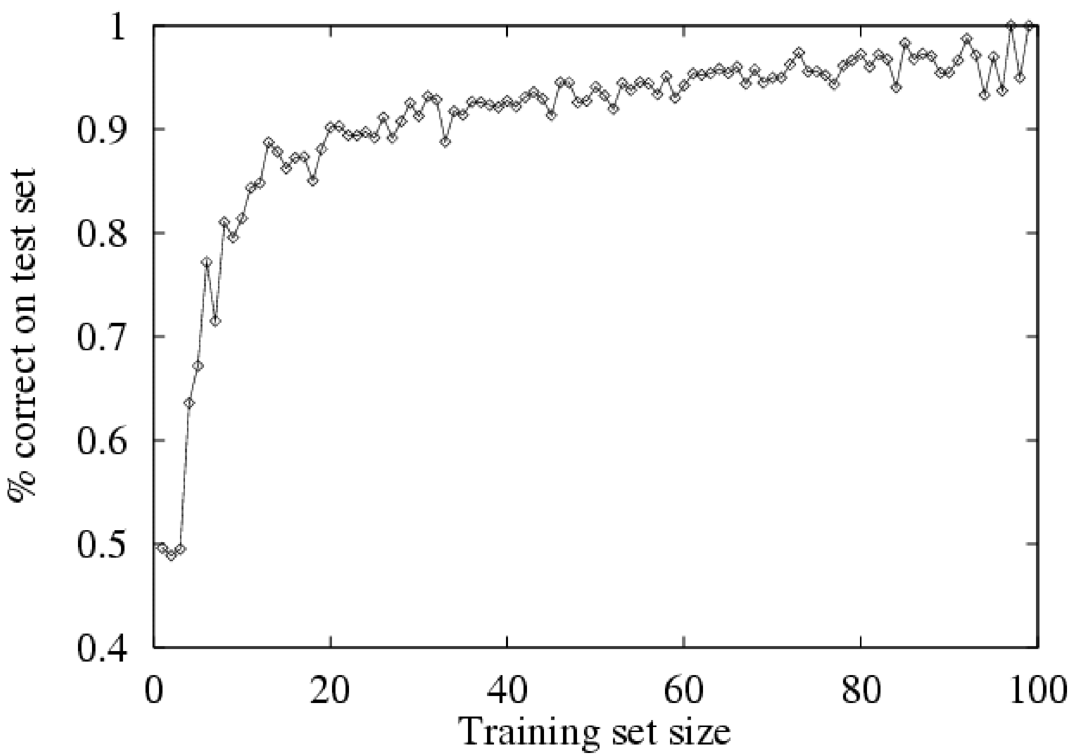
- more complex hypothesis not justified by small amount of data

Performance Evaluation

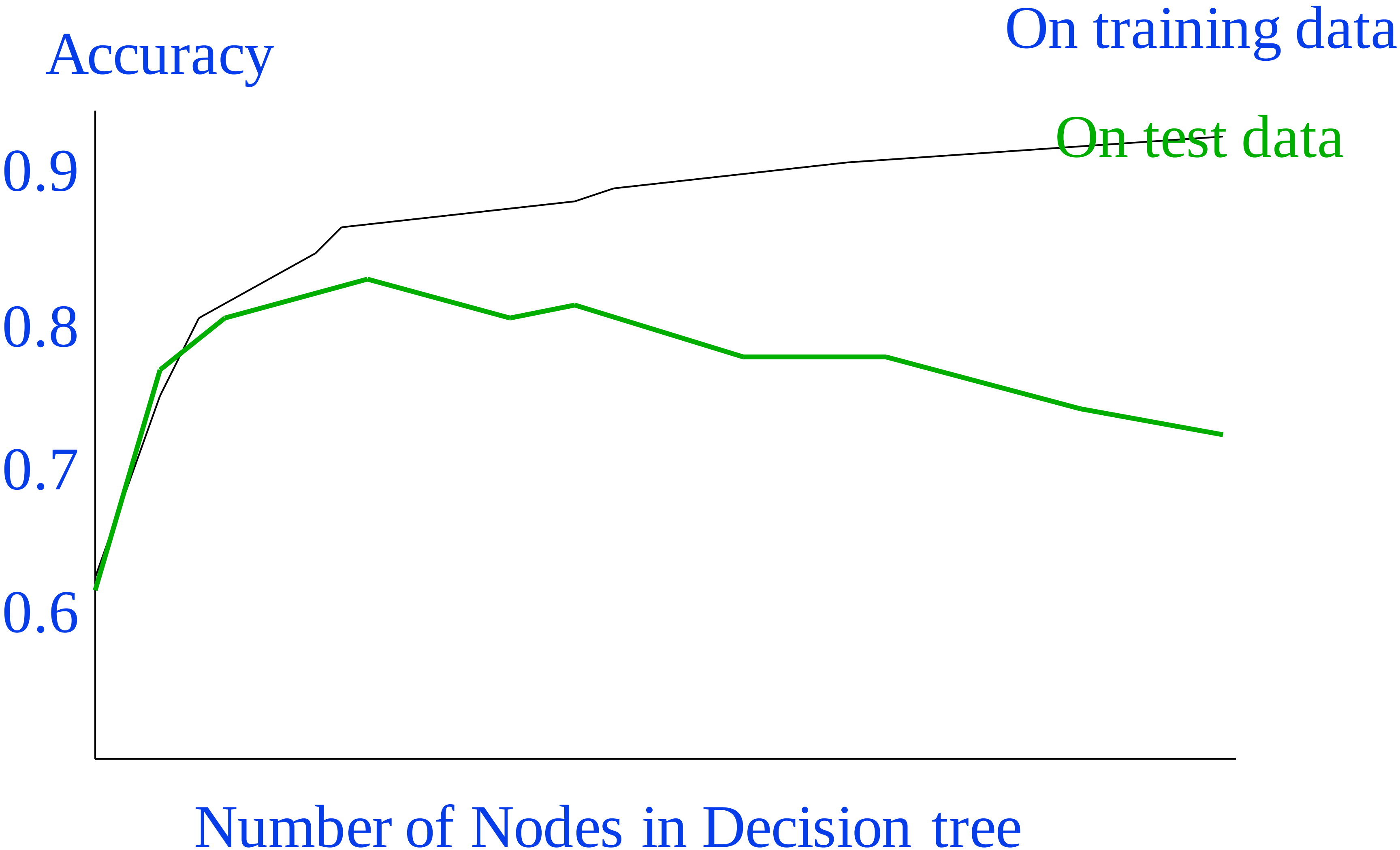
How do we know that the learned tree $h \approx f$?

Answer: Try h on a new test set of examples

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



Overfitting



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

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Rule #2 of Machine Learning

The *best* hypothesis almost never achieves 100% accuracy on the training data.

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

Avoiding Overfitting

How can we avoid overfitting?

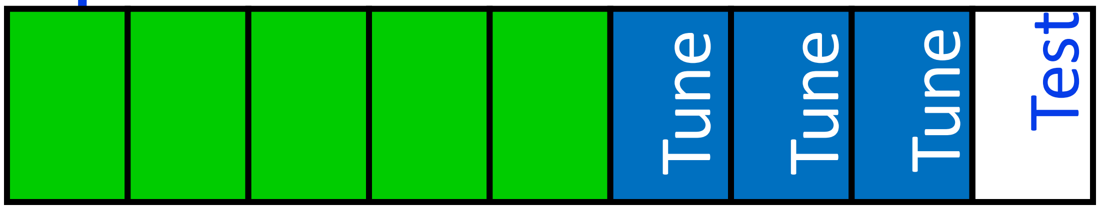
- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

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

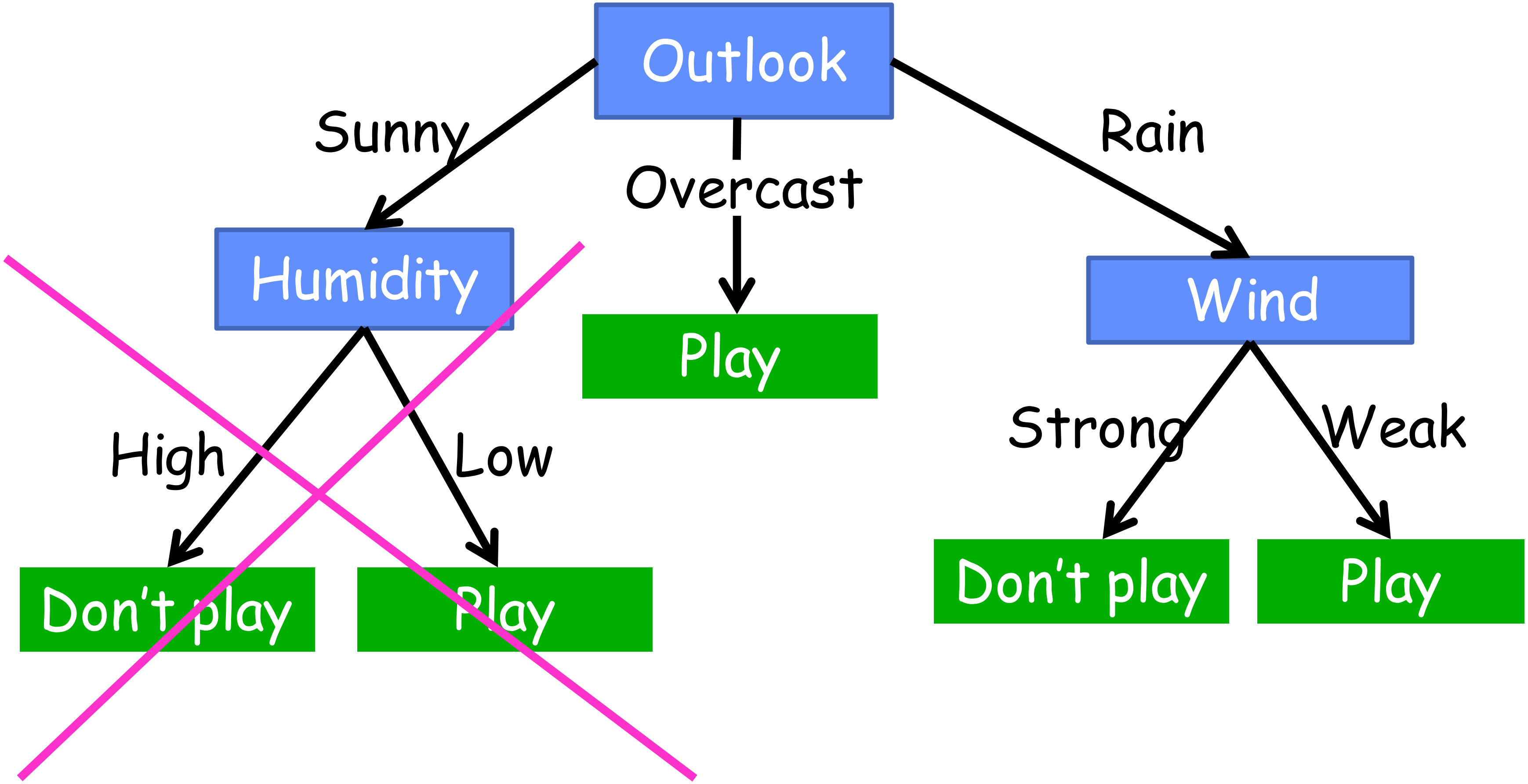
Split data into train and validation set



Repeat until pruning is harmful

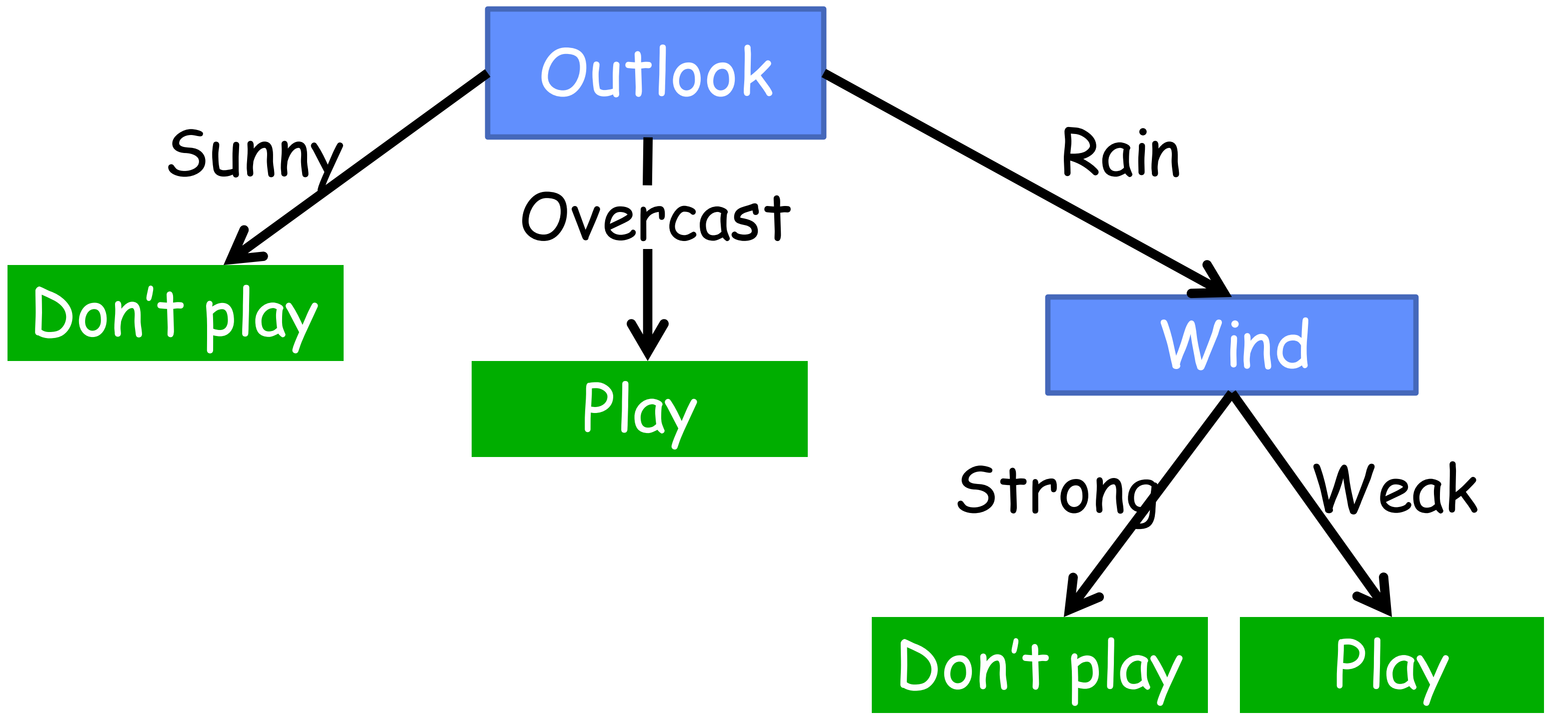
- Remove each subtree and replace it with majority class and evaluate on validation set
- Remove subtree that leads to largest gain in accuracy

Reduced Error Pruning Example



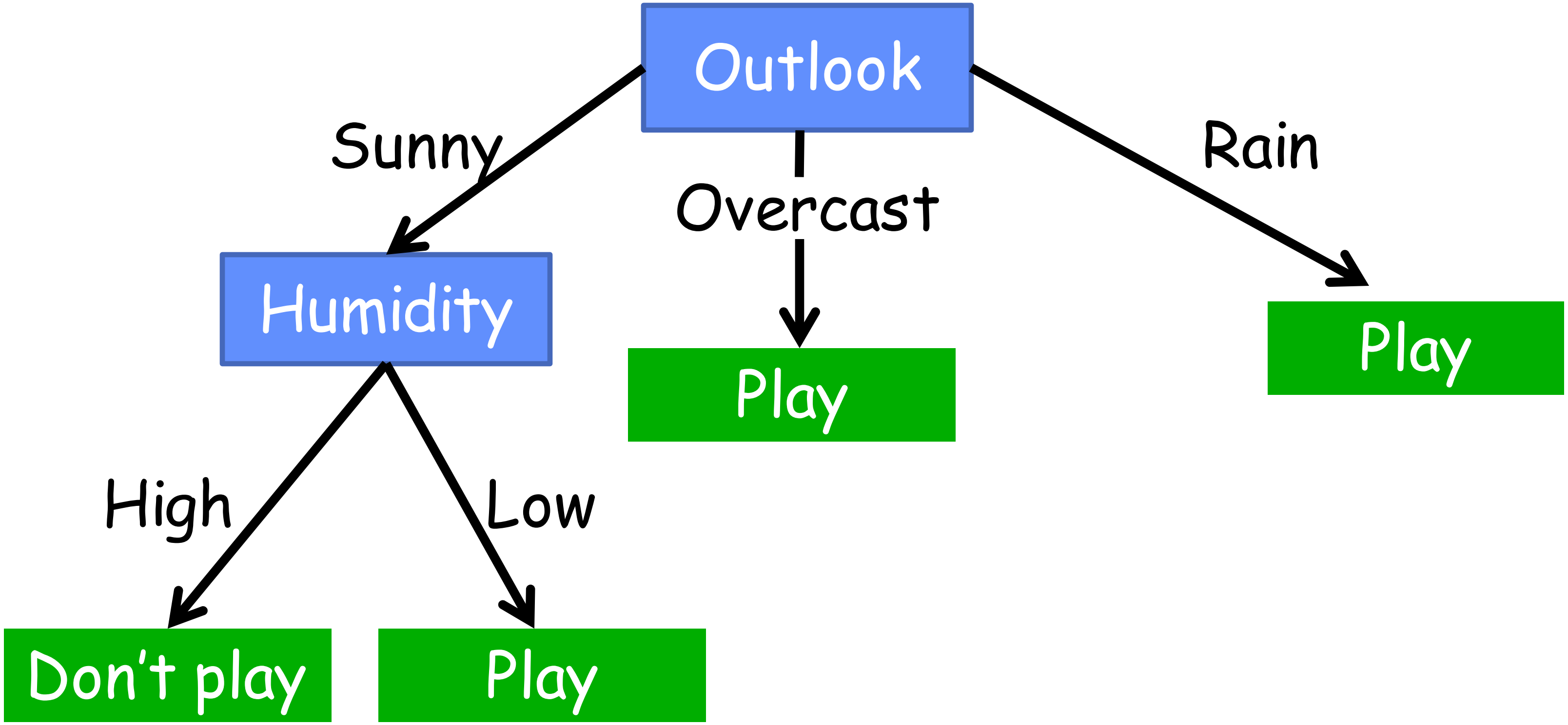
Validation set accuracy = 0.75

Reduced Error Pruning Example



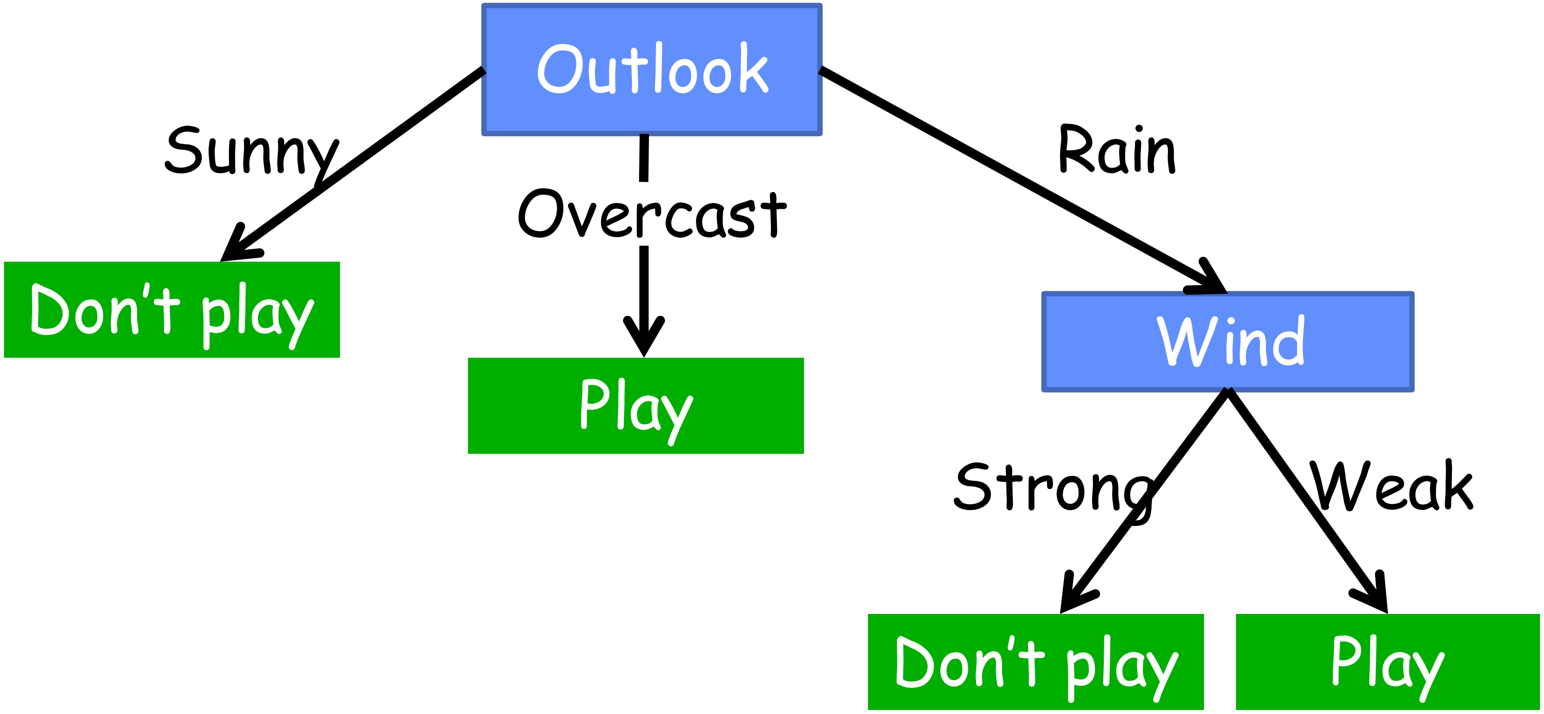
Validation set accuracy = 0.80

Reduced Error Pruning Example



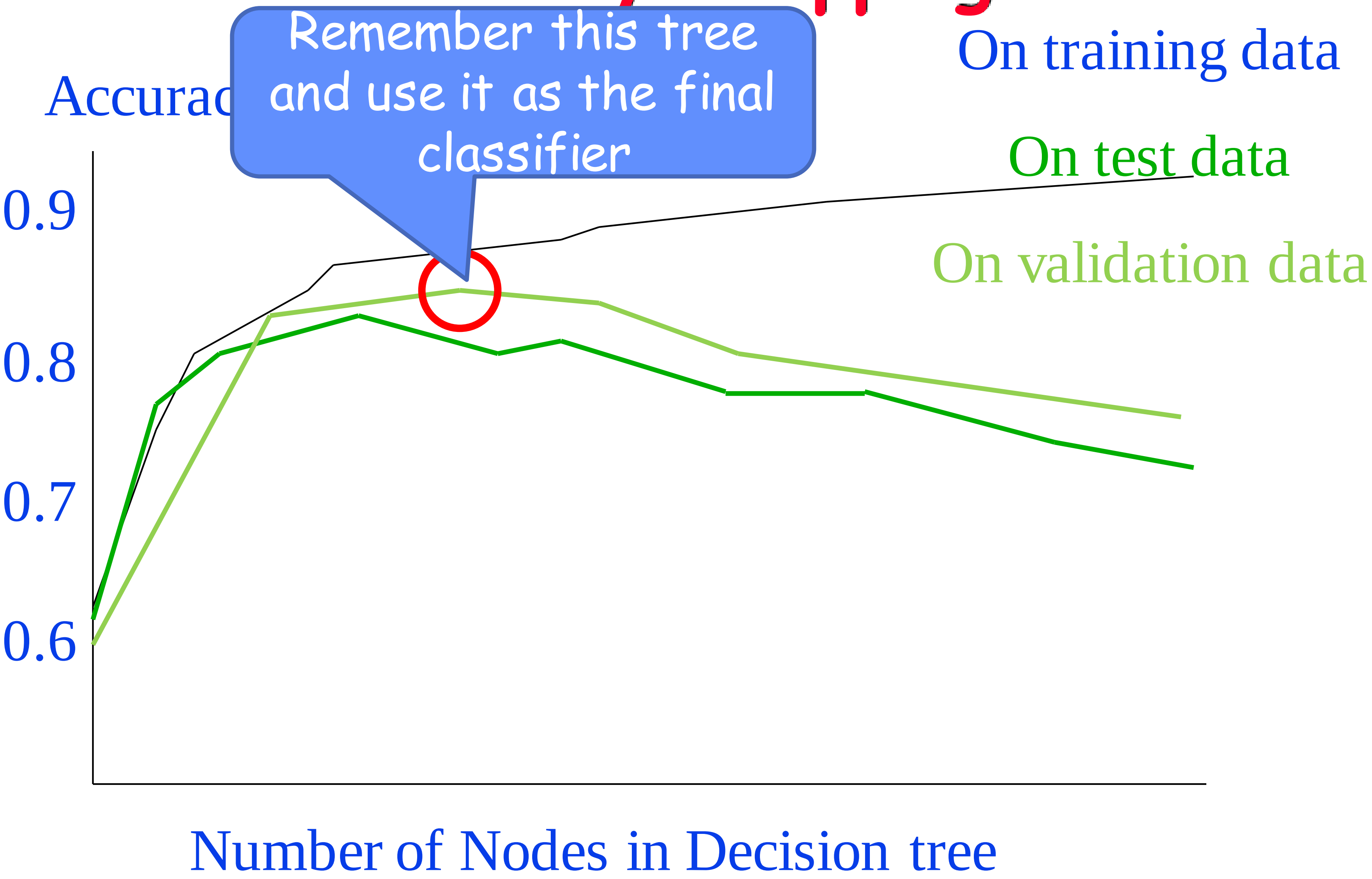
Validation set accuracy = 0.70

Reduced Error Pruning Example



Use this as final tree

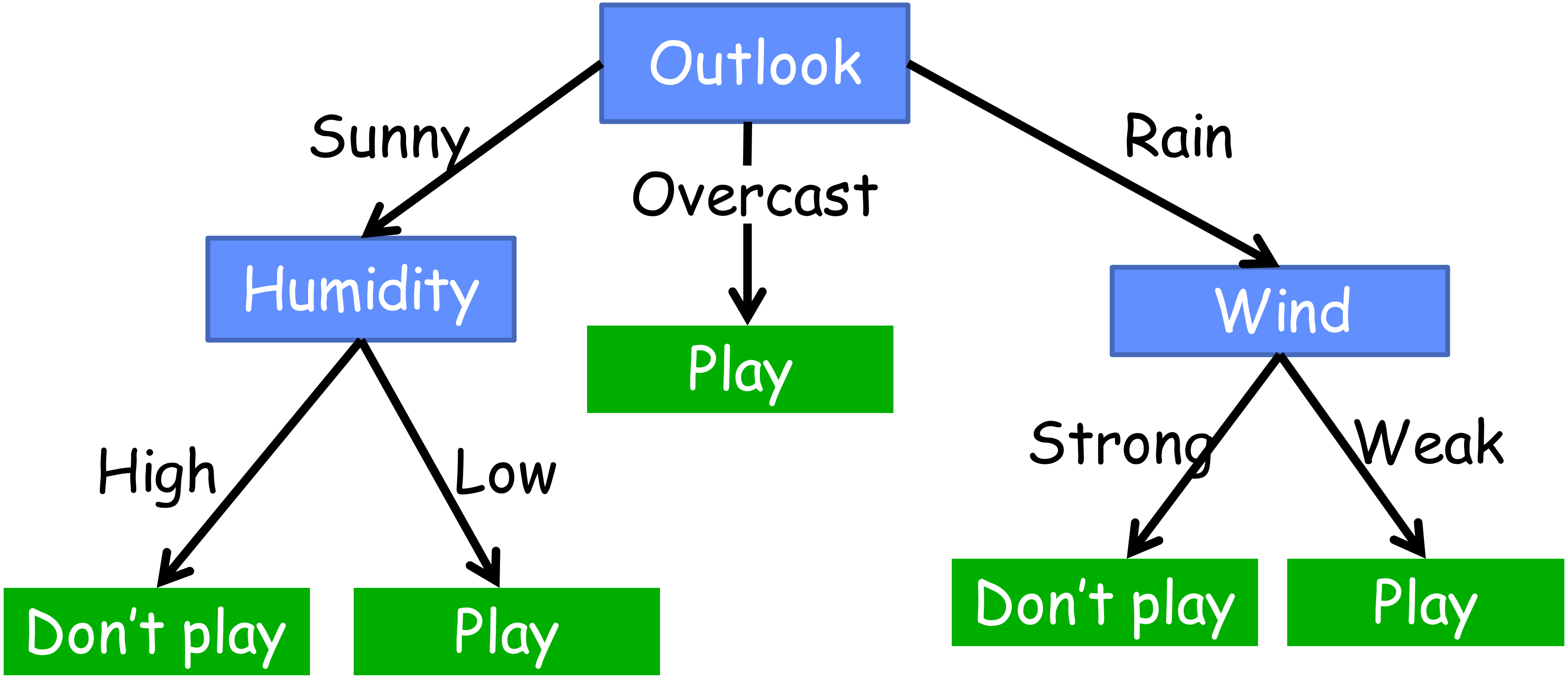
Early Stopping



Post Rule Pruning

- Split data into train and validation set
- Prune each rule independently
  - Remove each pre-condition and evaluate accuracy
  - Pick pre-condition that leads to largest improvement in accuracy
- Note: ways to do this using training data and statistical tests

Conversion to Rule



Outlook = Sunny $\wedge$ Humidity = High $\Rightarrow$ Don’t play

Outlook = Sunny $\wedge$ Humidity = Low $\Rightarrow$ Play

Outlook = Overcast $\Rightarrow$ Play

Scaling Up

- ID3, C4.5, etc. assume data fits in main memory (OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data (OK for up to millions of examples)
- VFDT: at most one sequential scan (OK for up to billions of examples)

Decision Trees - Strengths

Very Popular Technique

Fast

Useful when

- Instances are attribute-value pairs
- Target Function is discrete
- Concepts are likely to be disjunctions
- Attributes may be noisy

Decision Trees – Weaknesses

Less useful for continuous outputs

Can have difficulty with continuous input features as well…

- E.g., what if your target concept is a circle in the x1, x2 plane?
  - Hard to represent with decision trees…
  - Very simple with instance-based methods we'll discuss later…