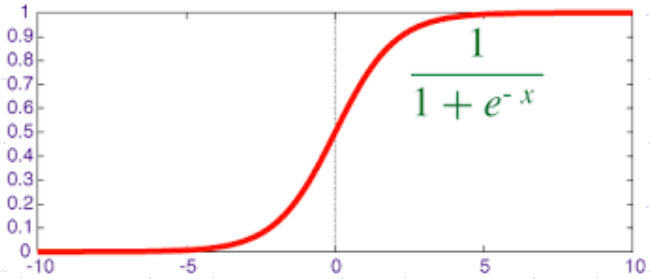
Can we use this type of model for classification?

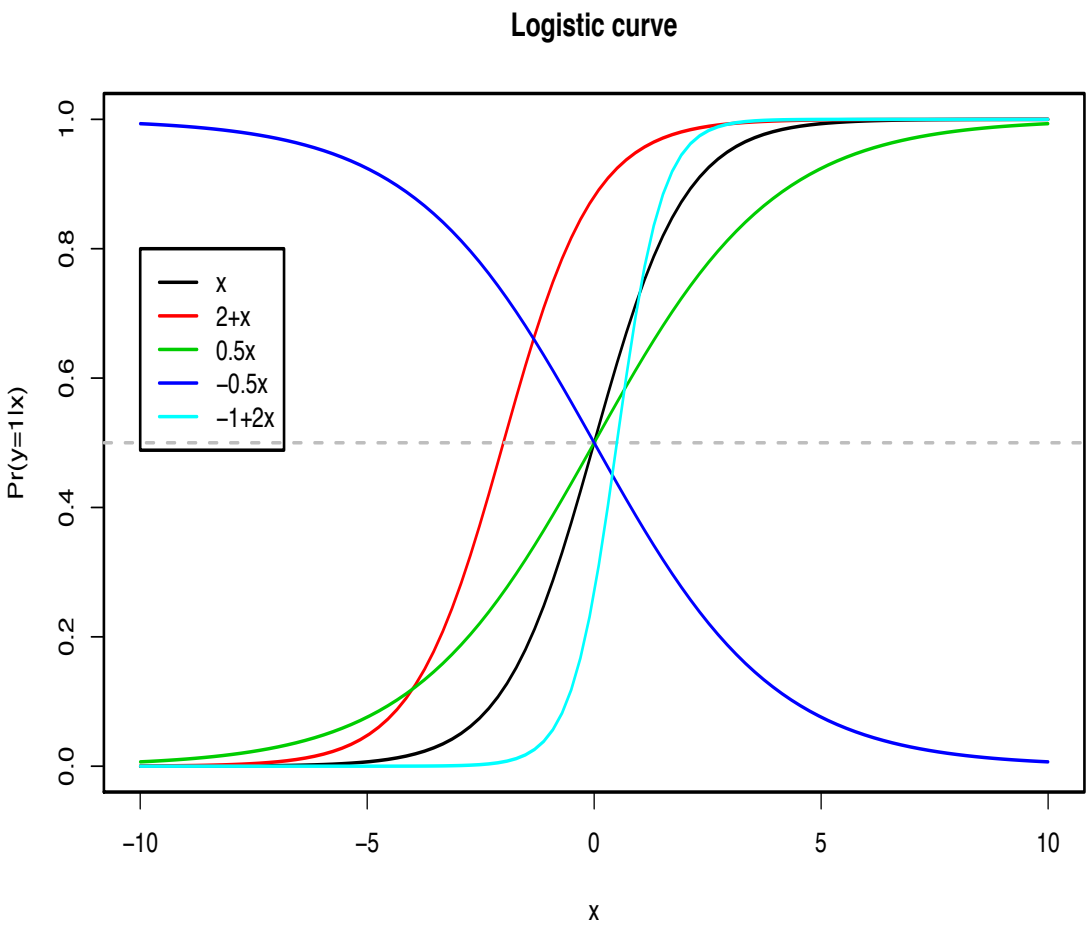
- For example, suppose I wanted h_θ to output the probability of spam/ham… $h_\theta(f) = P(spam \mid f)$
  - Not immediately useful since $h_\theta(f) = f^T\theta$ is not bounded [0,1]
  - Idea: "squash" the function using a sigmoid: $h_\theta(f) = \sigma(f^T\theta)$



$$h_\theta(f) = \frac{1}{1 + e^{-f^T\theta}}$$

  - This is called ***logistic regression***
  - $h_\theta(f) > 0.5 \Rightarrow$ predict 1
  - $h_\theta(f) < 0.5 \Rightarrow$ predict 0
  - *Threshold tunable*

How Coefficients Effect the Shape of the Logistic Curve



Logistic Regression Cost Function

Cost function or loss function

$$L(h_\theta(f), y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

$$y = 1 \Rightarrow L = -\log \hat{y} \quad \text{“want } \hat{y} \text{ large”}$$

$$y = 0 \Rightarrow L = -\log(1 - \hat{y}) \quad \text{“want } \hat{y} \text{ small”}$$

Logistic Regression Cost Function

Total training loss

$$\sum_{i=1}^{m} L(h_\theta(f^{(i)}), y^{(i)})$$

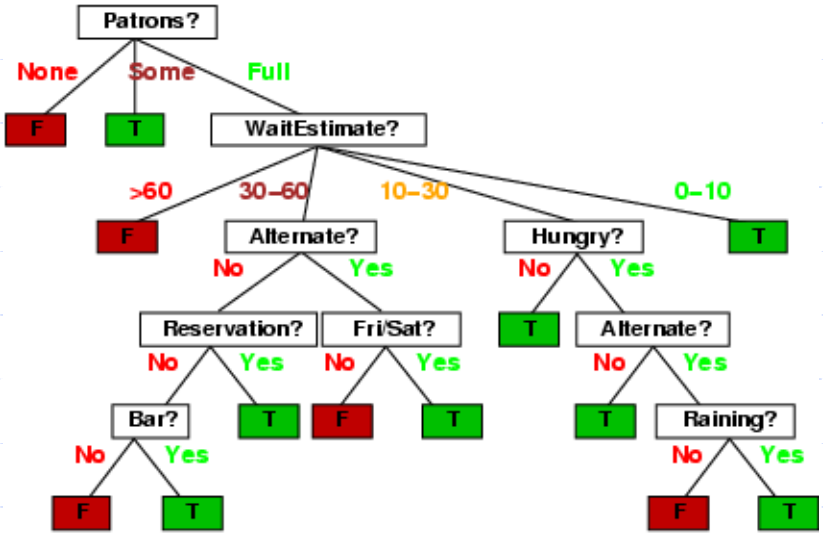
Gradient descent

$$\theta \leftarrow \theta - \alpha \sum_{i=1}^{m} \nabla_\theta L(h_\theta(f^{(i)}), y^{(i)})$$

Other interpretations of loss*

- We've given rather informal convexity-based justifications for both the squared loss in (non)linear regression and log loss in logistic regression
- It turns out there's also a clean probabilistic motivation for both
- Regression:
  - Assume independent, identically distributed (i.i.d) Gaussian noise
  - Maximize log-likelihood of the data given parameters —> least squares
- Logistic regression:
  - Assume i.i.d. Bernoulli distribution
  - Maximize log-likelihood of the data given parameters —> logistic loss

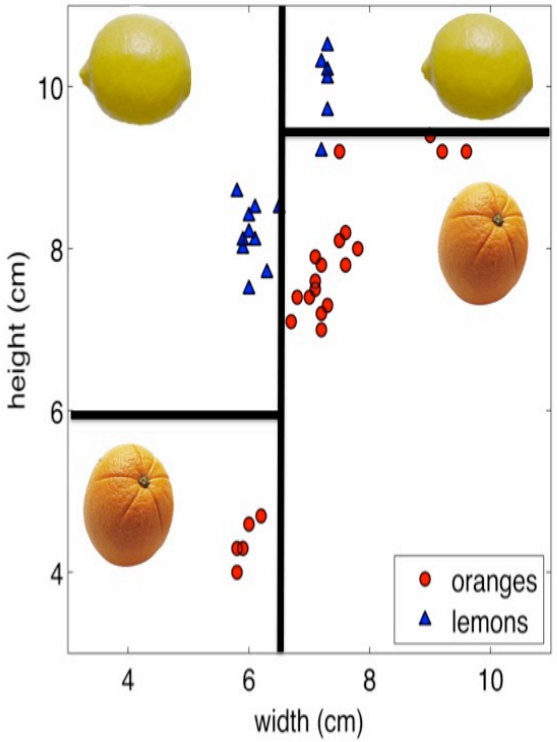
Learning Decision Trees

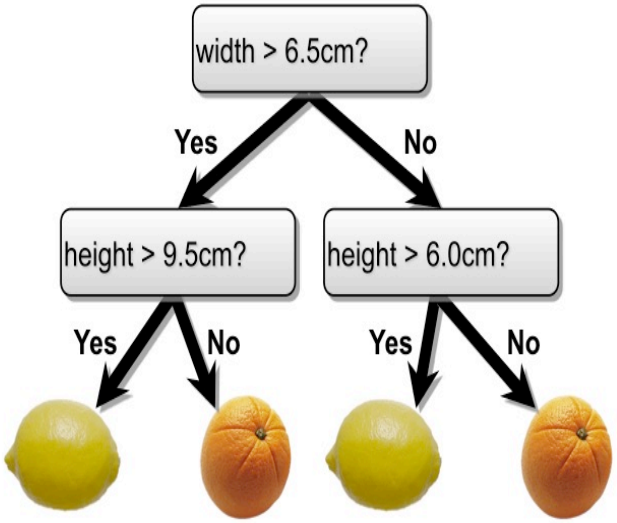


Learning Decision Trees

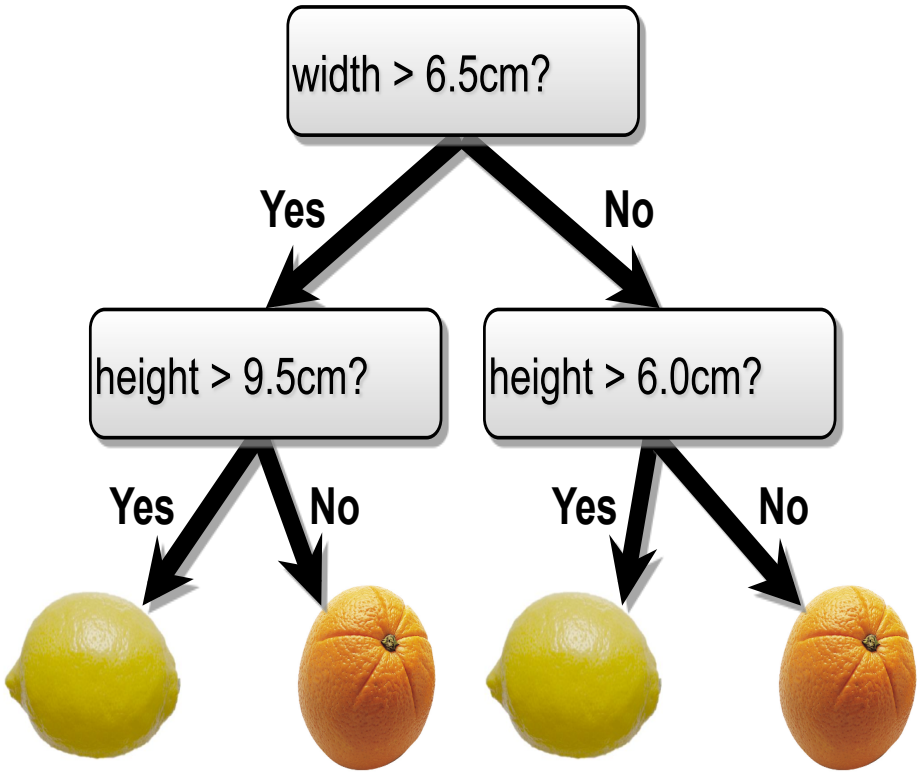
- One of the simplest, yet most successful learning algorithms
- Easy to implement
- Inputs – discrete or continuous
- Outputs – discrete (classification) or continuous (regression)
- Decision tree nodes represent sequence of tests
- Decision tree representation is very natural for humans – e.g., 'How To' manuals

Example

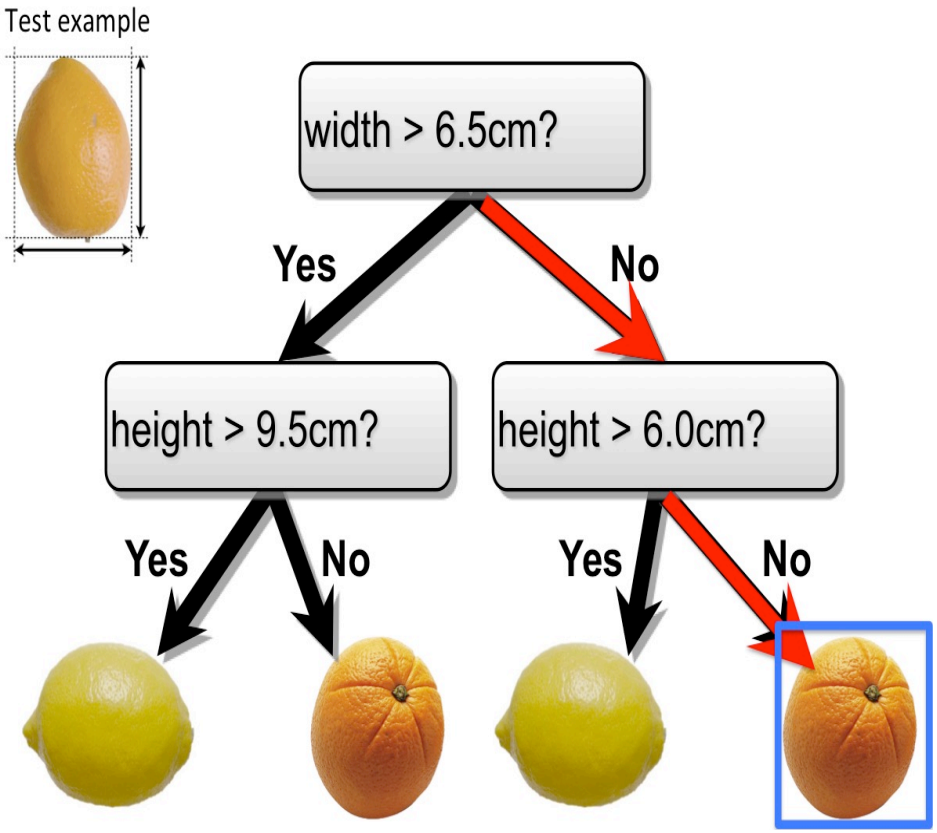

10
8
6
4
height (cm)
4
6
8
10
width (cm)
oranges
lemons


width > 6.5cm?
Yes
No
height > 9.5cm?
height > 6.0cm?
Yes
No
Yes
No

Example



Example


Test example
width > 6.5cm?
Yes
No
height > 9.5cm?
height > 6.0cm?
Yes
No
Yes
No

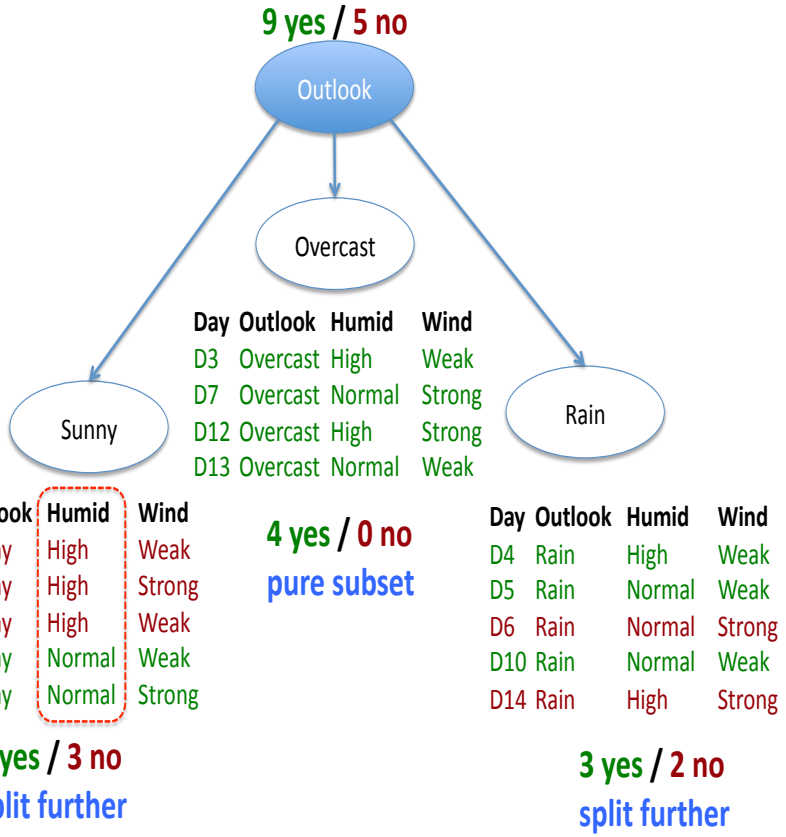
Predict If John Will Play Tennis (1)

Training examples: **9 yes / 5 no**

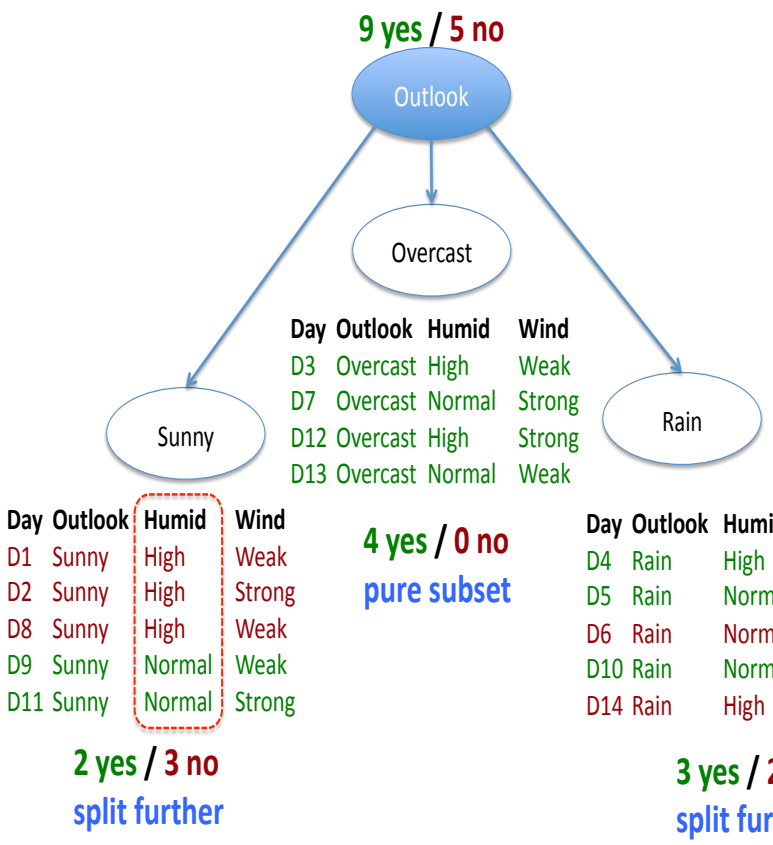
Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

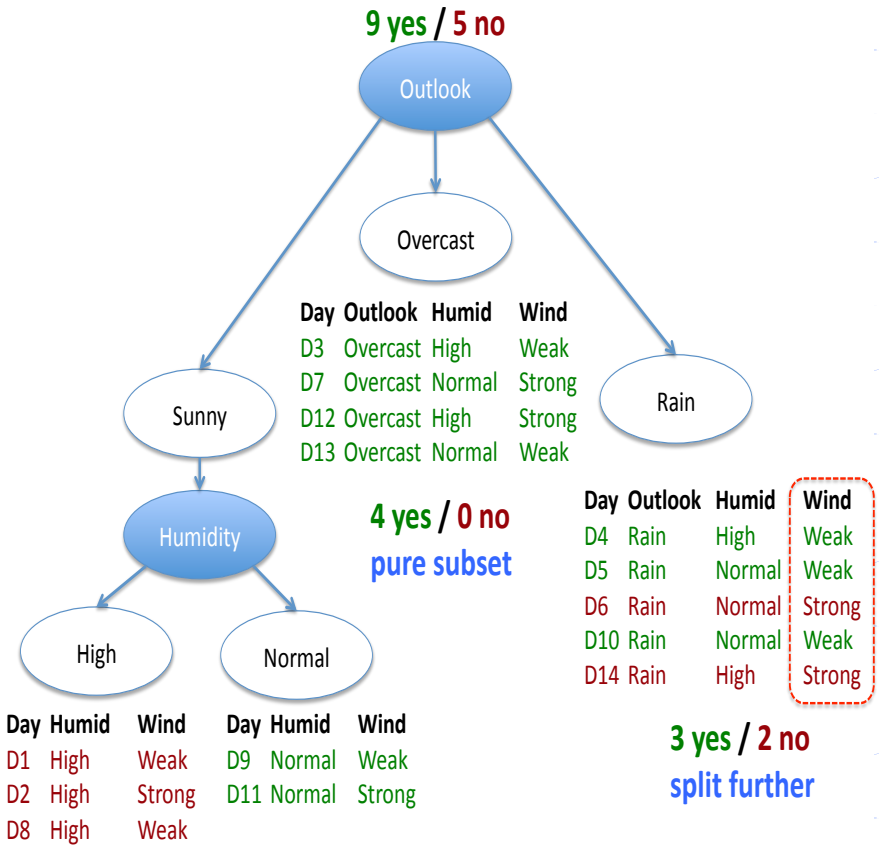
New data:

D15	Rain	High	Weak	?

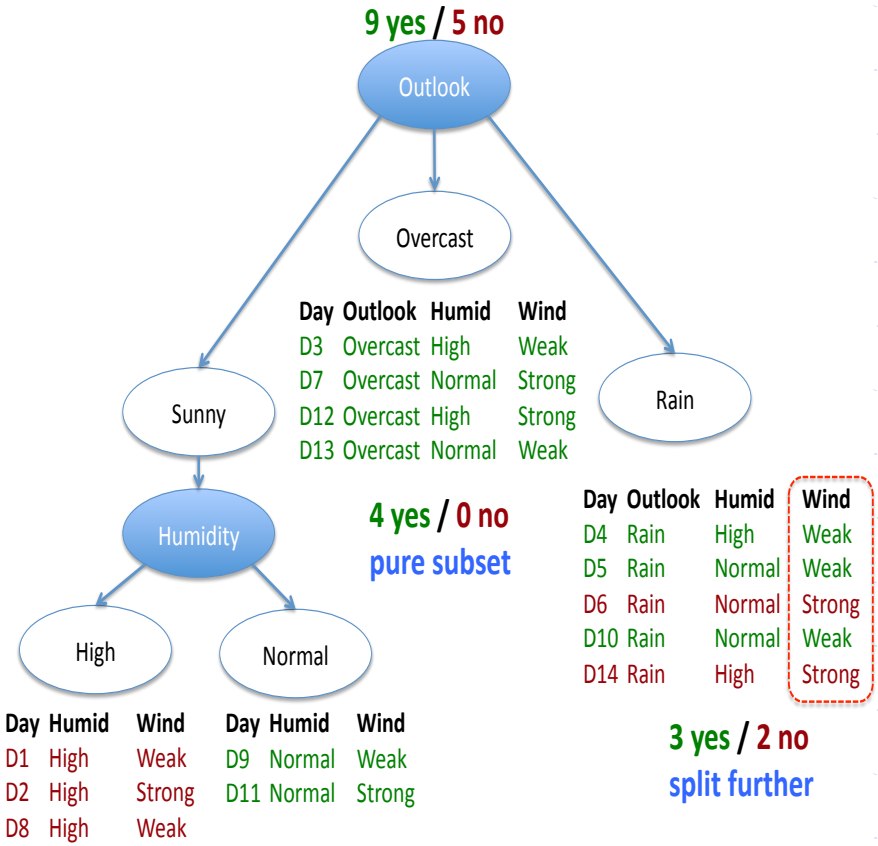


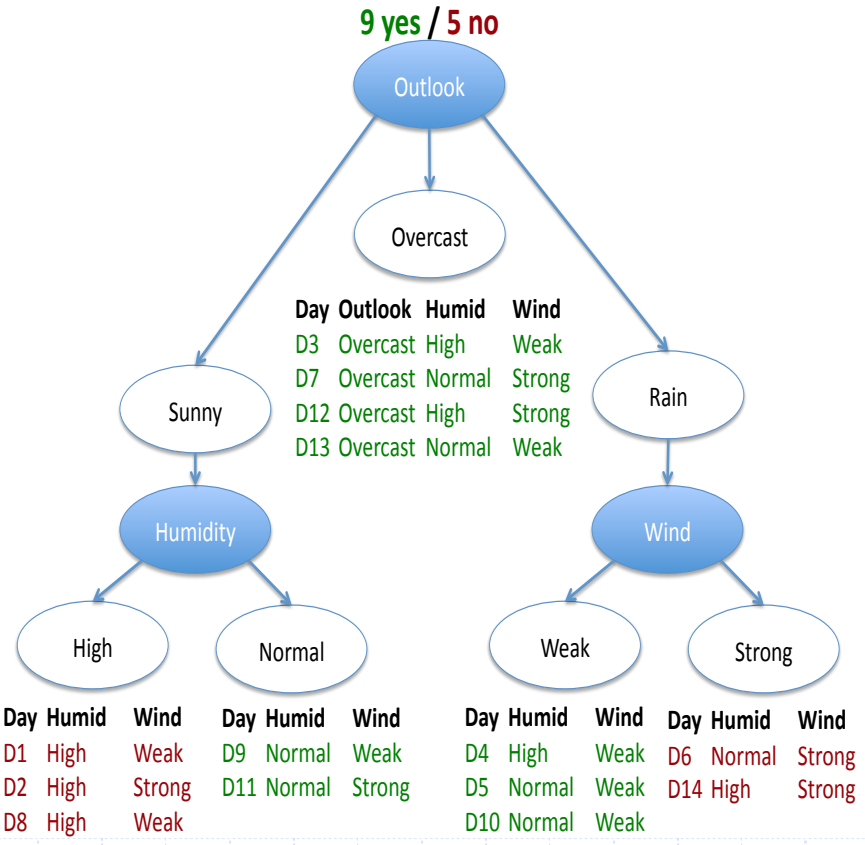
Predict If John Will Play Tennis (2)



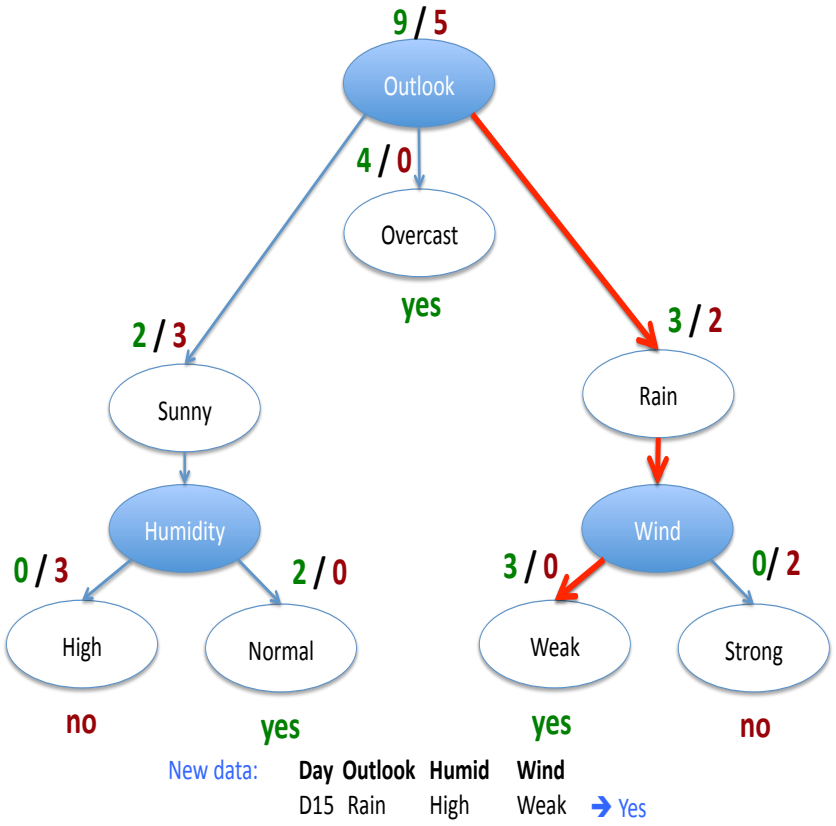


Predict If John Will Play Tennis (3)

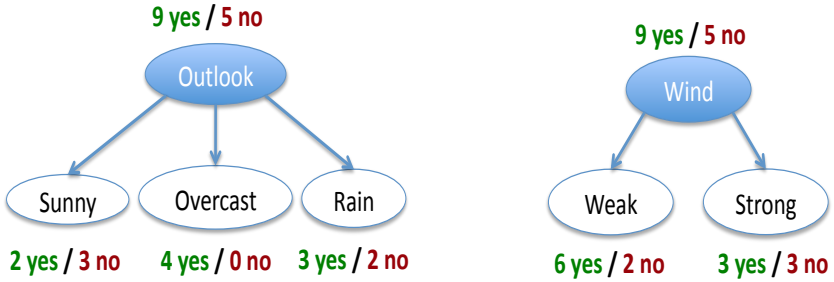




Predict If John Will Play Tennis (4)



Which Attributes to Split on



- We want to measure purity of split
  - more certain about Yes/No after the split
    - pure set (4 yes / 0 no) => completely certain (100%)
    - impure (3 yes / 3 no) => completely uncertain (50%)
    - 4 yes / 0 no as pure as 0 yes / 4 no

Learning Decision Trees

- Learning the simplest (smallest) decision tree is an NP complete problem
- Greedy heurist:
  - Start from an empty decision tree
  - Split on next best attribute
  - Recursively build internal nodes

When to Use Decision Tree

- Decision tree is good when:
  - Instances described by attribute-value pairs
  - Target function is discrete
  - Interpretability of learned hypothesis is desired (rule sets)
  - Training data might be noisy

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

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)

Attribute-based representations

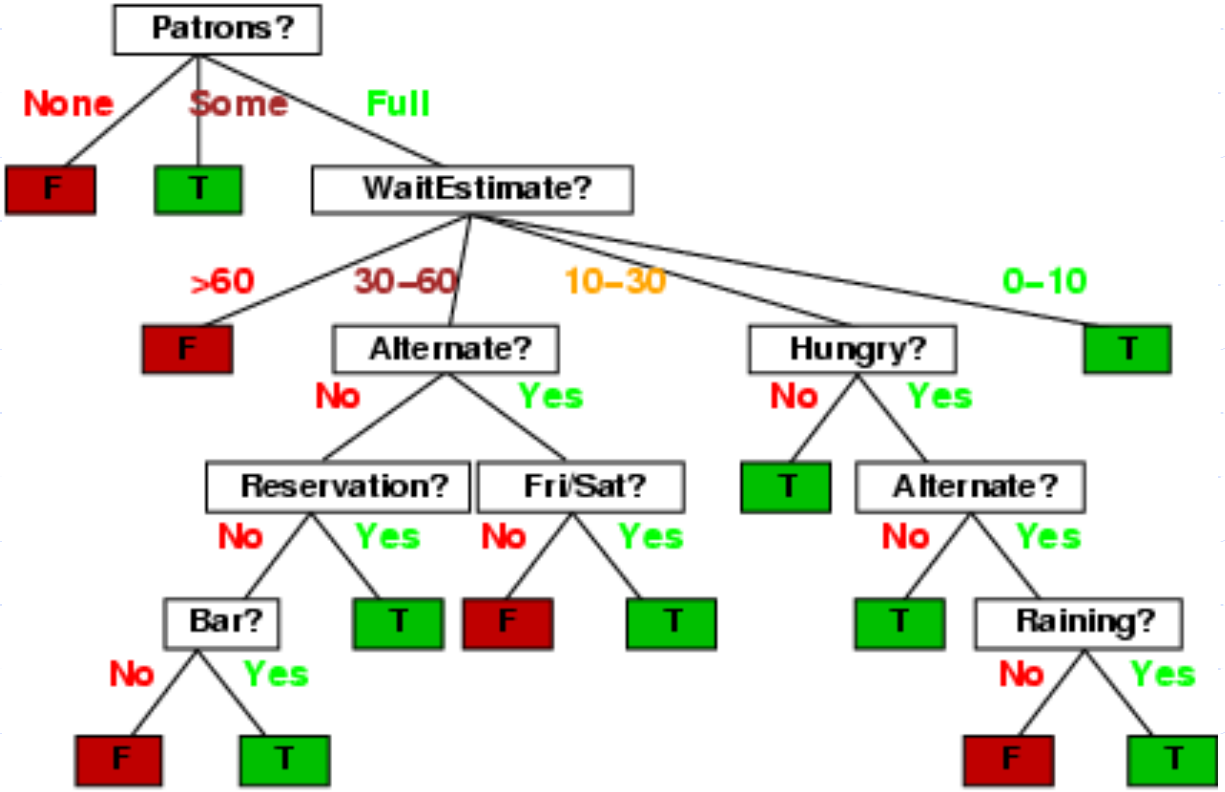
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

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

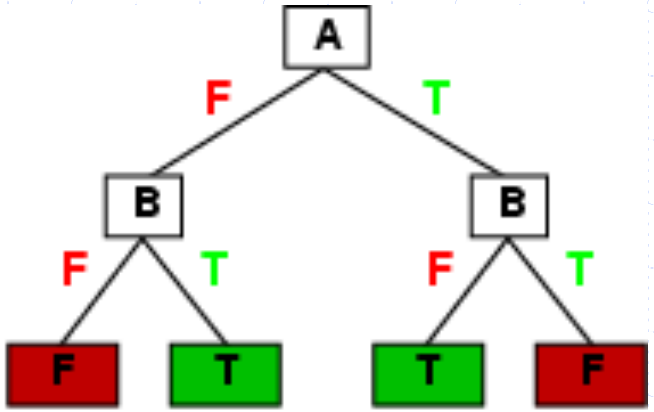
- here is the "true" tree for deciding whether to wait:



Expressiveness

- 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 (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples

- Prefer to find more compact decision trees

Hypothesis spaces

How many distinct functions with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- For 10 attributes, 2^{1024} or about 10^{308} different functions; for 20 attributes, $10^{300,000}$

- How many trees? More than this, since more than one tree can represent the same function

What Makes a Good Tree?

- Small
  - Ockham's razor
    - Simpler is better
  - Avoids over-fitting
- A decision tree may be human readable, but not use human logic
  - The decision tree you would write for a problem may differ from what is generated by a decision tree learning algorithm

Small Trees

- How do we build small trees that accurately capture data?
- Optimal decision tree learning is NP-complete

- Constructing Optimal Binary Decision Trees is NP-complete. Laurent Hyafil, RL Rivest. Information Processing Letters, Vol. 5, No. 1. (1976), pp. 15-17.

Greedy Algorithms

- Like many NP-complete problems we can get pretty good solutions
- Most decision tree learning is by greedy algorithms
  - Adjustments are usually to fix greedy selection problems
- Top down decision tree learning
  - Recursive algorithms

Try #1

- function BuildDecisionTree(data, labels):
  - if all labels are the same
    - return leaf node for that label
  - else
    - let f be the best feature for splitting
    - left = BuildDecisionTree(data with f=0, labels with f=0)
    - right= BuildDecisionTree(data with f=1, labels with f=1)
    - return Tree(f, left, right)

Does this always terminate?

Base Cases

- All data have same label
  - Return that label
- No examples
  - Return majority label of all data
- No further splits possible
  - Return majority label of passed data

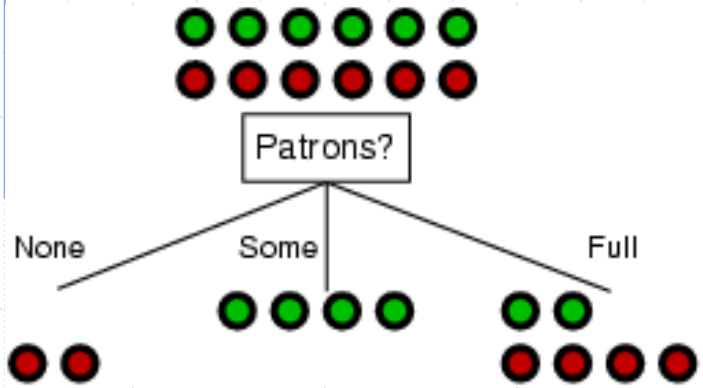
Decision tree learning

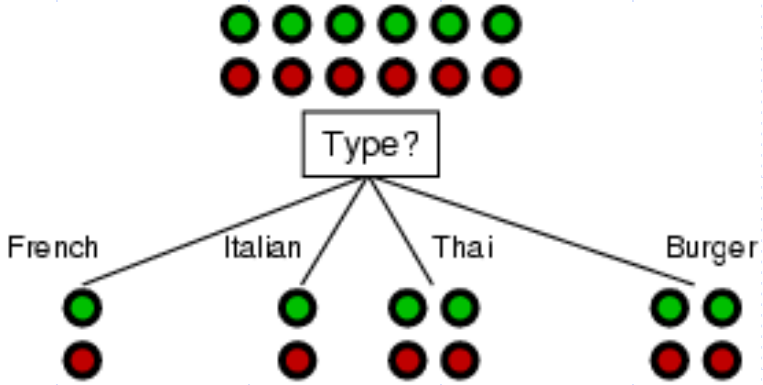
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
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

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





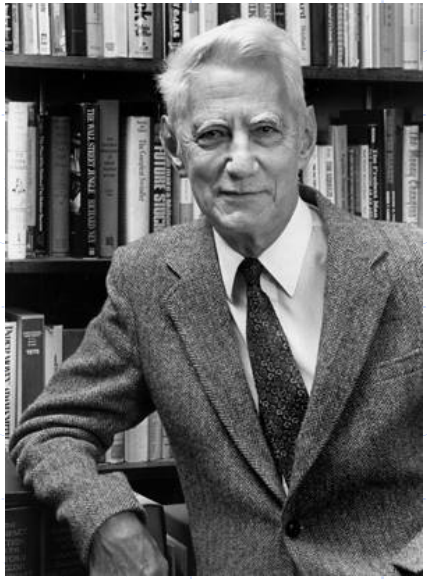
- *Patrons?* is a better choice

Selecting Attributes

- The best feature for splitting
  - The most *informative attribute*
  - Select the attribute that is most informative about the labels
- Information theory

Information Theory

- The quantification of information
- Founded by Claude Shannon
  - Landmark paper in 1948
  - Noisy channel theorem



Information Theory

- A brief introduction…

Information Theory

- In information theory, **entropy** is a measure of the uncertainty associated with a random variable. It quantifies, using expected value, the information contained in a message in bits. Equivalently, it is a measure of the average information content one is missing when one does not know the value of the random variable (Wikipedia)
- Definition:

$$H(Y) = -\sum_{y} p(Y = y) \log_2 p(Y = y)$$

- One interpretation of entropy is the expected number of bits needed to encode Y or questions needed to guess Y.

$$H(Y) = -\sum_{y} p(Y = y) \log_2 p(Y = y)$$

Example

- Fair coin: p(heads) = 0.5
  - - (.5log.5 + .5log.5)= 1
- Biased coin: p(heads) = 0.99
  - - (.99log.99 + .01log.01) = 0.08

Conditional Entropy & IG

- To quantify predictiveness of a attribute X for Y, we consider the *conditional entropy*, or the expected number of bits needed to encode Y or questions needed to guess Y, **knowing** X.

$$H(Y \mid X) = \sum_x p(X = x) H(Y \mid X = x)$$

- So to measure the reduction in entropy of Y from knowing X, we use *information gain (IG)*:

$$IG(Y \mid X) = H(Y) - H(Y \mid X)$$

Selecting Features

- The best feature for splitting
  - The most *informative feature*
  - The feature with the highest *information gain*

Using information theory

- To implement `Choose-Attribute` in the DTL algorithm
- For a training set containing *p* positive examples and *n* negative examples:

$$B(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Introduce B(p,n), which is just H(X), X~(p/p+n,n/p+n)

Information gain

- A chosen attribute *A* divides the training set *E* into subsets $E_1, \ldots, E_v$ according to their values for *A*, where *A* has v distinct values.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} B(p_i, n_i)$$

- Information Gain (IG) or reduction in entropy from the attribute test:

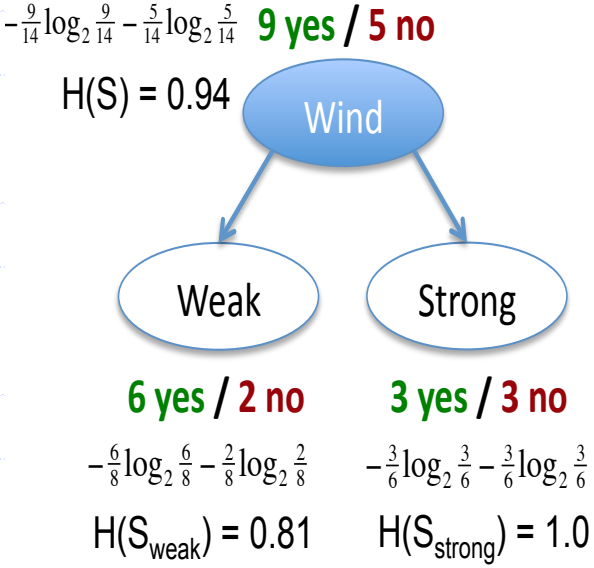
$$IG(A) = B(p, n) - remainder(A)$$

- Choose the attribute with the largest IG

Information Gain

- Expected drop in entropy after split:

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



$$Gain(S, Wind) = H(S) - 8/14 \cdot H(S_{weak}) - 6/14 \cdot H(S_{strong})$$

Information gain

For the training set, $p = n = 6$, $B(6/12) = 1$ bit

Consider the attributes *Patrons* and *Type* (and others too):

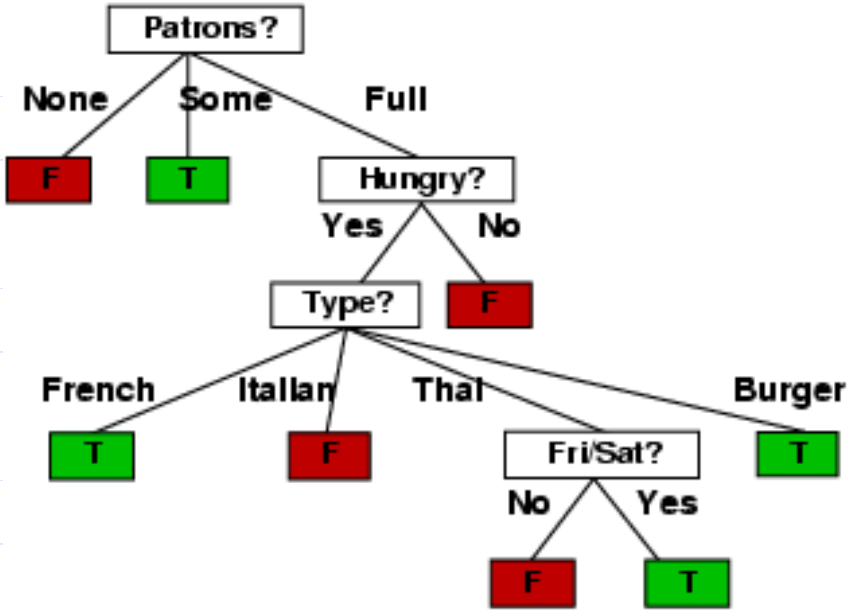
$$IG(Patrons) = 1 - [\frac{2}{12} B(0,2) + \frac{4}{12} B(4,0) + \frac{6}{12} B(2,4)] = .0541 \text{ bits}$$

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

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Example contd.

- Decision tree learned from the 12 examples:

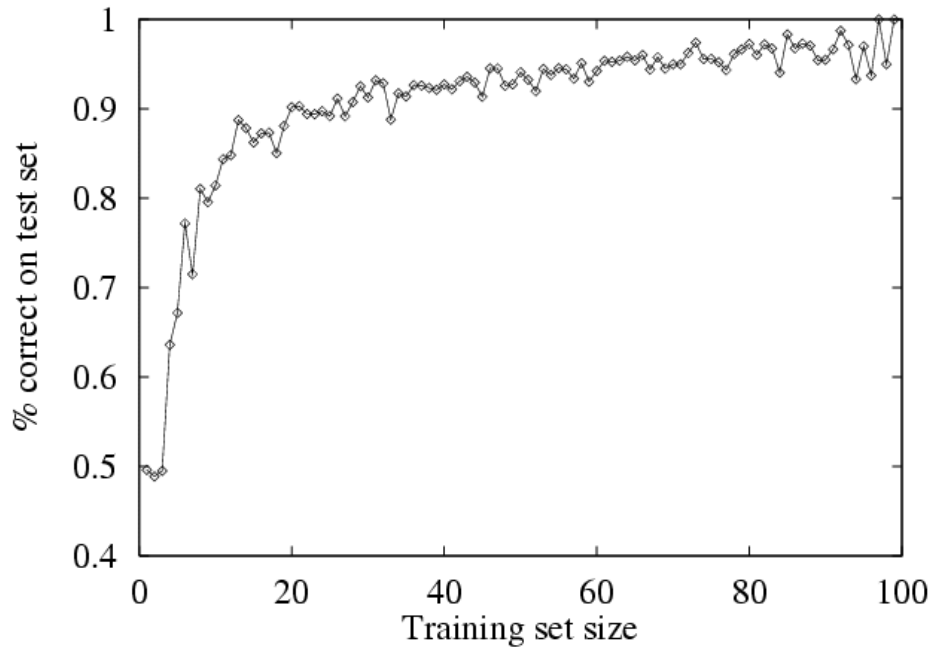


- Substantially simpler than "true" tree---a more complex hypothesis isn't justified by small amount of data

Performance measurement

- How do we know that $h \approx f$?
  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



Tradeoff

- Complete trees
  - can over-fit badly
  - Lots of variance
- 0 depth trees (return most likely label) have no variance
  - Totally biased towards majority label
- A good tree balances between these two
  - How do we learn balanced trees?

Pruning: New Base Cases

- Stop when too few examples
- Stop when max depth reached
- Stop when my classification error is not much more than average of my children
- X^2 pruning- stop when remainder is no more likely than chance

Extensions of the decision tree learning algorithm

- Multi-value attributes: Use gain ratios
- Real-valued data: Use split points
- Noisy data and overfitting: Use pruning
- Missing attributes
- C4.5 is popular decision tree system that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

DT Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set