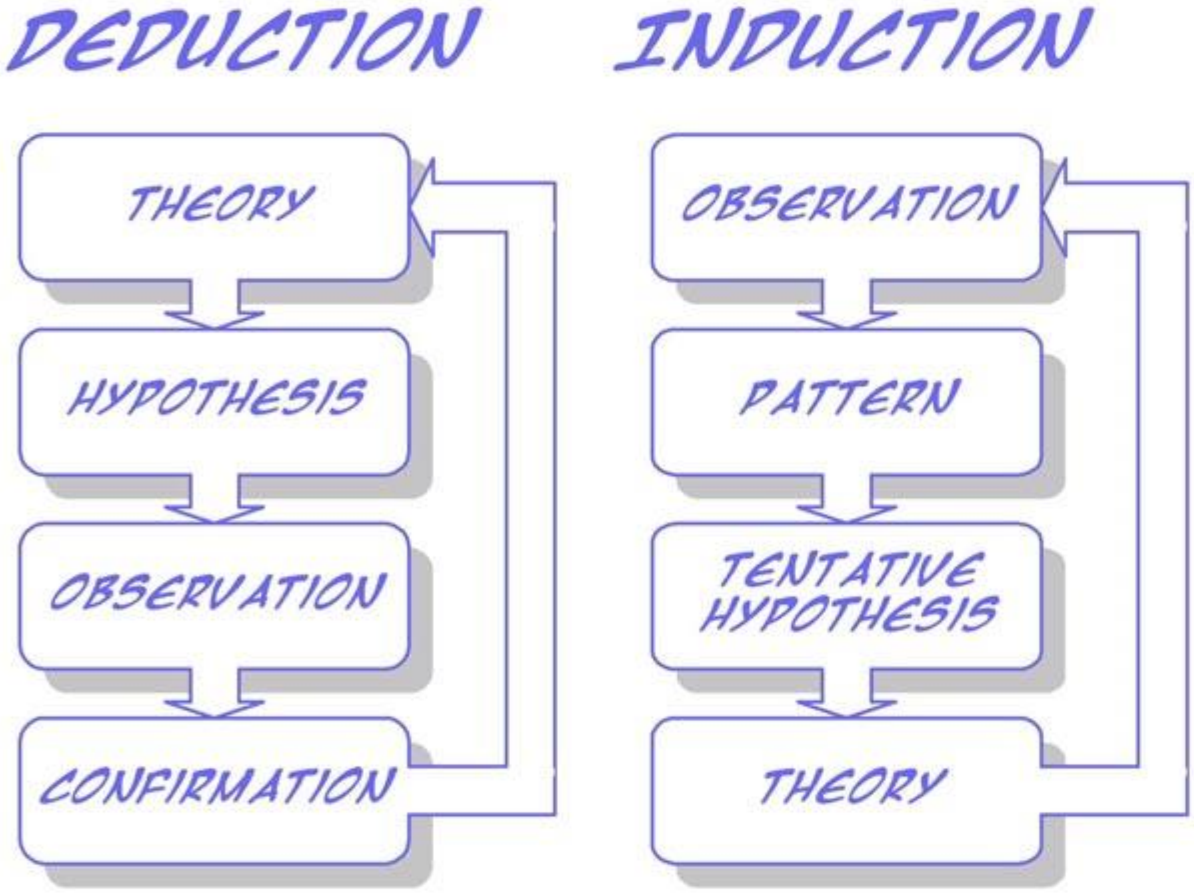
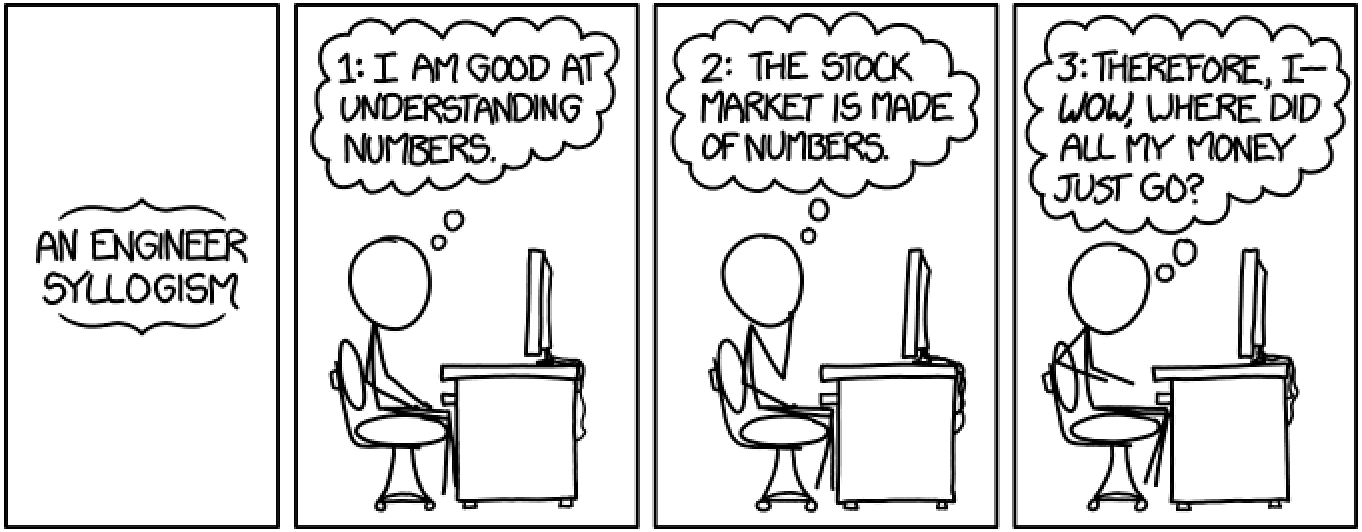
Learning from Observations



Outline

- Learning agents
- Inductive learning
- Decision tree learning
- Measuring learning performance

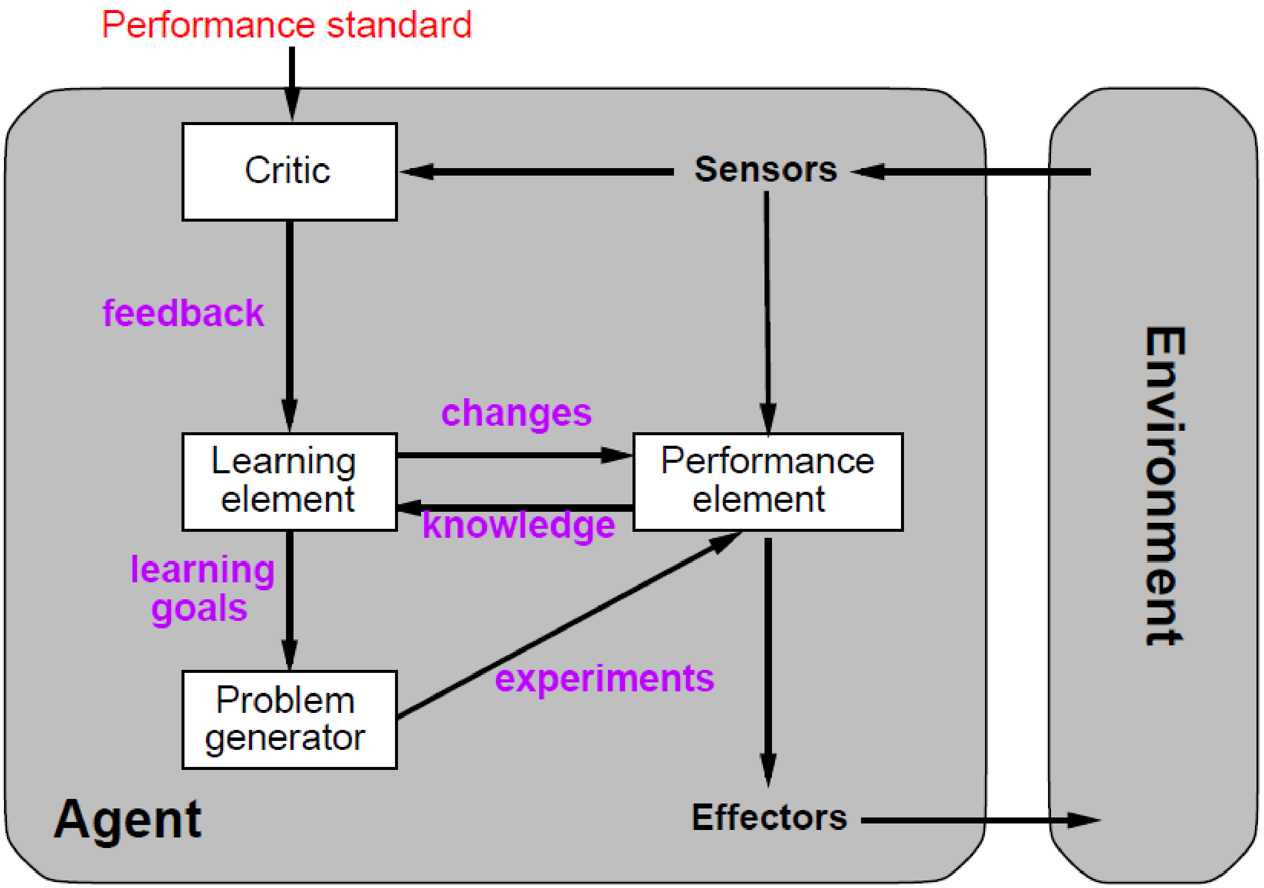


http://xkcd.com/1570/

Learning

- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Learning Agents



Learning Element

- Design of learning element is dictated by
  - What type of performance element is used
  - Which functional component is to be learned
  - How that functional component is represented
  - What kind of feedback is available
- Supervised learning: correct answers for each instance
- Reinforcement learning: occasional rewards
- Unsupervised learning: agent learns patterns without explicit feedback
- Example scenarios:

Performance element	Component	Representation	Feedback
Alpha−beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor−state axioms	Outcome
Utility−based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept−action fn	Neural net	Correct action

Inductive Learning (a.k.a. Science)

- Simplest form: learn a function from examples (tabula rasa)
  - f is the target function
- An example is a pair x, f(x), e.g.,

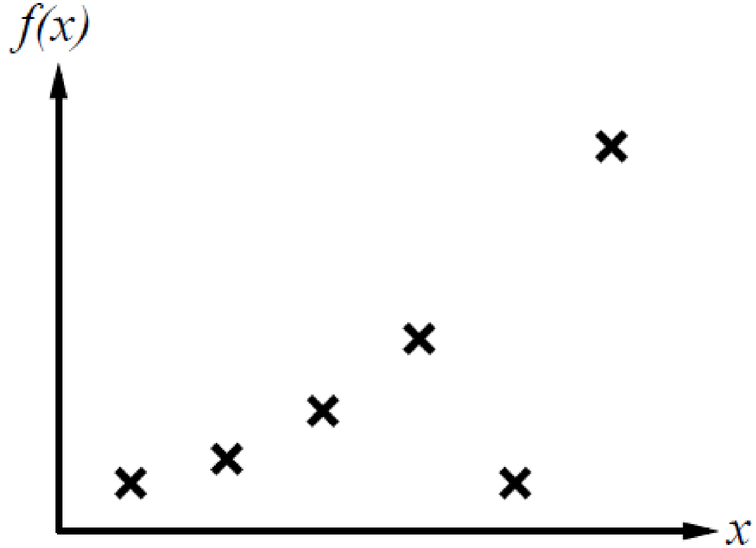
O	O	X
	X	
X		

$, \quad +1$

- Problem: find a(n) hypothesis h
  - Such that h ≈ f, given a training set of examples
- This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes a deterministic, observable “environment"
  - Assumes examples are given
  - Assumes that the agent wants to learn f - why?

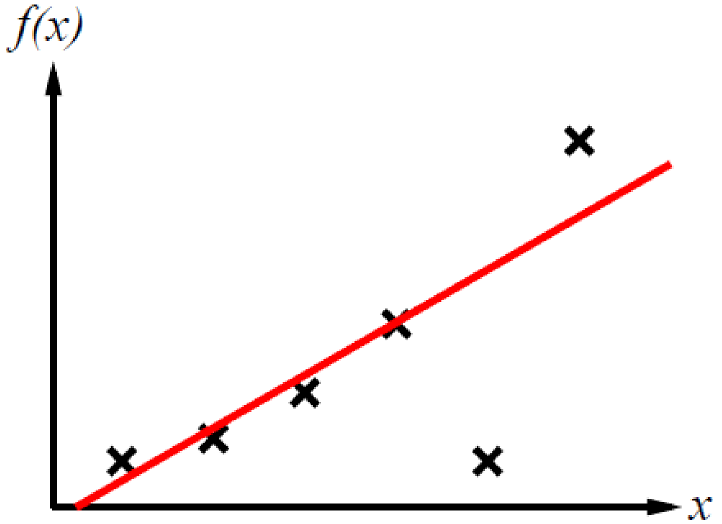
Inductive Learning Method

- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



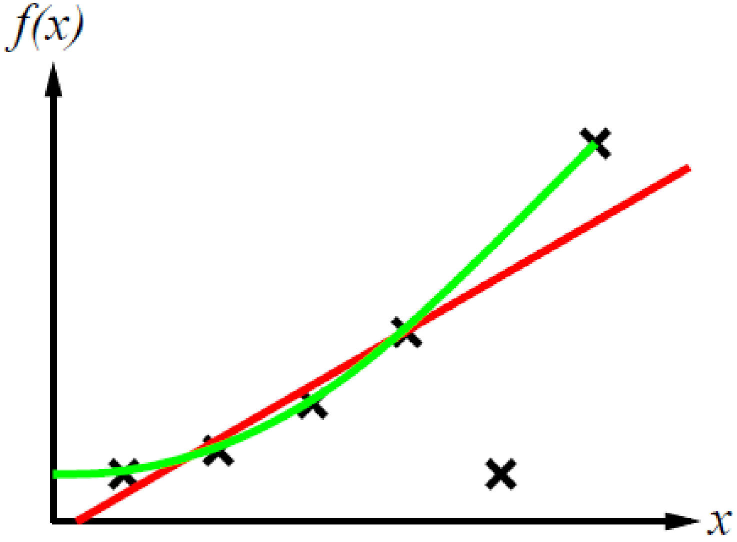
Inductive Learning Method

- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



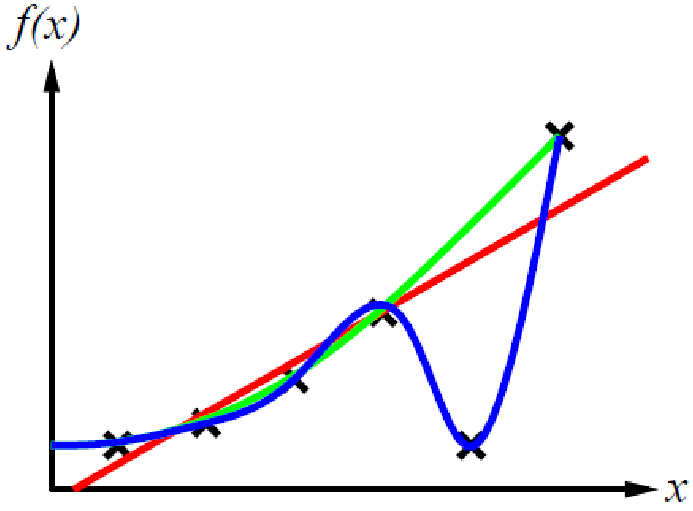
Inductive Learning Method

- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



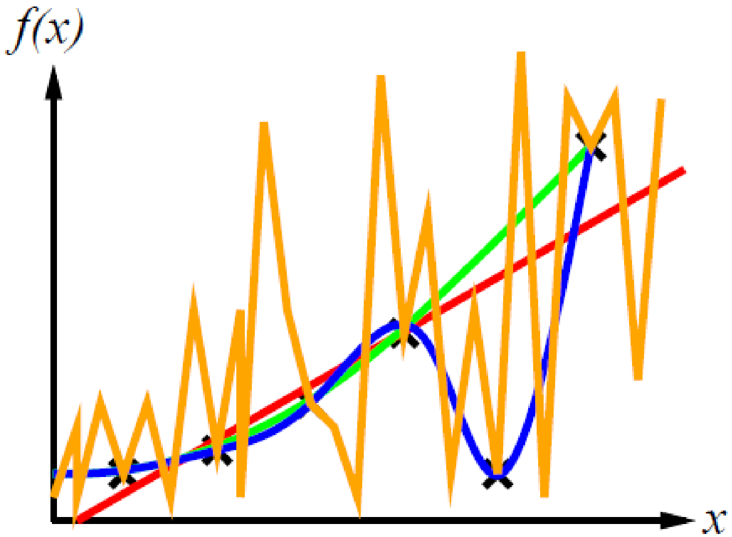
Inductive Learning Method

- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



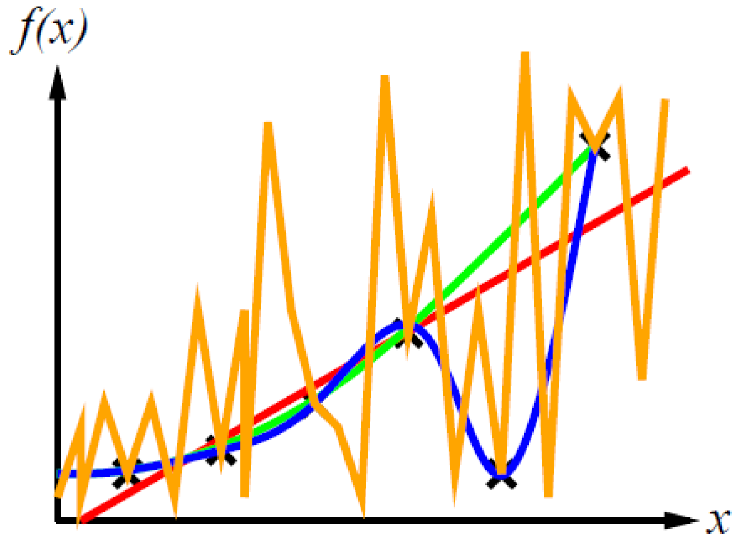
Inductive Learning Method

- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:



Inductive Learning Method

- Construct/adjust h to agree with f on training set (h is consistent if it agrees with f on all examples)
- E.g., curve fitting:
- Ockham's razor: maximize a combination of consistency and simplicity



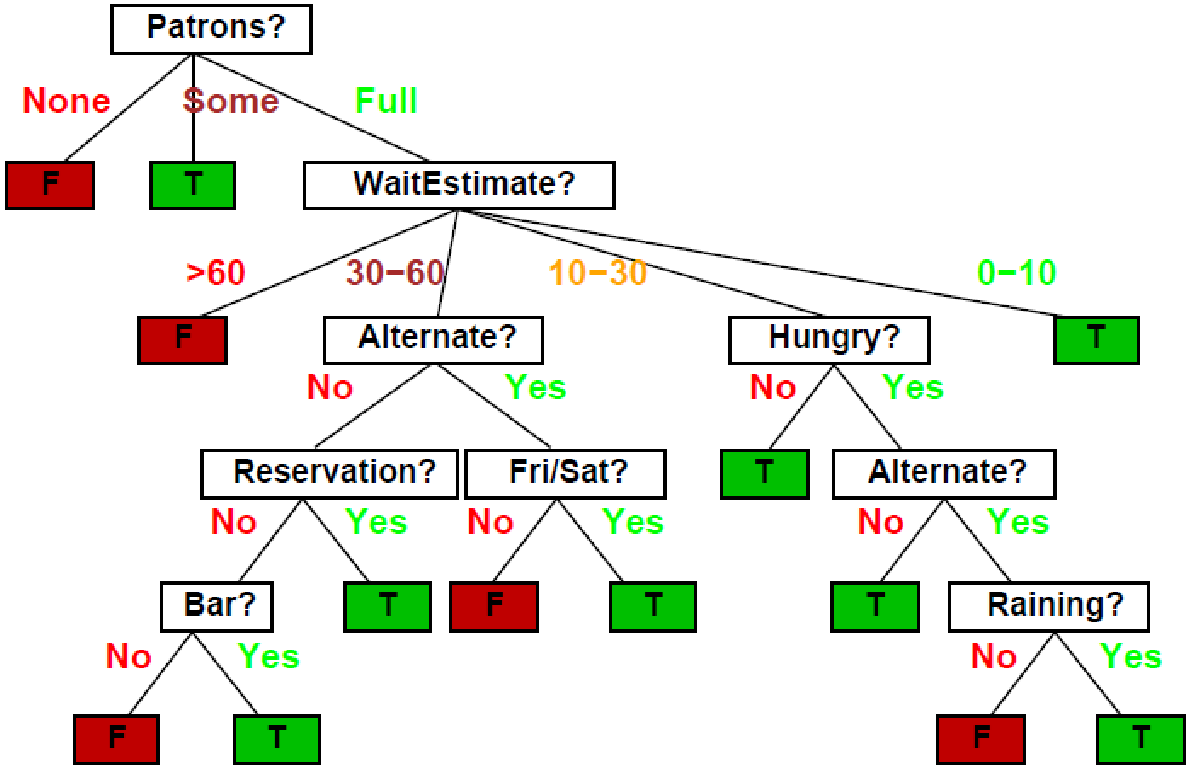
Attribute-Based Representations

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

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

Decision Trees

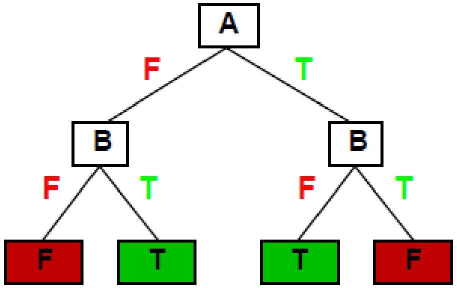
- One possible representation for hypotheses
- E.g., here is a tree for deciding whether to wait:



Expressiveness

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

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



Hypothesis Spaces

- How many distinct decision trees with n Boolean attributes??

Hypothesis Spaces

- How many distinct decision trees with n Boolean attributes??
  - = number of Boolean functions

Hypothesis Spaces

- How many distinct decision trees with n Boolean attributes??
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows

Hypothesis Spaces

- How many distinct decision trees with n Boolean attributes??
- = number of Boolean functions
- = number of distinct truth tables with 2^n rows = 2^{2^n}

Hypothesis Spaces

- How many distinct decision trees with n Boolean attributes??
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows = 2^{2^n}
  - E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

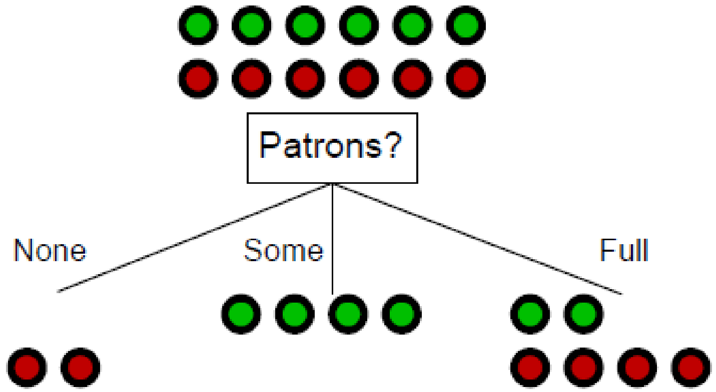
Decision Tree Learning

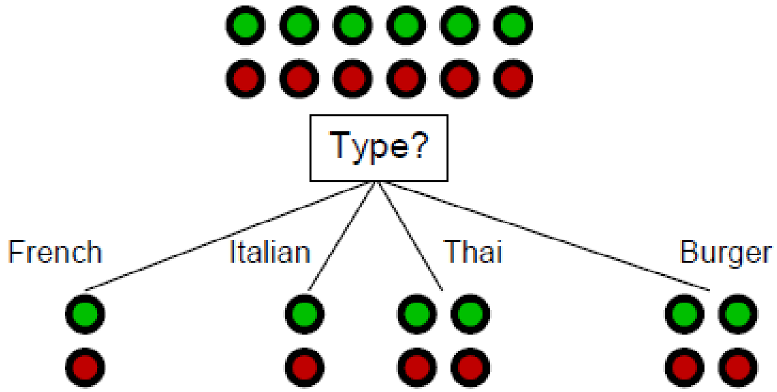
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant" attribute as root of (sub)tree

```
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 - gives information about the classification





Information Theory

- Information answers questions
- The more clueless I am about the answer initially, the more information is contained in the answer
- Scale: 1 bit = answer to Boolean question with prior <0.5, 0.5>
- 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)

Information

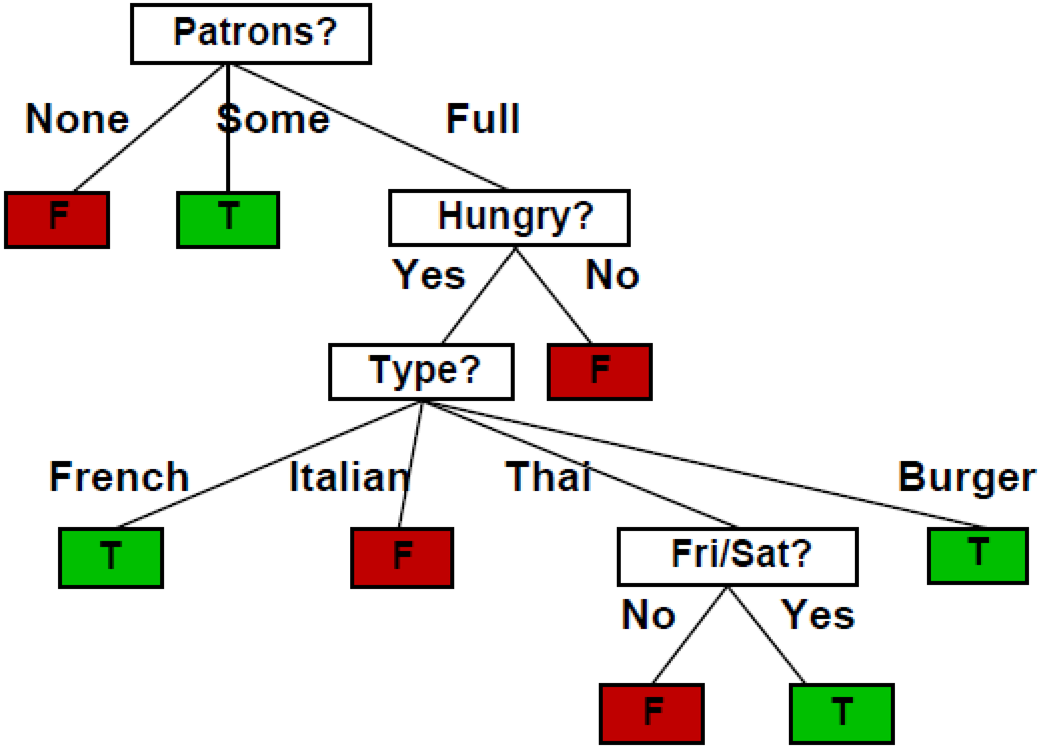
- Suppose we have p positive and n negative examples at the root
  - ⇒ $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
- E.g., for 12 restaurant examples, p=n=6 so we need 1 bit
- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification
- Let E_i have p_i positive and n_i negative examples
  - ⇒ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
  - ⇒ expected number of bits per example over all branches is

$$\Sigma_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$$

- For Patrons?, this is 0.459 bits, for Type this is (still) 1 bit
  - ⇒ choose the attribute that minimizes the remaining information needed

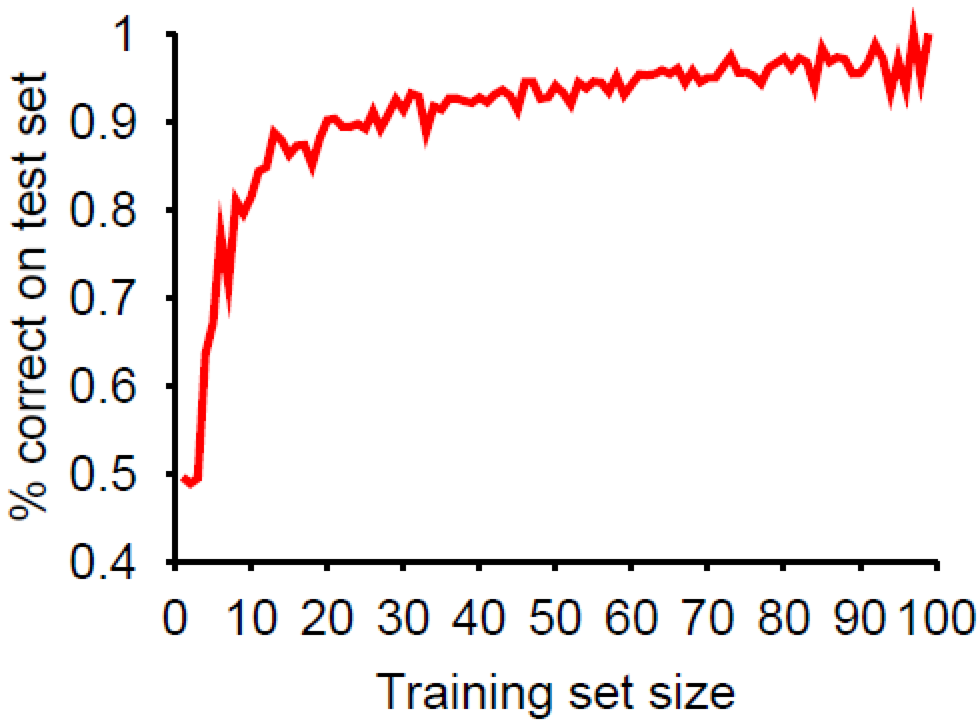
Example

- Decision tree learned from the 12 examples:
- Substantially simpler than “true" tree - a more complex hypothesis isn't justified by small amount of data



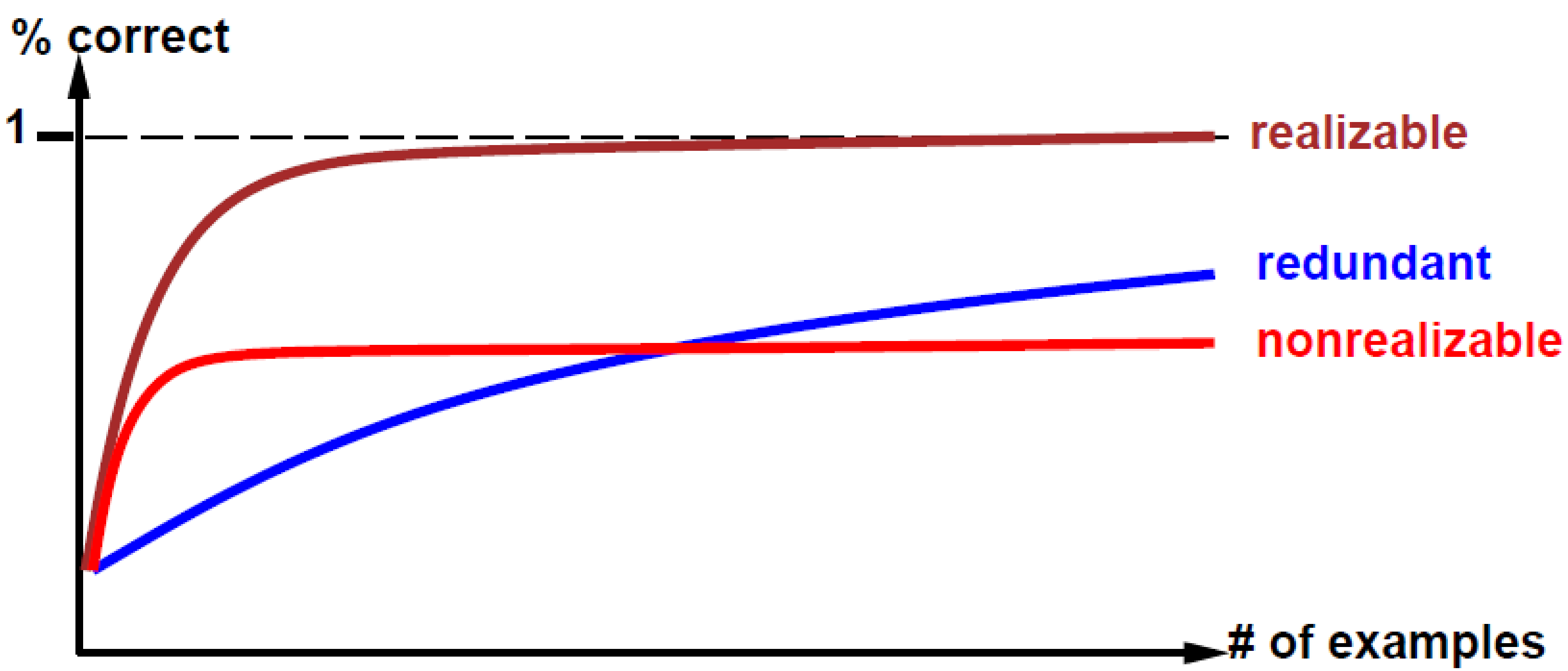
Performance Measurement

- How do we know that h ≈ 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



Performance Measurement

- 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)



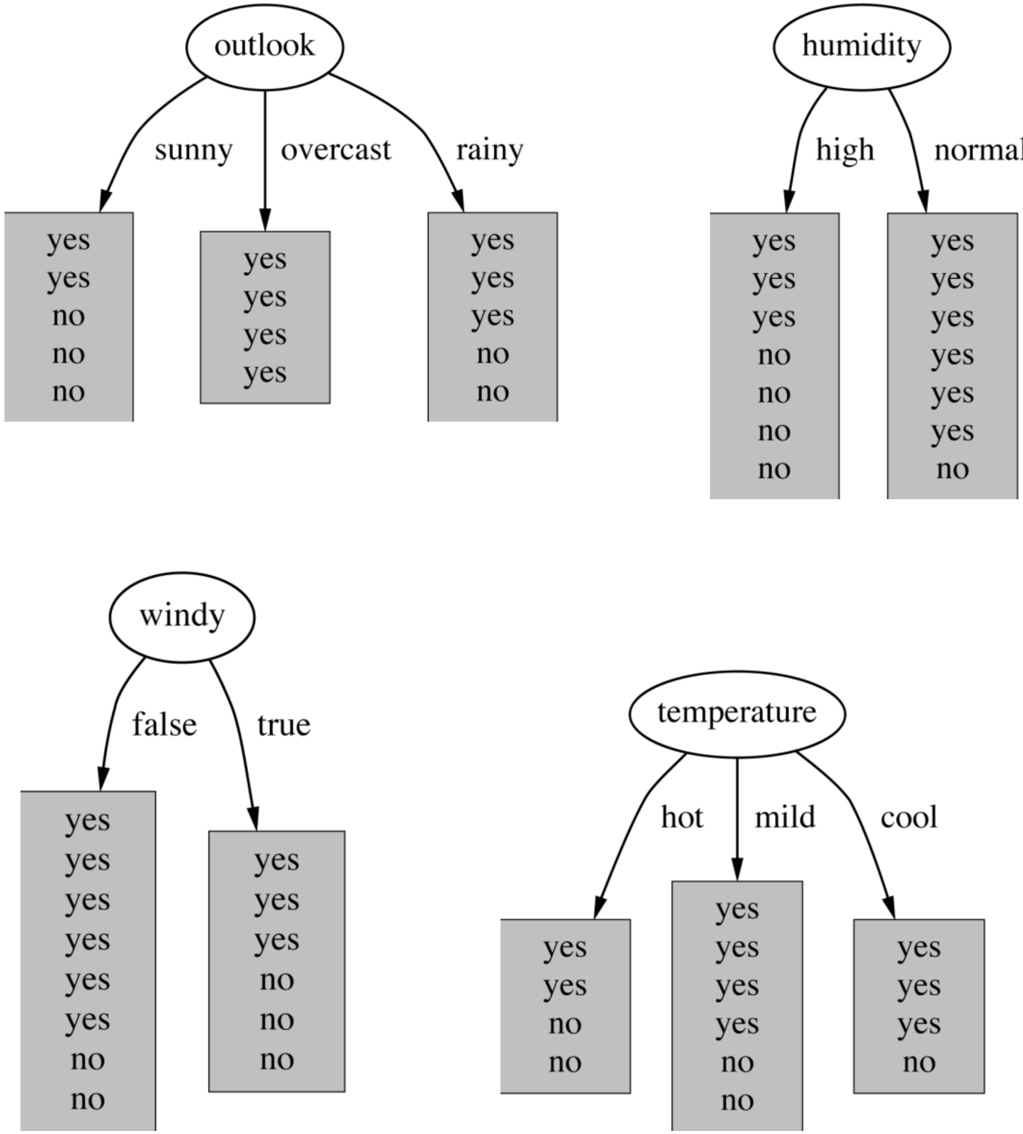
Summary

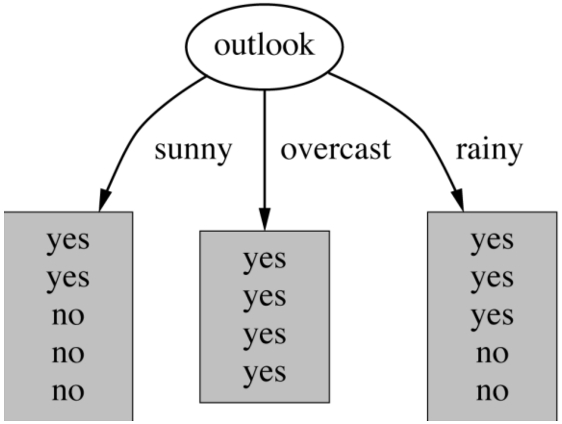
- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation
- For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set

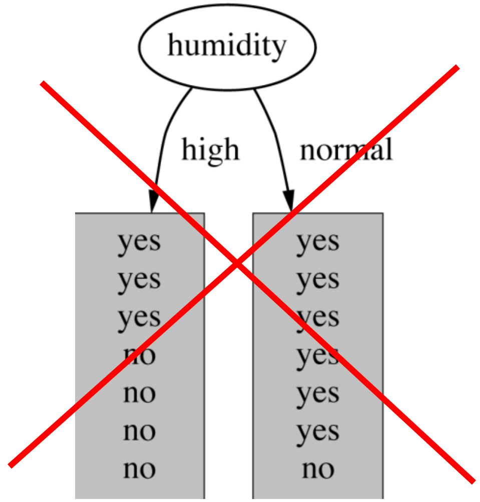
Constructing Decision Trees

- Strategy: Top Down Recursive *divide-and-conquer* fashion
  - First: Select attribute for root node
    Create branch for each possible attribute value
  - Then: Split instances into subsets
    One for each branch extending from the node
  - Finally: Repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class or there are no more attributes to split on

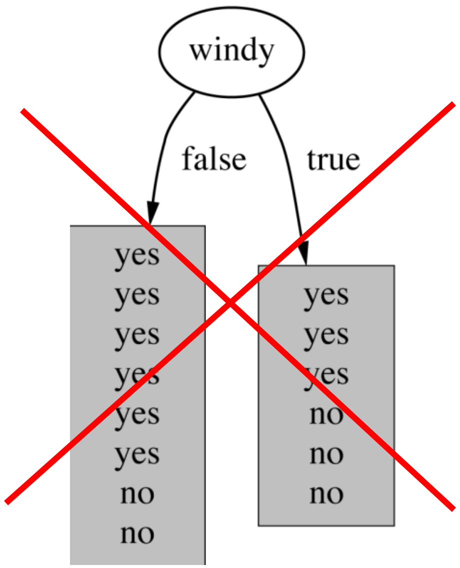
Which Attribute to Select?

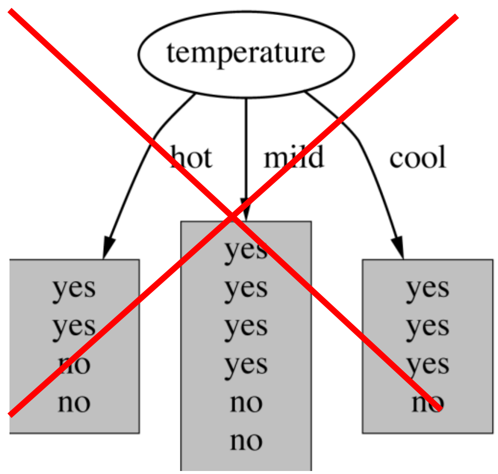






Which Attribute to Select?





Criterion for Attribute Selection

- Which is the best attribute?
  - Want to get the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes
- Popular impurity criterion: *information gain*
  - Information gain increases with the average purity of the subsets
- Strategy: Choose attribute that gives greatest information gain

Computing Information

- Measure information in *bits*
  - Given a probability distribution, the info required to predict an event is the distribution's *entropy*
  - Entropy gives the information required in bits (can involve fractions of bits)
    - Because were dealing with bits, the log is calculated in base 2
- Formula for computing the entropy:

$$\text{entropy}(p_{1,}p_{2,}\ldots,p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots -p_n \log p_n$$

An Algebraic Aside…

- Logarithms:
  - $b^y = x$
  - $y = \log_b x$
  - e.g. $2^4 = 16$, $4 = \log_2 16$
- To change to a different base:
  - $\log_b x = \log_{10} x / \log_{10} b$
  - e.g.
    - $\log_2 2 = \log_{10} 2 / \log_{10} 2 = 0.301 / 0.301 = 1$
    - $\log_2 4 = \log_{10} 4 / \log_{10} 2 = 0.602 / 0.301 = 2$
    - $\log_2 8 = \log_{10} 8 / \log_{10} 2 = 0.9031 / 0.301 = 3$

Example: Attribute *Outlook*

- *Outlook* = *Sunny* :

$$\text{info}([2,3]) = entropy(2/5,3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971\,bits$$

- *Outlook* = *Overcast* :

$$\text{info}([4,0]) = entropy(1,0) = -1\log(1) - 0\log(0) = 0\,bits$$

Note: this is normally undefined.

- *Outlook* = *Rainy* :

$$\text{info}([2,3]) = entropy(3/5,2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971\,bits$$

- Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971 = 0.693\,bits$$

Computing Information Gain

- Information gain: information before splitting – information after splitting

$$\text{gain}(\textit{Outlook}\,) = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2])$$
$$= 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

- Information gain for attributes from weather data:

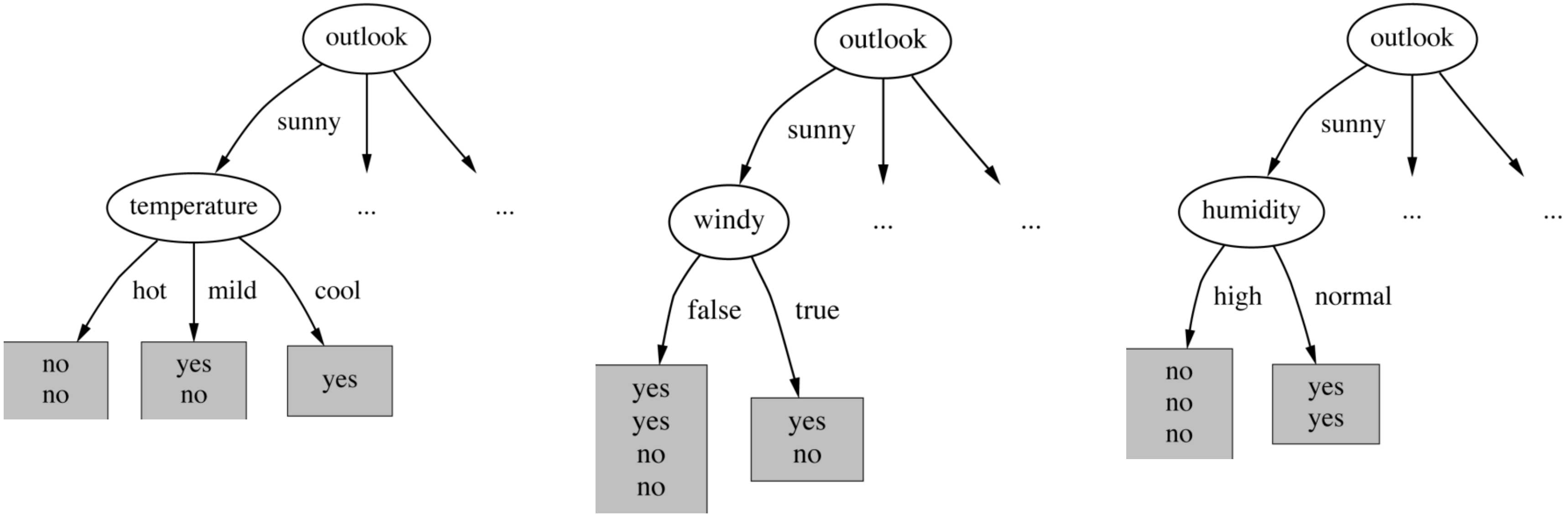
$$\text{gain}(\textit{Outlook}\,) = 0.247 \text{ bits}$$
$$\text{gain}(\textit{Temperature}\,) = 0.029 \text{ bits}$$
$$\text{gain}(\textit{Humidity}\,) = 0.152 \text{ bits}$$
$$\text{gain}(\textit{Windy}\,) = 0.048 \text{ bits}$$

Continuing to Split

gain(*Temperature*) = 0.571 bits

gain(*Humidity*) = 0.971 bits

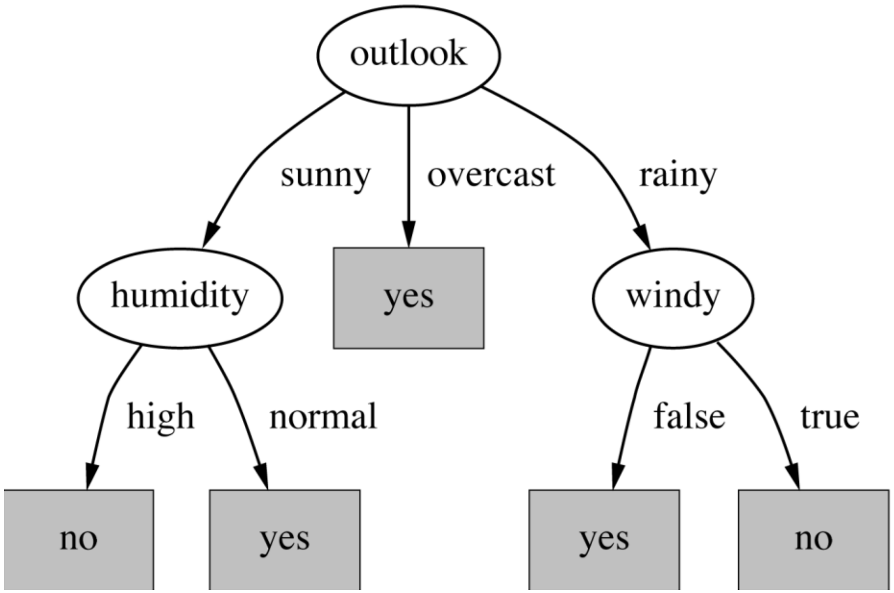
gain(*Windy*) = 0.020 bits



Final Decision Tree

- Note: not all leaves need to be pure; sometimes identical instances have different classes

  ⇒ Splitting stops when data can't be split any further



Wishlist for a Purity Measure

- Properties we require from a purity measure:
  - When node is pure, measure should be zero
  - When impurity is maximal (i.e. all classes equally likely), measure should be maximal
  - Measure should obey *multistage property* (i.e. decisions can be made in several stages):

$$\text{measure}([2,3,4]) = measure([2,7]) + (7/9) \times measure([3,4])$$

- Entropy satisfies all three properties!

Highly-Branching Attributes

- Problematic - attributes with a large number of values
- Subsets are more likely to be pure if there is a large number of values
  - ⇒Information gain is biased towards choosing attributes with a large number of values
  - ⇒This may result in *overfitting* (selection of an attribute that is non-optimal for prediction)

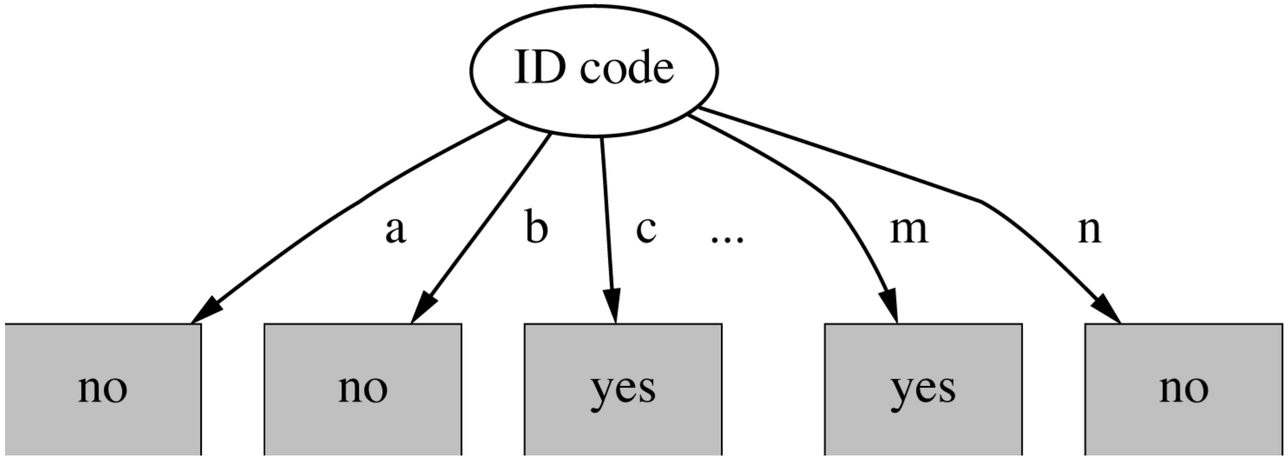
Weather Data with *ID Code*

ID code	Outlook	Temp.	Humidity	Windy	Play
A	Sunny	Hot	High	False	No
B	Sunny	Hot	High	True	No
C	Overcast	Hot	High	False	Yes
D	Rainy	Mild	High	False	Yes
E	Rainy	Cool	Normal	False	Yes
F	Rainy	Cool	Normal	True	No
G	Overcast	Cool	Normal	True	Yes
H	Sunny	Mild	High	False	No
I	Sunny	Cool	Normal	False	Yes
J	Rainy	Mild	Normal	False	Yes
K	Sunny	Mild	Normal	True	Yes
L	Overcast	Mild	High	True	Yes
M	Overcast	Hot	Normal	False	Yes
N	Rainy	Mild	High	True	No

Tree Stump for *ID Code* Attribute

- Entropy of split:

⇒Information gain is maximal for ID code (namely 0.940 bits)



$$\text{info}(\mathit{ID\ code}) = info([0,1]) + info([0,1]) + \cdots + info([0,1]) = 0 bits$$

Gain Ratio

- *Gain ratio*: a modification of the information gain that reduces its bias
- Gain ratio takes number and size of branches into account when choosing an attribute
  - It corrects the information gain by taking the *intrinsic information* of a split into account
- Intrinsic information:
  - Entropy of distribution of instances into branches (i.e. how much info do we need to tell which branch an instance belongs to)

Computing the Gain Ratio

- Example: intrinsic information for ID code

$$\text{info}([1,1,\ldots,1]) = 14 \times (-1/14 \times \log(1/14)) = 3.807\,bits$$

- Value of attribute decreases as intrinsic information gets larger
- Definition of gain ratio:

$$gain\_ratio(attribute) = \frac{gain(attribute)}{intrinsic\_info(attribute)}$$

- Example:

$$\text{gain\_}ratio(ID\ code) = \frac{0.940\,bits}{3.807\,bits} = 0.246$$

Gain Ratios for Weather Data

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	0.157	Gain ratio: 0.029/1.557	0.019

Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

More on the Gain Ratio

- *Outlook* still comes out top
- However *ID code* still has greater gain ratio
  - Standard fix: *ad hoc* test to prevent splitting on that type of attribute
- Problem with gain ratio: it may overcompensate
  - May choose an attribute just because its intrinsic information is very low
  - Standard fix: only consider attributes with greater than average information gain

Walking Through the Weather Example…

1. Calculate the information value of the problem as a whole.
2. For each attribute:
    A. Calculate the information in each of its potential values.
    B. Calculate the average information value of that attribute.
    C. Calculate the gain by subtracting its value from the information value of the problem as a whole.
3. Calculate the intrinsic information value of the split.
4. Calculate the ratio by dividing the attribute gain by the intrinsic information value.

Walking Through the Example…

1. Calculate the information value of the problem as a whole.

info([9,5]) = entropy(9/14, 5/14)

$= -9/14(\log_2 9/14) – 5/14(\log_2 5/14)$

$= -9/14((\log_{10} 9/14)/(\log_{10} 2)) – 5/14((\log_{10} 5/14)/(\log_{10} 2))$

= 0.940 bits

2. For each attribute:

A. Calculate the information in each of its potential values.

Outlook = Sunny

info([2,3]) = entropy(2/5, 3/5)

$= -2/5(\log_2 2/5) - 3/5(\log_2 3/5)$

$= -2/5((\log_{10} 2/5)/(\log_{10} 2)) - 3/5((\log_{10} 3/5)/(\log_{10} 2))$

= 0.971 bits

Outlook = Overcast

info([4,0]) = entropy(4/4, 0/4) = entropy(1, 0)

$= -1(\log_2 1) - 0(\log_2 0)$

$= -1((\log_{10} 1)/(\log_{10} 2)) - 0$

= 0 bits

Outlook = Rainy

info([2,3]) = entropy(2/5, 3/5)

$= -2/5(\log_2 2/5) - 3/5(\log_2 3/5)$

$= -2/5((\log_{10} 2/5)/(\log_{10} 2)) - 3/5((\log_{10} 3/5)/(\log_{10} 2))$

= 0.971 bits

Walking Through the Example…

Walking Through the Example…

2. For each attribute:

B. Calculate the average information value of that attribute.

info([3,2], [4,0], [3,2])

= 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971

= 0.693 bits

Walking Through the Example…

2. For each attribute:

C. Calculate the gain by subtracting its value from the information value of the problem as a whole.

info([9,5]) - info([2,3],[4,0], [2,3])

= 0.940 – 0.693

= 0.247

Walking Through the Example…

3. Calculate the intrinsic information value of the split.

info([5, 4, 5]) = entropy(5/14, 4/14, 5/14)

$= -5/14(\log_2 5/14) – 4/14(\log_2 4/14) – 5/14(\log_2 5/14)$

$= -5/14((\log_{10} 5/14)/(\log_{10} 2)) – 4/14((\log_{10} 4/14)/(\log_{10} 2)) - 5/14((\log_{10} 5/14)/(\log_{10} 2))$

= 1.577 bits

Walking Through the Example…

4. Calculate the ratio by dividing the attribute gain by the intrinsic information value.

Gain Ratio = Gain from Attribute / Intrinsic Value of Split

= 0.247 / 1.577

= 0.157

Walking Through the Example…

- Now you try the math for an attribute, (Temperature, Humidity, or Windy) and see if your numbers come out the same as those listed on slide 19.

Discussion

- Top-down induction of decision trees: ID3, algorithm developed by Ross Quinlan
  - Gain ratio just one modification of this basic algorithm
  - C4.5: deals with numeric attributes, missing values, noisy data
- There are many other attribute selection criteria (but little difference in accuracy of result)