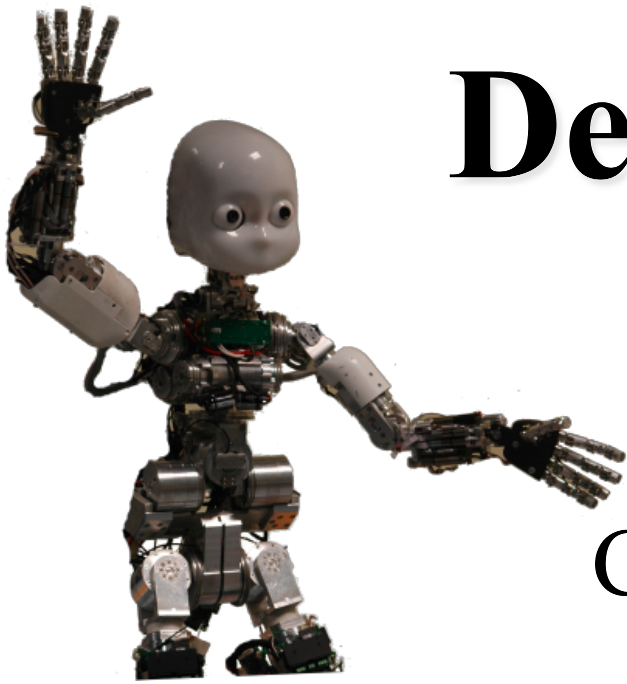
Machine Learning: Decision Trees

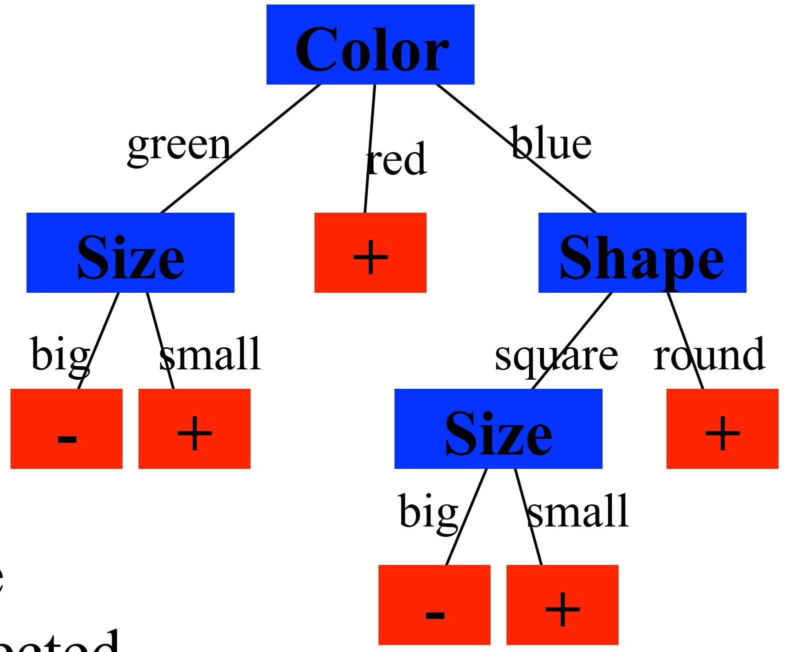


Chapter 18.1-18.3

Some material adopted from notes by Chuck Dyer

Learning decision trees

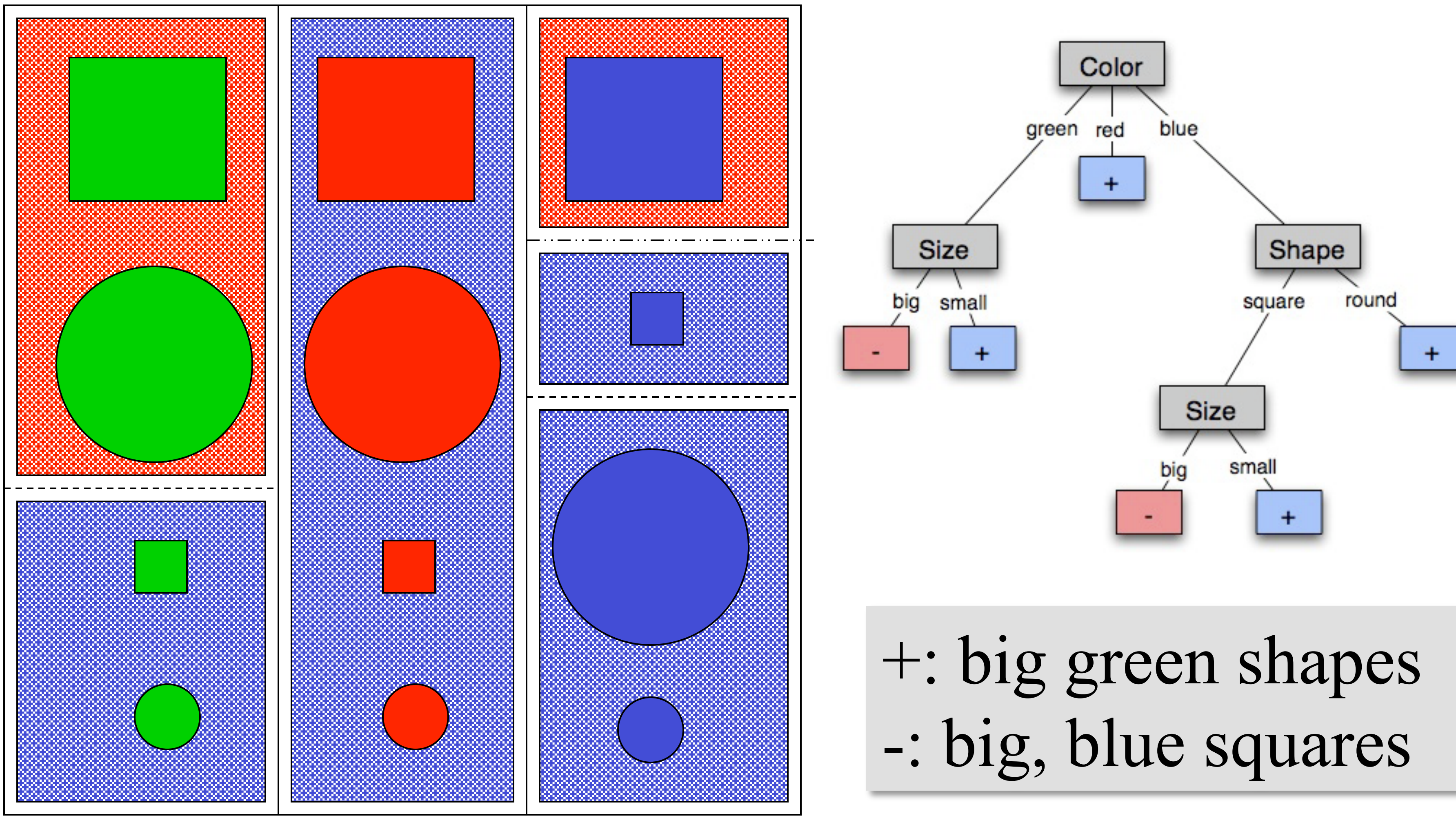
- Goal: Build a **decision tree** to classify examples as positive or negative instances of a concept using supervised learning from a training set
- A **decision tree** is a tree where
  – each non-leaf node has associated with it an attribute (feature)
  –each leaf node has associated with it a classification (+ or -)
  –each arc has associated with it one of the possible values of the attribute at the node from which the arc is directed



- Generalization: allow for >2 classes
  –e.g., for stocks, classify into {sell, hold, buy}

A decision tree-induced partition

The red groups are + examples, blue -

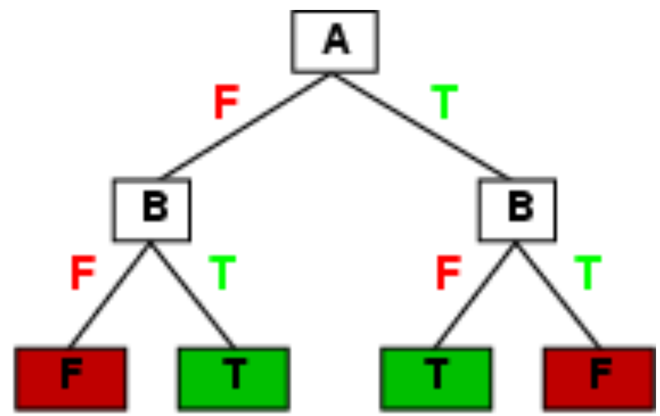


+: big green shapes
-: big, blue squares

Expressiveness of Decision Trees

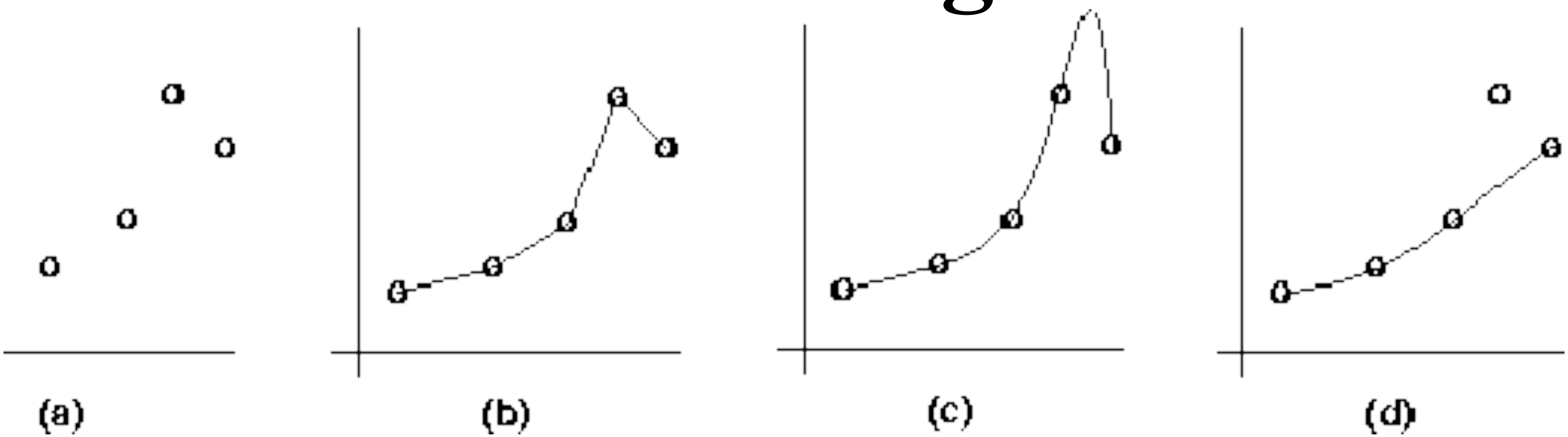
- Can express any function of the input attributes, e.g.
- For Boolean functions, truth table row → path to leaf:

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



- Trivially, there's 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
- We prefer to find more compact decision trees

Inductive learning and bias



- Suppose that we want to learn a function f(x) = y and we are given some sample (x,y) pairs, as in figure (a)
- There are several hypotheses we could make about this function, e.g.: (b), (c) and (d)
- A preference for one over the others reveals the **bias** of our learning technique, e.g.:
  - prefer piece-wise functions
  - prefer a smooth function
  - prefer a simple function and treat outliers as noise

Preference bias: Occam’s Razor

- AKA Occam’s Razor, Law of Economy, or Law of Parsimony
- Principle stated by William of Ockham (1285-1347)
  - “*non sunt multiplicanda entia praeter necessitatem*”
  - entities are not to be multiplied beyond necessity
- The simplest consistent explanation is the best
- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
- Finding the provably smallest decision tree is NP-hard, so instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

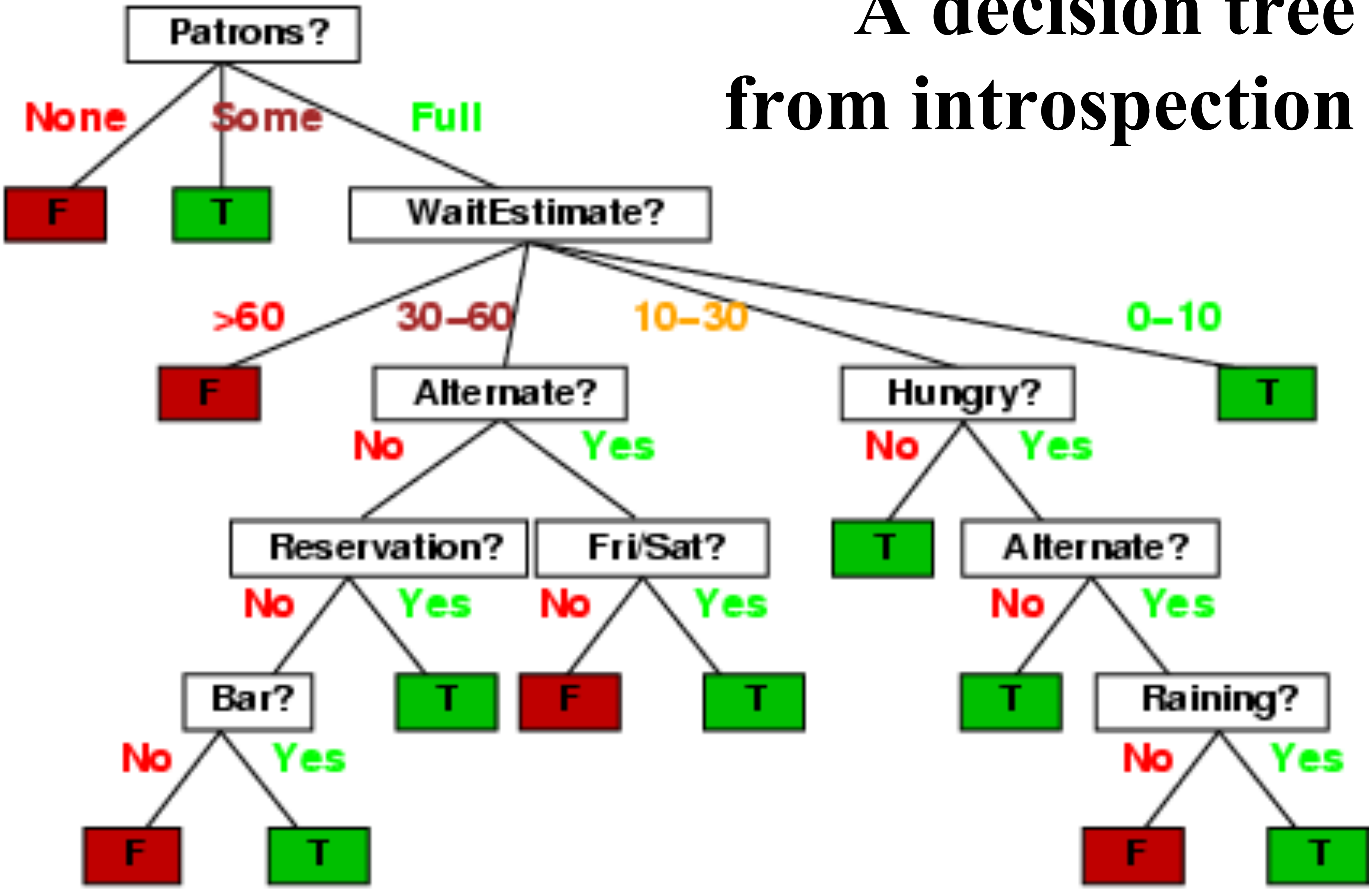
Hypothesis spaces

- **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, 18,446,744,073,709,551,616 trees
- **How many conjunctive hypotheses (e.g., *Hungry* ∧ ¬*Rain*)?**
  - Each attribute can be in (positive), in (negative), or out
    ⇒3^n distinct conjunctive hypotheses
  - e.g., with 6 Boolean attributes, 729 trees
- **A more expressive hypothesis space**
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent with training set
    ⇒ may get worse predictions in practice

R&N's restaurant domain

- Develop decision tree for decision patron makes when deciding whether or not to wait for a table
- Two classes: wait, leave
- Ten attributes: Alternative available? Bar in restaurant? Is it Friday? Are we hungry? How full is the restaurant? How expensive? Is it raining? Do we have reservation? What type of restaurant is it? Estimated waiting time?
- Training set of 12 examples
- ~ 7000 possible cases

A decision tree from introspection



Attribute-based representations

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

- Examples described by attribute values (Boolean, discrete, continuous), e.g., situations where I will/won't wait for a table
- Classification of examples is positive (T) or negative (F)
- Serves as a training set

ID3/C4.5 Algorithm

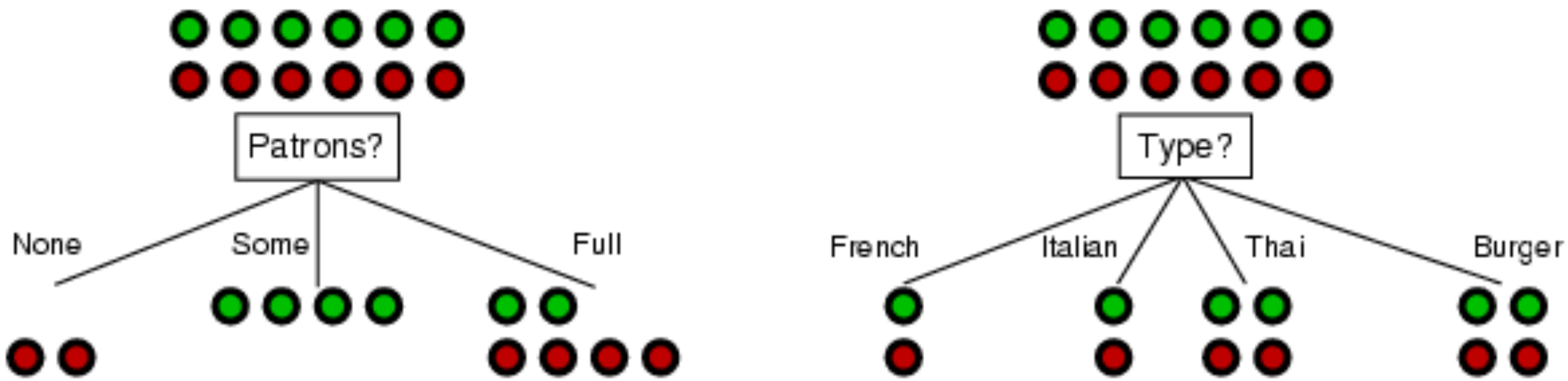
- A greedy algorithm for decision tree construction developed by Ross Quinlan circa 1987
- Top-down construction of tree by recursively selecting "best attribute" to use at the current node in tree
  - Once attribute is selected for current node, generate child nodes, one for each possible value of attribute
  - Partition examples using possible values of attribute, and assign these subsets of the examples to appropriate child node
  - Repeat for each child node until all examples associated with a node are either all positive or all negative

Choosing the best attribute

- Key problem: choosing which attribute to split a given set of examples
- Some possibilities:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose attribute with smallest number of possible values
  - **Most-Values:** Choose attribute with largest number of possible values
  - **Max-Gain:** Choose the attribute that has largest expected *information gain*–i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

Choosing an attribute

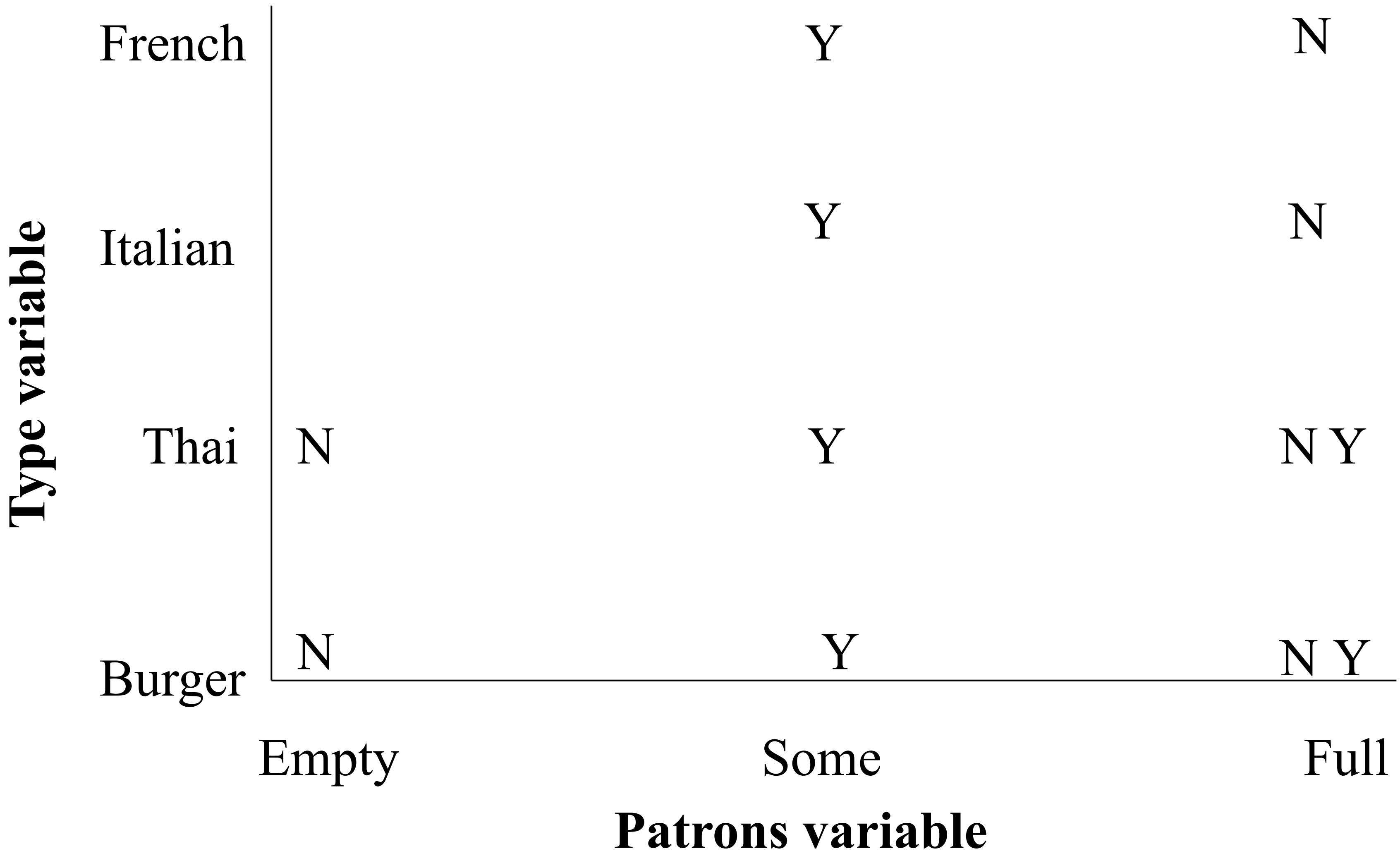
Idea: good attribute splits examples into subsets that are (ideally) *all positive* or *all negative*



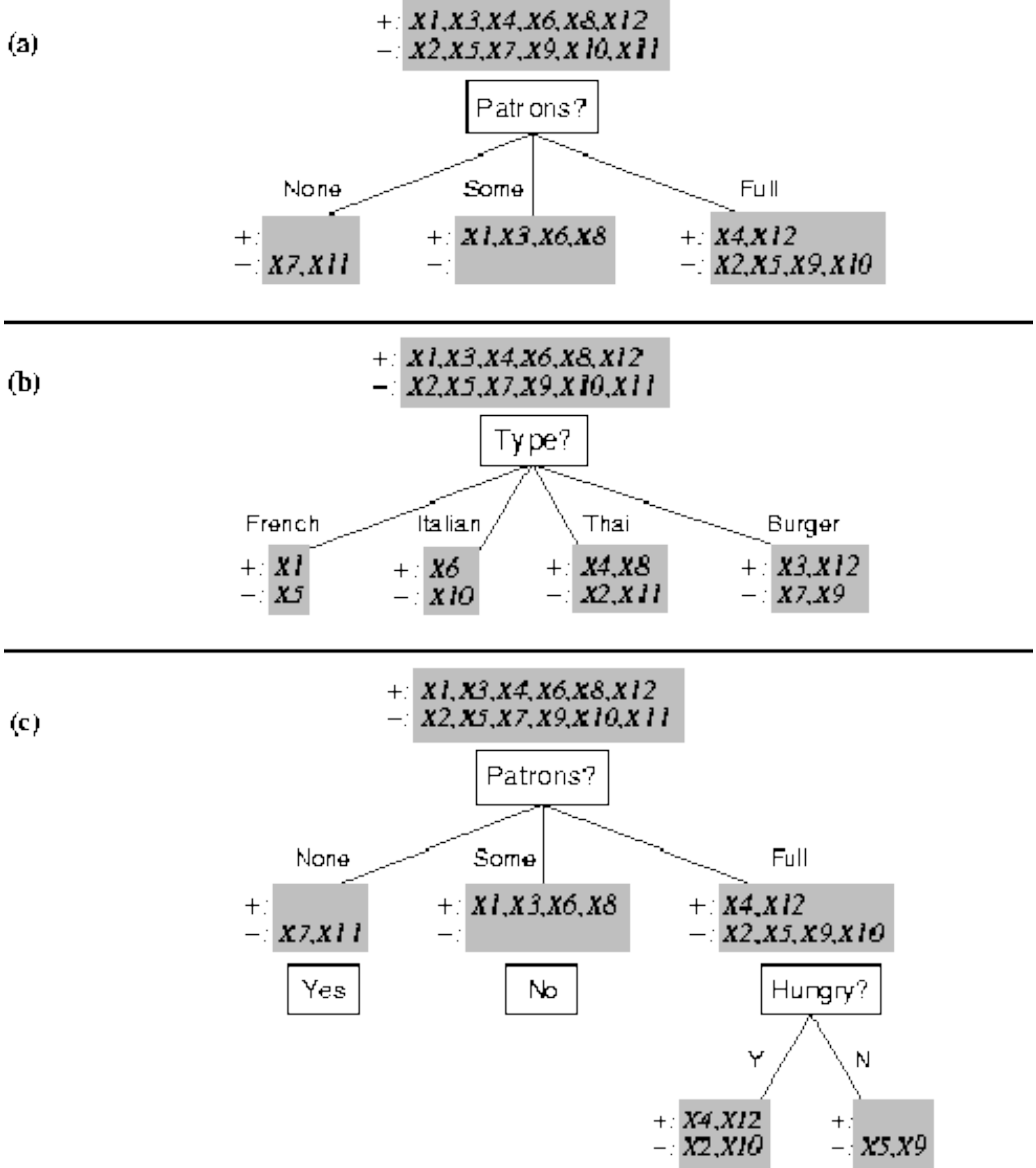
Which is better: *Patrons?* or *Type?*

Restaurant example

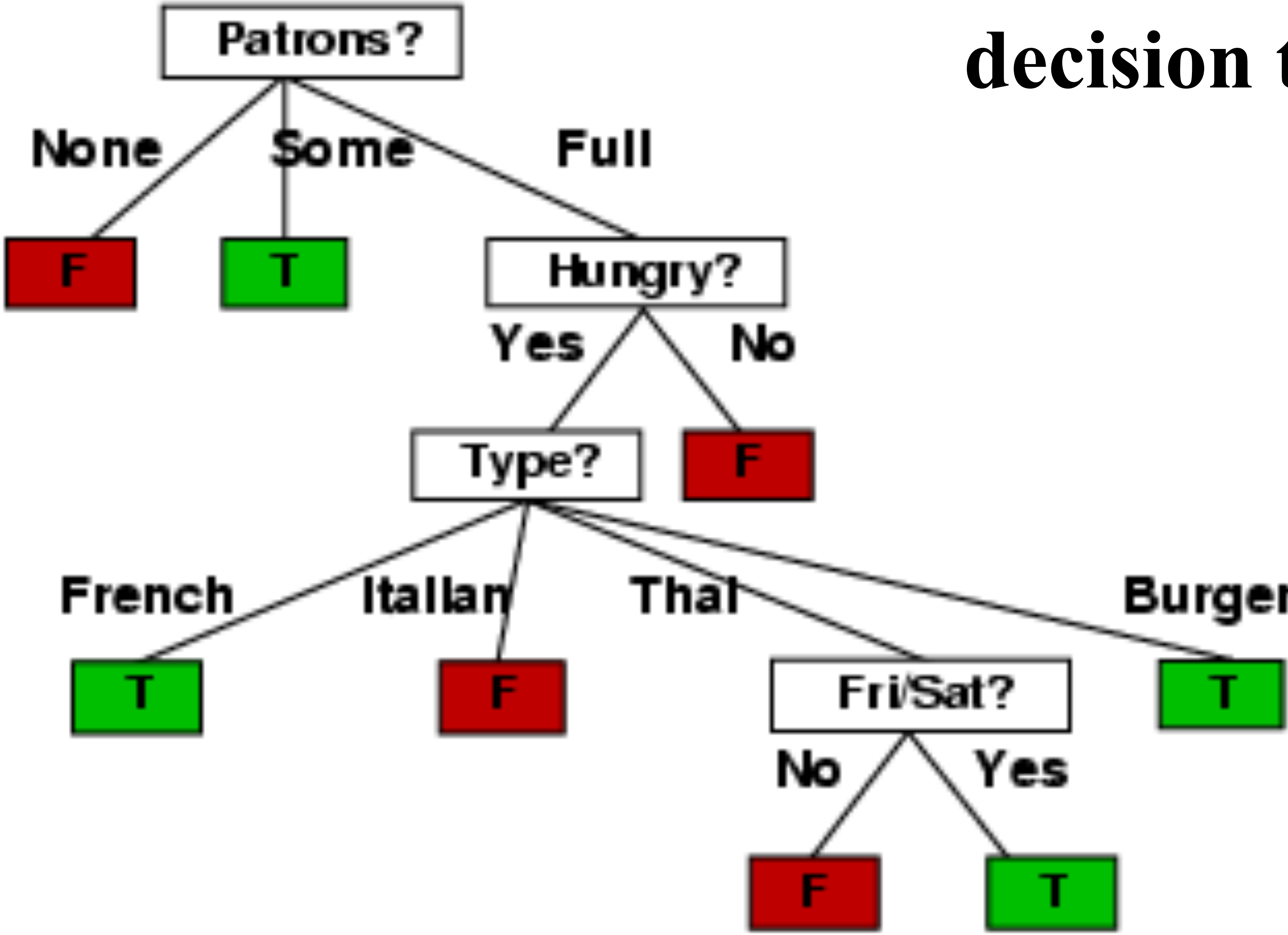
Random: Patrons or Wait-time; **Least-values**: Patrons; **Most-values**: Type; **Max-gain**: ???



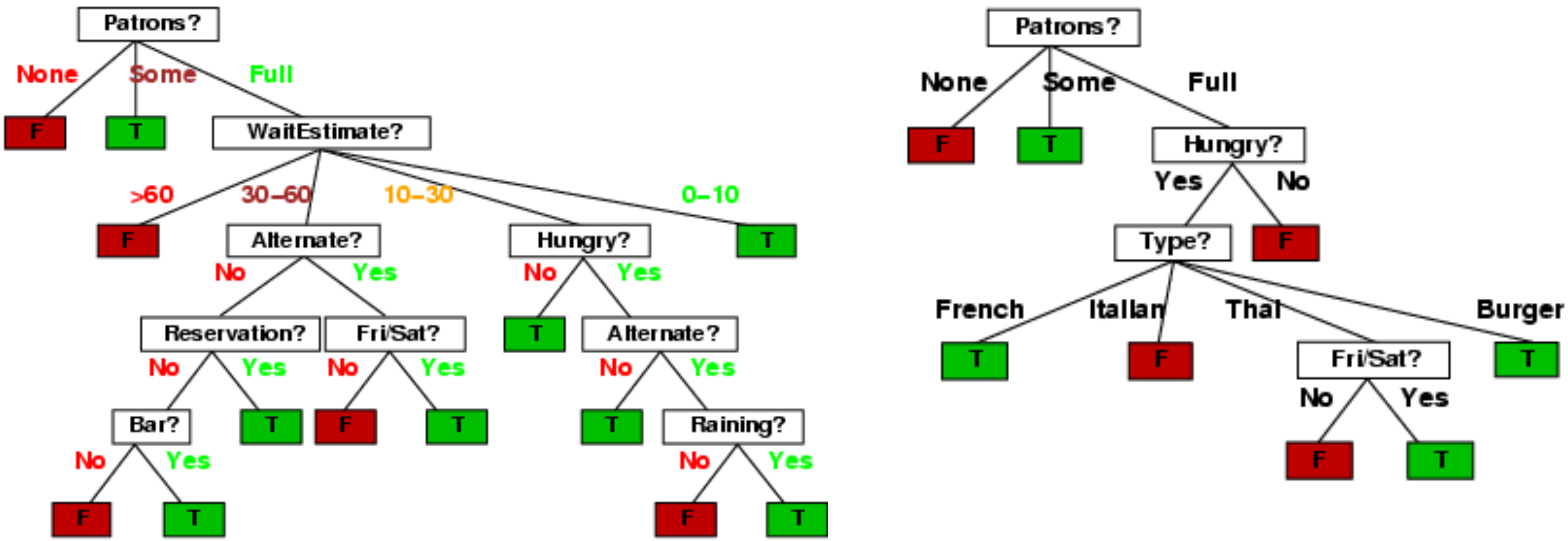
Splitting examples by testing attributes



ID3-induced decision tree



Compare the two Decision Trees



Information theory 101

- Information theory sprang almost fully formed from the seminal work of Claude E. Shannon at Bell Labs

  A Mathematical Theory of Communication, *Bell System Technical Journal*, 1948.

- Intuitions
  - Common words (a, the, dog) shorter than less common ones (parlimentarian, foreshadowing)
  - Morse code: common (probable) letters have shorter encodings
- Information measured in minimum number of bits needed to store or send some information
- The measure of data (information entropy) is the average number of bits needed to storage or send

Information theory 101

- Information is measured in bits
- Information conveyed by message depends on its probability
- For n equally probable possible *messages*, each has prob. *1/n*
- Information conveyed by message is -log(p) = log(n)

  e.g., with 16 messages, then log(16) = 4 and we need 4 bits to identify/send each message

- Given probability distribution for n messages $P = (p_1, p_2 \ldots p_n)$, the information conveyed by distribution (aka *entropy* of P) is:

  $$I(P) = -(\underbrace{p_2}_{\text{probability of msg 2}} \ldots)$$

probability of msg 2

info in msg 2

Information theory II

- Information conveyed by distribution (aka *entropy* of P):

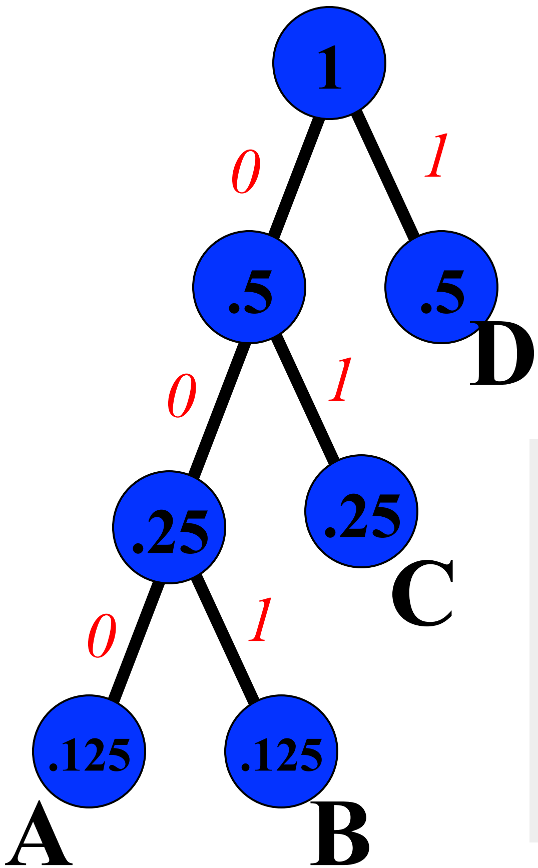
  $I(P) = -(p_1*\log(p_1) + p_2*\log(p_2) + .. + p_n*\log(p_n))$
- Examples:
  - If P is (0.5, 0.5) then I(P) = .5*1 + 0.5*1 = 1
  - If P is (0.67, 0.33) then I(P) = -(2/3*log(2/3) + 1/3*log(1/3)) = 0.92
  - If P is (1, 0) then I(P) = 1*1 + 0*log(0) = 0
- The more uniform the probability distribution, the greater its information: more information is conveyed by a message telling you which event actually occurred
- Entropy is the average number of bits/message needed to represent a stream of messages

Example: Huffman code

- In 1952 MIT student David Huffman devised, in course of doing a homework assignment, an elegant coding scheme which is optimal in the case where all symbols' probabilities are integral powers of 1/2.
- A Huffman code can be built in the following manner:
  - Rank symbols in order of probability of occurrence
  - Successively combine two symbols of the lowest probability to form a new composite symbol; eventually we will build a binary tree where each node is the probability of all nodes beneath it
  - Trace a path to each leaf, noticing direction at each node

Huffman code example

M	P
A	.125
B	.125
C	.25
D	.5



M	code	length	prob	
A	000	3	0.125	0.375
B	001	3	0.125	0.375
C	01	2	0.250	0.500
D	1	1	0.500	0.500
average message length				1.750

If we use this code to many messages (A,B,C or D) with this probability distribution, then, over time, the average bits/message should approach 1.75

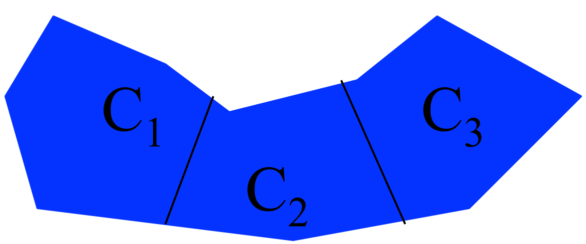
Information for classification

If a set T of records is partitioned into disjoint exhaustive classes $(C_1, C_2, .., C_k)$ on the basis of the value of the class attribute, then information needed to identify class of an element of T is:

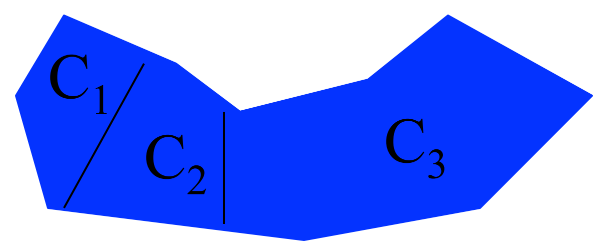
$$\text{Info}(T) = I(P)$$

where P is the probability distribution of partition $(C_1, C_2, .., C_k)$:

$$P = (|C_1|/|T|, |C_2|/|T|, ..., |C_k|/|T|)$$



High information

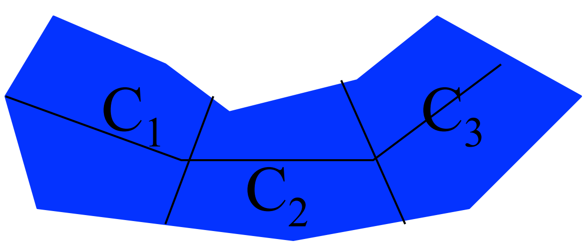


Low information

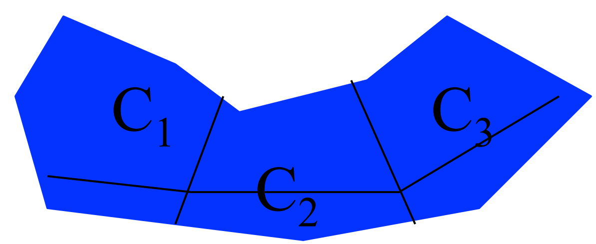
Information for classification II

If we partition T wrt attribute X into sets $\{T_1,T_2, ..,T_n\}$, the information needed to identify class of an element of T becomes the weighted average of the information needed to identify the class of an element of T_i, i.e. the weighted average of $Info(T_i)$:

$$Info(X,T) = \sum |T_i|/|T| * Info(T_i)$$



High information



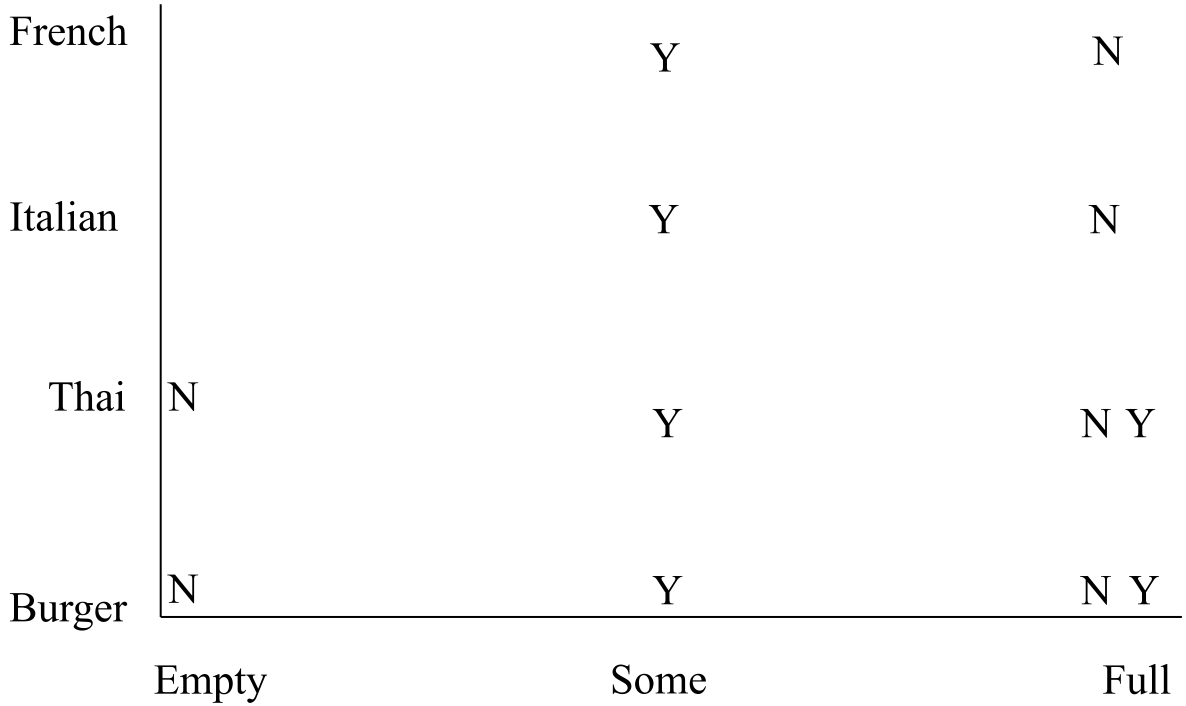
Low information

Information gain

- **Gain(X,T) = Info(T) - Info(X,T)** is difference between
  - info needed to identify element of T and
  - info needed to identify element of T after value of attribute X known
- This is the **gain in information due to attribute X**
- Use to rank attributes and build DT where each node uses attribute with greatest gain of those not yet considered (in path from root)
- The intent of this ordering is to Create small DTs to minimize questions

Computing Information Gain

- I(T) = ?
- I (Pat, T) = ?
- I (Type, T) = ?



Gain (Pat, T) = ?

Gain (Type, T) = ?

Computing information gain

I(T) =

- (.5 log .5 + .5 log .5)

= .5 + .5 = 1

I (Pat, T) =

2/12 (0) + 4/12 (0) +

6/12 (- (4/6 log 4/6 +

2/6 log 2/6))

= 1/2 (2/3*.6 +

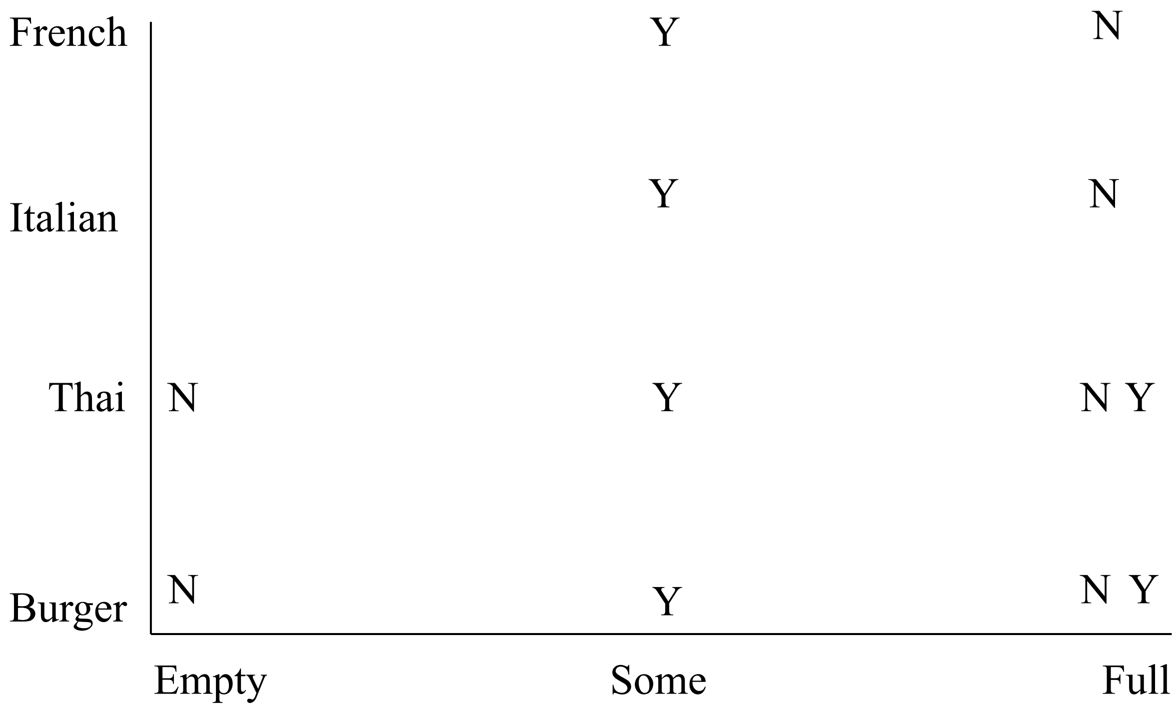
1/3*1.6)

= .47

I (Type, T) =

2/12 (1) + 2/12 (1) +

4/12 (1) + 4/12 (1) = 1



Gain (Pat, T) = 1 - .47 = .53

Gain (Type, T) = 1 – 1 = 0

The ID3 algorithm builds a decision tree, given a set of non-categorical attributes C1, C2, .., Cn, the class attribute C, and a training set T of records

```
function ID3(R:input attributes, C:class attribute,
S:training set) returns decision tree;
   If S is empty, return single node with value Failure;
   If every example in S has same value for C, return
   single node with that value;
   If R is empty, then return a single node with most
   frequent of the values of C found in examples S;
   # causes errors -- improperly classified record
   Let D be attribute with largest Gain(D,S) among R;
   Let {dj| j=1,2, .., m} be values of attribute D;
   Let {Sj| j=1,2, .., m} be subsets of S consisting of
             records with value dj for attribute D;
   Return tree with root labeled D and arcs labeled
     d1..dm going to the trees ID3(R-{D},C,S1). . .
     ID3(R-{D},C,Sm);
```

How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

Extensions of ID3

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

Using gain ratios

- The information gain criterion favors attributes that have a large number of values
  - If we have an attribute D that has a distinct value for each record, then Info(D,T) is 0, thus Gain(D,T) is maximal
- To compensate for this Quinlan suggests using the following ratio instead of Gain:

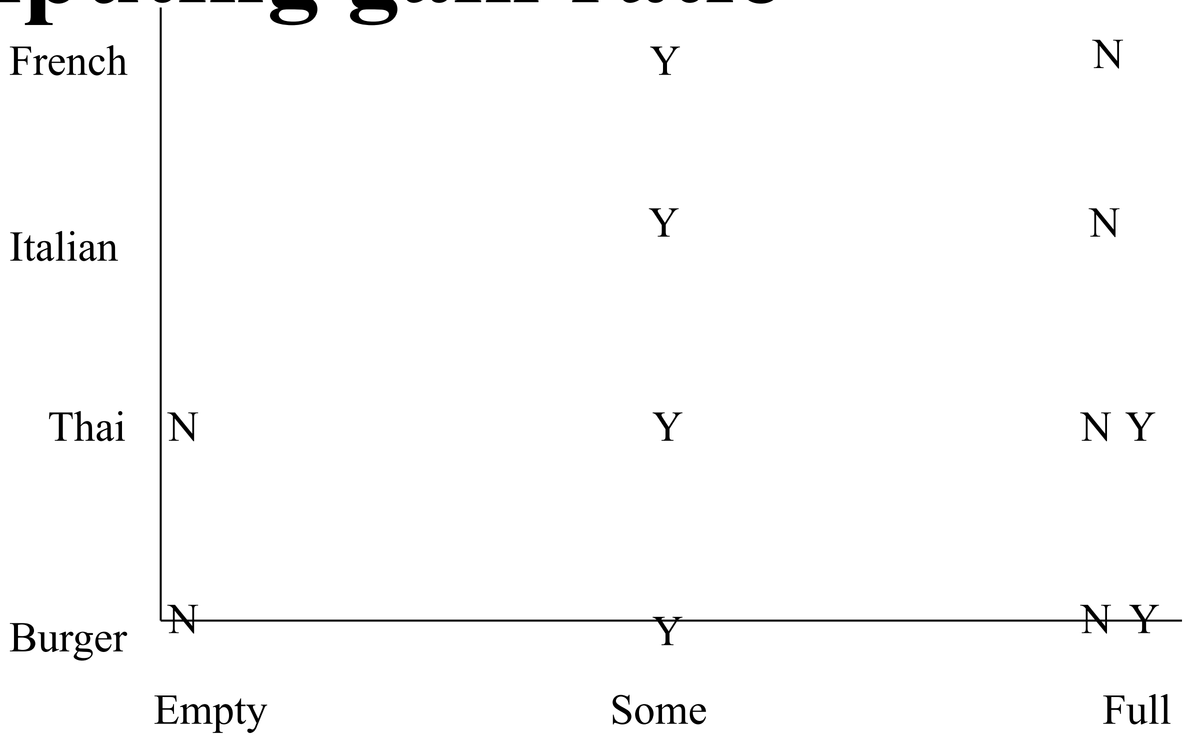
  GainRatio(D,T) = Gain(D,T) / SplitInfo(D,T)
- SplitInfo(D,T) is the information due to the split of T on the basis of value of categorical attribute D

  SplitInfo(D,T) = I(|T1|/|T|, |T2|/|T|, .., |Tm|/|T|)

where {T1, T2, .. Tm} is the partition of T induced by value of D

Computing gain ratio

- $I(T) = 1$
- $I(Pat, T) = .47$
- $I(Type, T) = 1$



Gain (Pat, T) =.53
Gain (Type, T) = 0

SplitInfo (Pat, T) = - (1/6 log 1/6 + 1/3 log 1/3 + 1/2 log 1/2) = 1/6*2.6 + 1/3*1.6 + 1/2*1 = 1.47

SplitInfo (Type, T) = 1/6 log 1/6 + 1/6 log 1/6 + 1/3 log 1/3 + 1/3 log 1/3 = 1/6*2.6 + 1/6*2.6 + 1/3*1.6 + 1/3*1.6 = 1.93

GainRatio (Pat, T) = Gain (Pat, T) / SplitInfo(Pat, T) = .53 / 1.47 = .36

GainRatio (Type, T) = Gain (Type, T) / SplitInfo (Type, T) = 0 / 1.93 = 0

Real-valued data

- Select set of thresholds defining intervals
- Each becomes a discrete value of attribute
- Use some simple heuristics, e.g. always divide into quartiles
- Use domain knowledge…
  - divide age into infant (0-2), toddler (3-5), school-aged (5-8)
- Or treat this as another learning problem:
  - Try different ways to discretize the continuous variable; see which yield better results w.r.t. some metric
  - E.g., try midpoint between every pair of values

Noisy data

- Many kinds of “noise” can occur in the examples:
- Two examples have same attribute/value pairs, but different classifications
- Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
- The classification is wrong (e.g., + instead of -) because of some error
- Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome

Overfitting

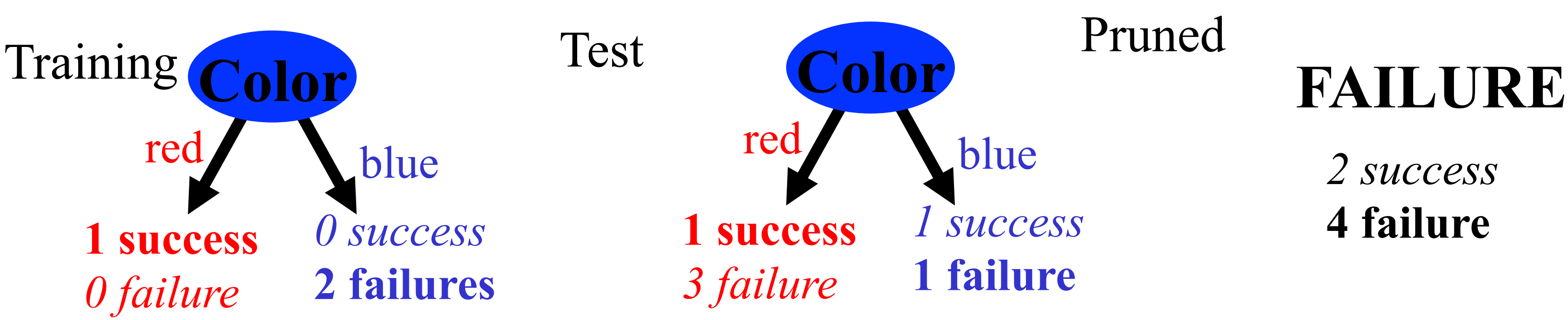
- Irrelevant attributes, can result in *overfitting* the training example data
- If hypothesis space has many dimensions (large number of attributes), we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features
- If we have too little training data, even a reasonable hypothesis space will 'overfit'

Overfitting

- Fix by by removing irrelevant features
  - E.g., remove ‘year observed’, ‘month observed’, ‘day observed’, ‘observer name’ from feature vector
- Fix by getting more training data
- Fix by pruning lower nodes in the decision tree
  - E.g., if gain of the best attribute at a node is below a threshold, stop and make this node a leaf rather than generating children nodes

Pruning decision trees

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. E.g.,
  - Training: one training red success and two training blue failures
  - Test: three red failures and one blue success
  - Consider replacing this subtree by a single Failure node.
- After replacement we will have only two errors instead of five:



Converting decision trees to rules

- It is easy to derive rules from a decision tree: write a rule for each path from the root to a leaf
- In that rule the left-hand side is built from the label of the nodes and the labels of the arcs
- The resulting rules set can be simplified:
  - Let LHS be the left hand side of a rule
  - LHS' obtained from LHS by eliminating some conditions
  - Replace LHS by LHS' in this rule if the subsets of the training set satisfying LHS and LHS' are equal
  - A rule may be eliminated by using meta-conditions such as "if no other rule applies"

http://archive.ics.uci.edu/ml

UCI Machine Learning Repos ×

archive.ics.uci.edu/ml/

About Citation Policy Donate a Data Set Contact

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Google™ Custom Search | Search

View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 233 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. Our old web site is still available, for those who prefer the old format. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians. We have also set up a mirror site for the Repository.

Supported By: In Collaboration With: Rexa.info • Research • People • Connections

233 data sets

Latest News:

2010-03-01: Note from donor regarding Netflix data
2009-10-16: Two new data sets have been added.
2009-09-14: Several data sets have been added.
2008-07-23: Repository mirror has been set up.
2008-03-24: New data sets have been added!
2007-06-25: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope
2007-04-13: Research papers that cite the repository have been associated to specific data sets.

Featured Data Set: Yeast

Task: Classification
Data Type: Multivariate
 # Attributes: 8
 # Instances: 1484

Predicting the Cellular Localization Sites of Proteins

Newest Data Sets:

2012-10-21: QtyT40I10D100K
2012-10-19: Legal Case Reports
2012-09-29: seeds
2012-08-30: Individual household electric power consumption
2012-08-15: Northix
2012-08-06: PAMAP2 Physical Activity Monitoring
2012-08-04: Restaurant & consumer data
2012-08-03: CNAE-9

Most Popular Data Sets (hits since 2007):

386214: Iris
272233: Adult
237503: Wine
195947: Breast Cancer Wisconsin (Diagnostic)
182423: Car Evaluation
151635: Abalone
135419: Poker Hand
113024: Forest Fires

UCI Machine Learning Repository: Zoo Data Set

http://archive.ics.uci.edu/ml/datasets/Zoo

About Citation Policy Donate a Data Set Contact

Search

Repository Web

Google™

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Zoo Data Set

Download: Data Folder, Data Set Description

Abstract: Artificial, 7 classes of animals

http://archive.ics.uci.edu/ml/datasets/Zoo

Data Set Characteristics:	Multivariate	**Number of Instances:**	101	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	17	**Date Donated**	1990-05-15
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	18038

Zoo data

animal name: string
hair: Boolean
feathers: Boolean
eggs: Boolean
milk: Boolean
airborne: Boolean
aquatic: Boolean
predator: Boolean
toothed: Boolean
backbone: Boolean
breathes: Boolean
venomous: Boolean
fins: Boolean
legs: {0,2,4,5,6,8}
tail: Boolean
domestic: Boolean
catsize: Boolean
type: {mammal, fish, bird, shellfish, insect, reptile, amphibian}

101 examples

aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
bear,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
boar,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
buffalo,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
calf,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1,mammal
carp,0,0,1,0,0,1,0,1,1,0,0,1,0,1,1,0,fish
catfish,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
cavy,1,0,0,1,0,0,0,1,1,1,0,0,4,0,1,0,mammal
cheetah,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
chicken,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0,bird
chub,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
clam,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,shellfish
crab,0,0,1,0,0,1,1,0,0,0,0,0,4,0,0,0,shellfish
…

Zoo example

```
aima-python> python
>>> from learning import *
>>> zoo
<DataSet(zoo): 101 examples, 18 attributes>
>>> dt = DecisionTreeLearner()
>>> dt.train(zoo)
>>> dt.predict(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0]) #eggs=1
'fish'
>>> dt.predict(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0]) #eggs=0
'mammal'
```

Zoo example

```
>> dt.dt
DecisionTree(13, 'legs', {0: DecisionTree(12, 'fins', {0:
DecisionTree(8, 'toothed', {0: 'shellfish', 1: 'reptile'}), 1:
DecisionTree(3, 'eggs', {0: 'mammal', 1: 'fish'})}), 2:
DecisionTree(1, 'hair', {0: 'bird', 1: 'mammal'}), 4:
DecisionTree(1, 'hair', {0: DecisionTree(6, 'aquatic', {0:
'reptile', 1: DecisionTree(8, 'toothed', {0: 'shellfish', 1:
'amphibian'})}), 1: 'mammal'}), 5: 'shellfish', 6:
DecisionTree(6, 'aquatic', {0: 'insect', 1: 'shellfish'}), 8:
'shellfish'})
```

Zoo example

```
>>> dt.dt.display()
Test legs
 legs = 0 ==> Test fins
    fins = 0 ==> Test toothed
       toothed = 0 ==> RESULT =  shellfish
       toothed = 1 ==> RESULT =  reptile
    fins = 1 ==> Test eggs
       eggs = 0 ==> RESULT =  mammal
       eggs = 1 ==> RESULT =  fish
 legs = 2 ==> Test hair
    hair = 0 ==> RESULT =  bird
    hair = 1 ==> RESULT =  mammal
 legs = 4 ==> Test hair
    hair = 0 ==> Test aquatic
       aquatic = 0 ==> RESULT =  reptile
       aquatic = 1 ==> Test toothed
          toothed = 0 ==> RESULT =  shellfish
          toothed = 1 ==> RESULT =  amphibian
    hair = 1 ==> RESULT =  mammal
 legs = 5 ==> RESULT =  shellfish
 legs = 6 ==> Test aquatic
    aquatic = 0 ==> RESULT =  insect
    aquatic = 1 ==> RESULT =  shellfish
 legs = 8 ==> RESULT =  shellfish
```

Zoo example

```
>>> dt.dt.display()
Test legs
 legs = 0 ==> Test fins
    fins = 0 ==> Test toothed
       toothed = 0 ==> RESULT =  shellfish
       toothed = 1 ==> RESULT =  reptile
    fins = 1 ==> Test milk
       milk = 0 ==> RESULT =  fish
       milk = 1 ==> RESULT =  mammal
 legs = 2 ==> Test hair
    hair = 0 ==> RESULT =  bird
    hair = 1 ==> RESULT =  mammal
 legs = 4 ==> Test hair
    hair = 0 ==> Test aquatic
       aquatic = 0 ==> RESULT =  reptile
       aquatic = 1 ==> Test toothed
          toothed = 0 ==> RESULT =  shellfish
          toothed = 1 ==> RESULT =  amphibian
    hair = 1 ==> RESULT =  mammal
 legs = 5 ==> RESULT =  shellfish
 legs = 6 ==> Test aquatic
    aquatic = 0 ==> RESULT =  insect
    aquatic = 1 ==> RESULT =  shellfish
 legs = 8 ==> RESULT =  shellfish
```

Add the shark example to the training set and retrain

Summary: Decision tree learning

- Widely used learning methods in practice
- Can out-perform human experts in many problems
- Strengths include
  – Fast and simple to implement
  – Can convert result to a set of easily interpretable rules
  – Empirically valid in many commercial products
  – Handles noisy data
- Weaknesses include
  – Univariate splits/partitioning using only one attribute at a time so limits types of possible trees
  – Large decision trees may be hard to understand
  – Requires fixed-length feature vectors
  – Non-incremental (i.e., batch method)