ML 2: Information Theory

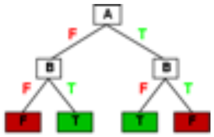
Cynthia Matuszek – CMSC 671

Material from Dr. Marie desJardin, Dr. Manfred Kerber,

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

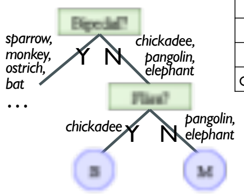


- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples
- We prefer to find more **compact** decision trees!

Bird or Mammal?

We should have split on feathers first!

1. S
2. G
3. P
4. Assign examples to child
5. Repeat until examples are +ve or -ve



Examples (training data)	Attributes			Outcome
	Bi-pedal	Flies	Feath-ers	
Sparrow	Y	Y	Y	B
Monkey	Y	N	N	M
Ostrich	Y	N	Y	B
Pangolin	N	N	N	M
Bat	Y	Y	N	M
Elephant	N	N	N	M
Chickadee	N	Y	Y	B

<u>Test</u>
mouse: <B:N, Fl:N, Fe:N>

ID3/C4.5

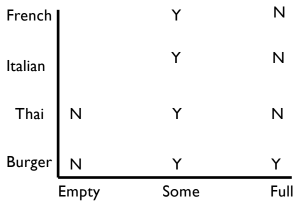
- A **greedy** algorithm for decision tree construction
  - Ross Quinlan, 1987
- Construct decision tree top-down by recursively selecting the "best attribute" to use at current node
  1. Select attribute for current node
  2. Generate child nodes (one for each possible value of attribute)
  3. Partition training data using attribute values
  4. Assign subsets of examples to the appropriate child node
  5. Repeat for each child node until all examples associated with a node are either all positive or all negative

Choosing the Best Attribute

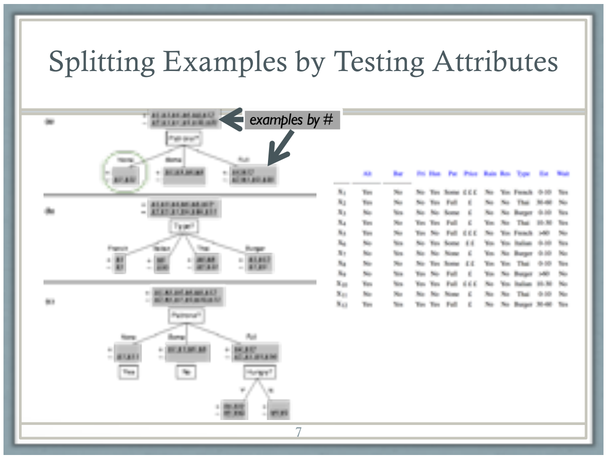
- **Key problem:** what attribute to split on?
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose attribute with smallest number of values
  - **Most-Values:** Choose attribute with largest number of values
  - **Max-Gain:** Choose attribute that has the largest expected information gain—the attribute that will result in the smallest expected size of the subtrees rooted at its children
- ID3 uses Max-Gain to select the best attribute

Restaurant Example

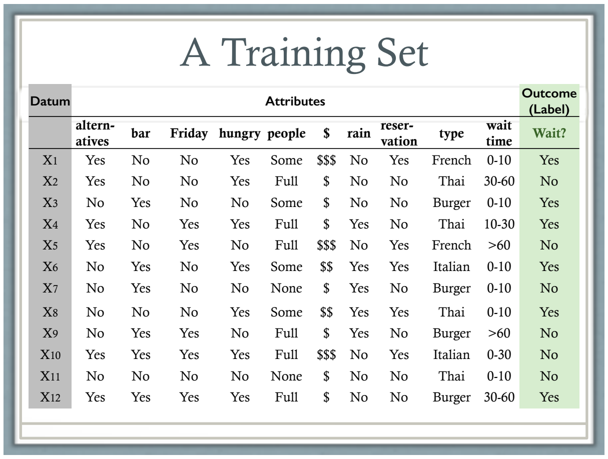
- What do these approaches split restaurants on, given the data in the table?
  - **Random:** Patrons or Type
  - **Least-values**: Patrons
  - **Most-values:** Type
  - **Max-gain:** ???



	Empty	Some	Full
French		Y	N
Italian		Y	N
Thai	N	Y	N
Burger	N	Y	Y

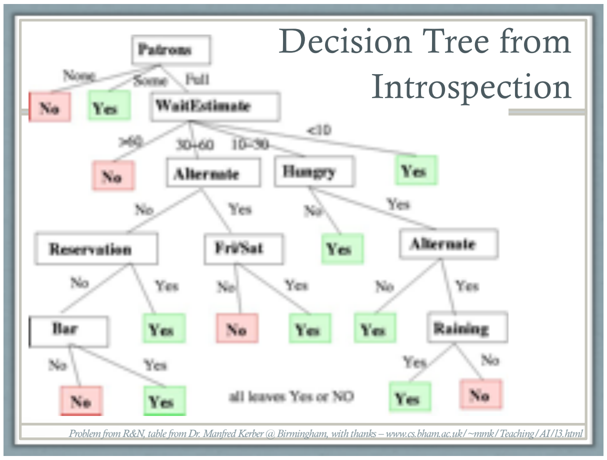


Splitting Examples by Testing Attributes



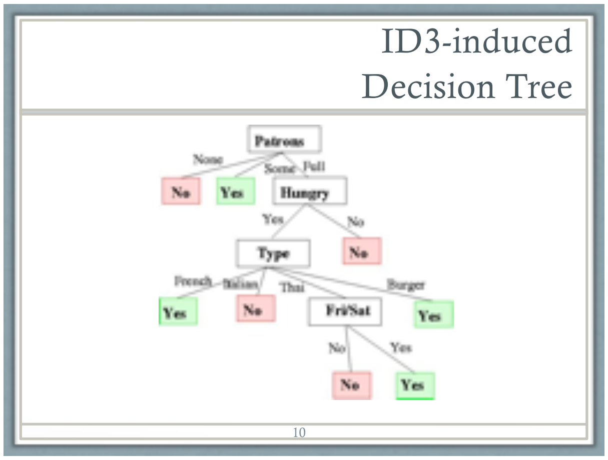
A Training Set

Datum	Attributes										Outcome (Label)
	altern-atives	bar	Friday	hungry	people	$	rain	reser-vation	type	wait time	Wait?
X1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
X3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
X4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10-30	Yes
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
X7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No
X8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	0-30	No
X11	No	No	No	No	None	$	No	No	Thai	0-10	No
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes

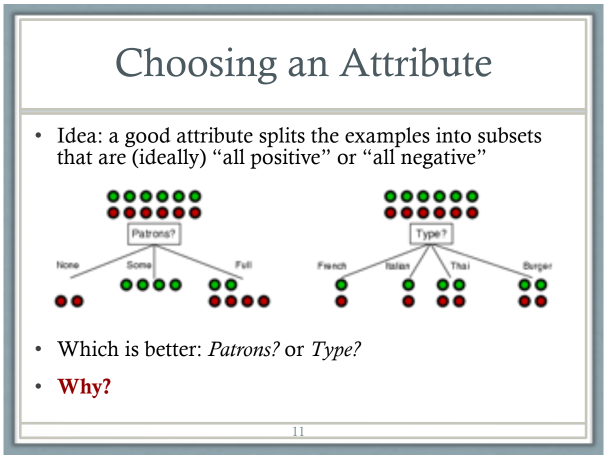


Decision Tree from Introspection

Problem from R&N, table from Dr. Manfred Kerber @ Birmingham, with thanks – www.cs.bham.ac.uk/~mmk/Teaching/AI/l3.html



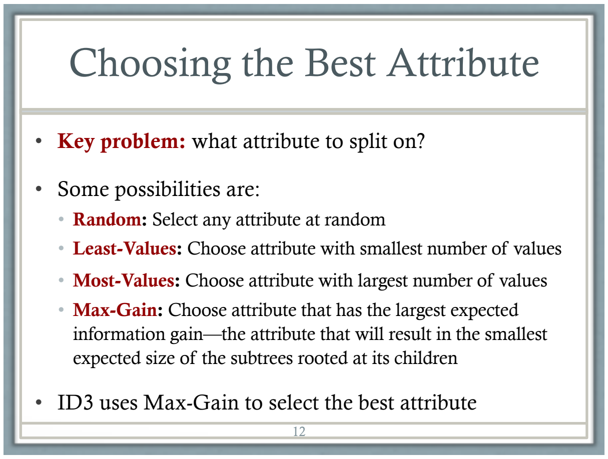
ID3-induced Decision Tree



Choosing an Attribute

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

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



Choosing the Best Attribute

- **Key problem:** what attribute to split on?
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose attribute with smallest number of values
  - **Most-Values:** Choose attribute with largest number of values
  - **Max-Gain:** Choose attribute that has the largest expected information gain—the attribute that will result in the smallest expected size of the subtrees rooted at its children
- ID3 uses Max-Gain to select the best attribute

Restaurant Example

- What do these approaches split restaurants on, given the data in the table?
  - **Random:**
  - **Least-values**:
  - **Most-values:**
  - **Max-gain:**

	Empty	Some	Full
French		Y	N
Italian		Y	N
Thai	N	Y	N
Burger	N	Y	Y

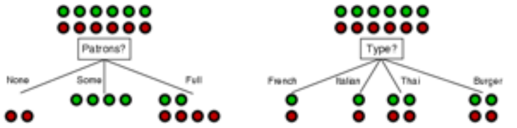
Information Theory 101

- **Information:** the **minimum number of bits** needed to store or send some information
  - Wikipedia: "The measure of data, known as information entropy, is usually expressed by the *average* number of bits needed for storage or communication"
- Intuition: minimize effort to communicate/store
  - Common words (a, the, dog) are shorter than less common ones (parliamentarian, foreshadowing)
  - In Morse code, common (probable) letters have shorter encodings

"A Mathematical Theory of Communication," Bell System Technical Journal, 1948, Claude E. Shannon, Bell Labs

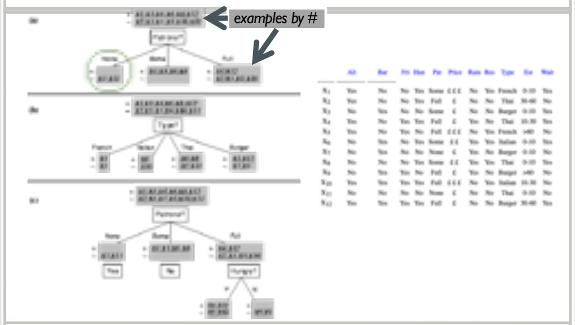
Choosing an Attribute

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



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

Splitting Examples by Testing Attributes



Information Theory 102

- Information is measured in **bits.**
- Information in a message depends on its probability.
- Given n equally probable possible messages, what is probability p_n of each one?

  1/n
- Information conveyed by a message is:

  $\log_2(n) = -\log_2(p)$
- Example: with 16 possible messages, $\log_2(16) = 4$, and we need 4 bits to identify/send each message

Information Theory 102.b

- Information conveyed by a message is

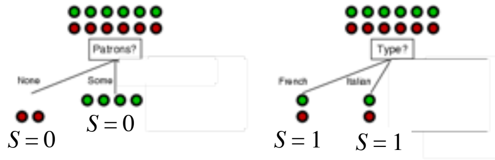
  $\log_2(n) = -\log_2(p)$
- Given a probability distribution for n messages:

  $P = (p_1,p_2...p_n)$
- The information conveyed by that distribution is:

  $I(P) = -(p_1*\log_2(p_1) + p_2*\log_2(p_2) + .. + p_n*\log_2(p_n))$
- This is the **entropy** of P.

Entropy Interlude

- Entropy (S): the homogeneity of a sample
  - If everything is the same, $S = 0$
  - If differences are even $S = 1$



Information Theory 103

- Entropy: **average** number of bits (per message) needed to represent a stream of messages

  $I(P) = -(p_1 * \log_2 (p_1) + p_2 * \log_2 (p_2) + .. + p_n * \log_2 (p_n))$
- Examples:
  - P = (0.5, 0.5) : I(P) = 1 → entropy of a fair coin flip
  - P = (0.67, 0.33) : I(P) = 0.92
  - P = (0.99, 0.01) : I(P) = 0.08
  - P = (1, 0) : I(P) = 0
- **As the distribution becomes more skewed, the amount of information *decreases*. Why?**
- **Because I can just predict the most likely element, and usually be right**

Entropy as Measure of **Homogeneity of Examples**

- Entropy can be used to characterize the (im)purity of an arbitrary collection of examples
- **Low entropy** implies **high homogeneity**
  - Given a collection S (like the table of 12 examples for the restaurant domain), containing positive and negative examples of some target concept, the entropy of S relative to its Boolean classification is:

    $I(S) = -(p_+ * \log_2 (p_+) + p_- * \log_2 (p_-))$

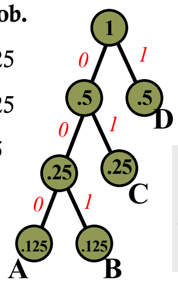
    Entropy([6+, 6-]) = 1 → entropy of the restaurant dataset
    Entropy([9+, 5-]) = 0.940

Huffman Code

- In 1952 MIT student David Huffman devised (while doing a homework assignment!) an encoding which is optimal when all symbols' probabilities are integral powers of ½
- To build a Huffman code:
  - Rank all symbols in order of probability of occurrence
  - Successively combine the 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 the direction at each node

Huffman Code Example

Msg.	**Prob.**
A	.125
B	.125
C	.25
D	.5

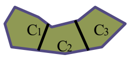


M	code	length	prob	
A	000	3	0.125	0.375
B	001	3	0.125	0.375
C	01	2	0.250	0.500
D	1	1	0.500	0.500
average message length				1.750

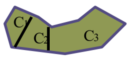
If we use this code to send many messages (A,B,C or D) with this probability distribution, then, over time, the average bits/message should approach 1.75

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:
  - Information needed to identify the class of an element of T is: Info(T) = I(P)
- 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

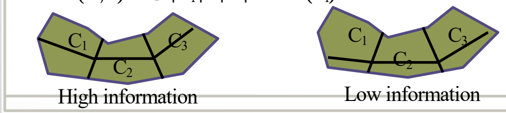


Low information

Information for Classification II

- Partition T w.r.t attribute X into sets $\{T_1,T_2, ..,T_n\}$
- To identify the class of some element of T, we need the weighted average of the information needed to identify the class of an element of T_i, i.e. the weighted average of $\text{Info}(T_i)$:

$\text{Info}(X,T) = S\,|T_i|/|T| * \text{Info}(T_i)$



Information Gain

- A chosen attribute A divides the training set E into subsets $E_1, \dots , E_v$ according to their values for A, where A has v distinct values.
- IG(S,A)—the ***information gain*** of an attribute A relative to a collection of examples S—is:

$$\text{Gain}(S,A) = I(S) - \textit{Remainder}(A) \qquad remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- This represents the difference between
  - I(S) – the entropy of the original collection S
  - *Remainder*(A) - expected value of the entropy after S is partitioned using attribute A
- This is the **gain in information due to attribute A**
  - Expected reduction in entropy
  - IG(S,A) or simply IG(A):

$$IG(S,A) = I(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times I(S_v) \qquad IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

Information Gain

- **Information gain**: how much entropy decreases (homogeneity increases) when a dataset is split on an attribute.
  - High homogeneity → high likelihood samples will have the same class
- Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches)

Information Gain, cont.

- Use to rank attributes and build DT (decision tree)!
- Choose nodes using attribute with **greatest gain**
  - → means least information remaining after split
  - I.e., subsets are all as skewed as possible
- Why?
  - Create small decision trees: predictions can be made with few attribute tests
  - Try to find a minimal process that still captures the data (Occam's Razor)

How Well Does it Work?

At least as accurate as human experts (sometimes)

- Diagnosing breast cancer: humans correct 65% of the time; decision tree classified 72% correct
- BP designed a decision tree for gas-oil separation for offshore oil platforms; replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example
- SKICAT (Sky Image Cataloging and Analysis Tool) used a DT to classify sky objects **an order of magnitude** fainter than was previously possible, with an accuracy of over 90%.

Converting Decision Trees to Rules

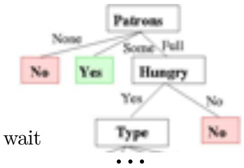
- 1 rule for each **path** in tree (from root to a leaf)
- Left-hand side: labels of nodes and arcs

Pa.=None → Don't wait

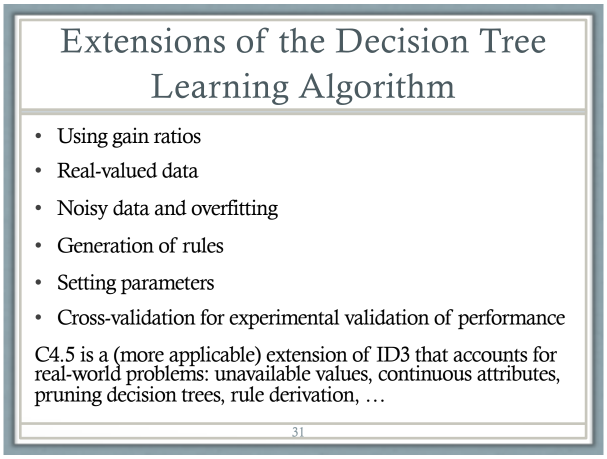
Pa.=Some → Wait

Pa.=F ∧ Hu.=No → Don't wait

etc...



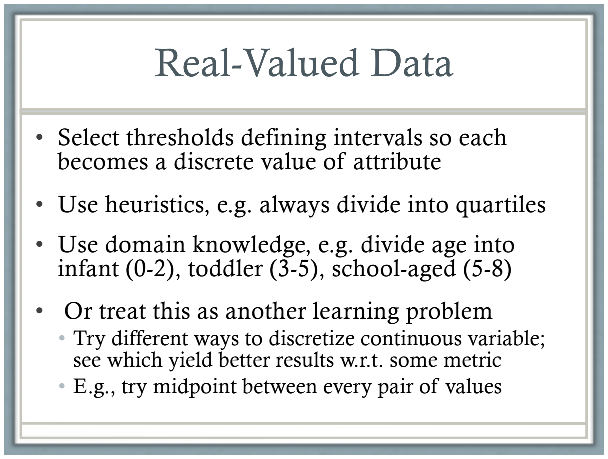
- Resulting rules can be simplified and reasoned over



Extensions of the Decision Tree Learning Algorithm

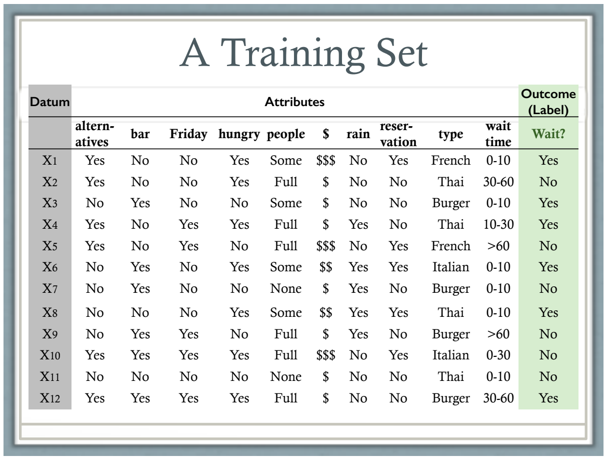
- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance

C4.5 is a (more applicable) extension of ID3 that accounts for real-world problems: unavailable values, continuous attributes, pruning decision trees, rule derivation, …



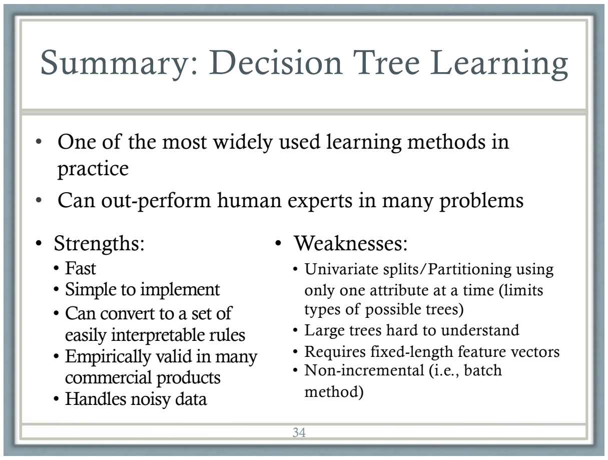
Real-Valued Data

- Select thresholds defining intervals so each becomes a discrete value of attribute
- Use heuristics, e.g. always divide into quartiles
- Use domain knowledge, e.g. 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 continuous variable; see which yield better results w.r.t. some metric
  - E.g., try midpoint between every pair of values



A Training Set

Datum	Attributes										Outcome (Label)
	altern-atives	bar	Friday	hungry	people	$	rain	reser-vation	type	wait time	Wait?
X_1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
X_2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
X_3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
X_4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10-30	Yes
X_5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
X_7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No
X_8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes
X_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	0-30	No
X_{11}	No	No	No	No	None	$	No	No	Thai	0-10	No
X_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes



Summary: Decision Tree Learning

- One of the most widely used learning methods in practice
- Can out-perform human experts in many problems

- Strengths:
  - Fast
  - Simple to implement
  - Can convert to a set of easily interpretable rules
  - Empirically valid in many commercial products
  - Handles noisy data

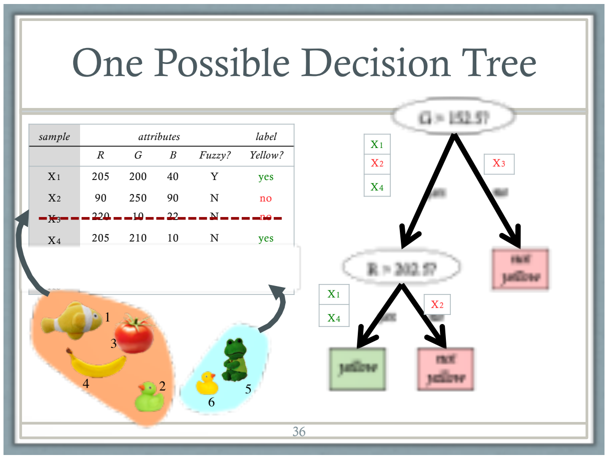
- Weaknesses:
  - Univariate splits/Partitioning using only one attribute at a time (limits types of possible trees)
  - Large trees hard to understand
  - Requires fixed-length feature vectors
  - Non-incremental (i.e., batch method)

ML II

I'VE GOT... CHEERIOS WITH A SHOT OF VERMOUTH.
AT LEAST IT'S BETTER THAN THE QUAIL EGGS IN WHIPPED CREAM AND MSG FROM LAST TIME.
ARE THESE SKITTLES BEEF-FRIED?
C'MON, GUYS, BE PATIENT. IN A FEW HUNDRED MORE MEALS, THE GENETIC ALGORITHM SHOULD CATCH UP TO EXISTING RECIPES AND START TO OPTIMIZE.

WE'VE DECIDED TO DROP THE CS DEPARTMENT FROM OUR WEEKLY DINNER PARTY HOSTING ROTATION.

PERMANENT LINK TO THIS COMIC: HTTPS://XKCD.COM/720/



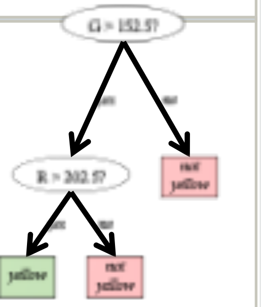
One Possible Decision Tree

sample	attributes				label
	R	*G*	*B*	*Fuzzy?*	*Yellow?*
X_1	205	200	40	Y	yes
X_2	90	250	90	N	no
~~X_3~~	~~220~~	~~10~~	~~22~~	~~N~~	~~no~~
X_4	205	210	10	N	yes

One Possible Decision Tree

- Predictions

	R	*G*	*B*	*Fuzzy?*	***Prediction: Is it yellow?***
X7	215	45	190	N	□ □



Overfitting

- Sometimes, model fits training data well but doesn't do well on test data
- Can be it "overfit" to the training data
  - Model is too **specific** to training data
  - Doesn't **generalize** to new information well

Examples (training data)	Attributes			Outcome
	Bipedal	Flies	Feathers	
Sparrow	Y	Y	Y	B
Monkey	Y	N	N	M
Ostrich	Y	N	Y	B
Bat	Y	Y	N	M
Elephant	N	N	N	M

- Learned model: (Y∧Y∧Y→B ∨ Y∧N∧N→M ∨ ...)

Overfitting 2

- Irrelevant attributes → overfitting
- If hypothesis space has many dimensions (many attributes), may find **meaningless regularity**
  - Ex: Name starts with [A-M] → Mammal

Examples (training data)	Attributes		Class
	Bi-pedal	Feath-ers	
Sparrow	Y	Y	B
Monkey	Y	N	M
Ostrich	Y	Y	B
Bat	Y	N	M
Elephant	N	N	M

Overfitting 3

- Incomplete training data → overfitting

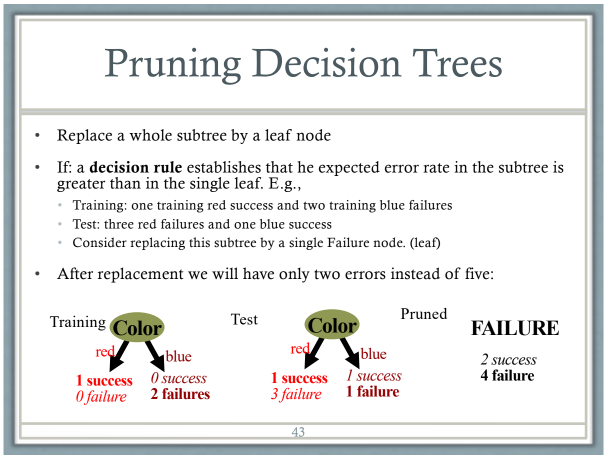
- Bad training/test split → overfitting

Overfitting

- Fix by…
  - Removing irrelevant features (e.g., remove 'first letter' from bird/mammal feature vector)
  - Getting more training data
  - Pruning low nodes in the decision tree (e.g., if improvement from best attribute at a node is below a threshold, stop and make this node a leaf rather than generating child nodes)
- Regularization
- Lots of other choices…

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
    - Errors in the data acquisition process, the preprocessing phase, //
  - 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
  - Some attributes are missing (are pangolins bipedal?)



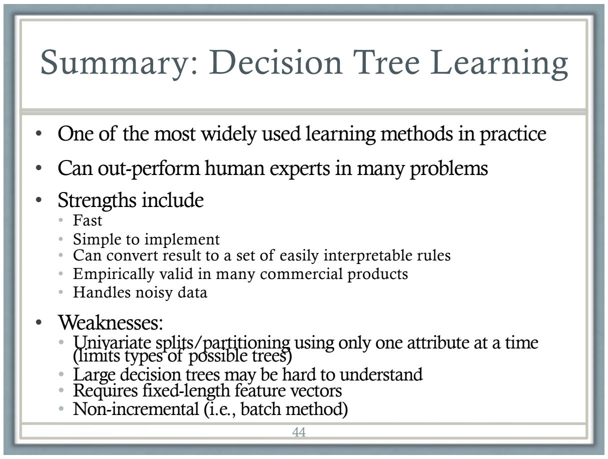
Pruning Decision Trees

- Replace a whole subtree by a leaf node
- If: a **decision rule** establishes that he 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. (leaf)
- After replacement we will have only two errors instead of five:

Training **Color**
red: **1 success** *0 failure*
blue: *0 success* **2 failures**

Test **Color**
red: **1 success** *3 failure*
blue: *1 success* **1 failure**

Pruned **FAILURE**
2 success
4 failure



Summary: Decision Tree Learning

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