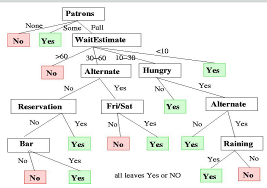
Machine Learning: Concepts & Decision Trees

(Ch. 18.1–18.3)



Today's Class

- Machine learning
  - What is ML?
  - Inductive learning
    - Supervised
    - Unsupervised
  - Decision trees
- Later: Bayesian learning, naïve Bayes, and BN learning

Questions?

- What's supervised learning?
  - What's classification? What's regression?
  - What's a hypothesis? What's a hypothesis space?
  - What are the training set and test set?
  - What is Ockham's razor?
- What's unsupervised learning?

What is Learning?

- "Learning denotes changes in a system that ... enable a system to do the same task more efficiently the next time." –Herbert Simon
- "Learning is constructing or modifying representations of what is being experienced." –Ryszard Michalski
- "Learning is making useful changes in our minds." –Marvin Minsky

Why Learn?

- Discover previously-unknown new things or structure
  - Data mining, scientific discovery
- Fill in skeletal or incomplete domain knowledge
- Build agents that can adapt to users or other agents
- Understand and improve efficiency of human learning
  - Use to improve methods for teaching and tutoring people (e.g., better computer-aided instruction)

Machine Leaning Successes

- Sentiment analysis
- Spam detection
- Machine translation
- Spoken language understanding
- Named entity detection
- Self driving cars
- Motion recognition (Microsoft X-Box)
- Identifying paces in digital images
- Recommender systems (Netflix, Amazon)
- Credit card fraud detection

Some Terminology

The Big Idea: given some data, you learn a model of how the world works that lets you predict new data.

- **Training Set:** Data from which you learn initially.
- **Model:** What you learn. A "model" of how inputs are associated with outputs.
- **Test set:** New data you test your model against.
- **Corpus:** A body of data. (pl.: corpora)
- **Representation:** The computational expression of data

8

Major Paradigms of ML (1)

- **Rote learning**: 1-1 mapping from inputs to stored representation, learning by memorization, association-based storage & retrieval
- **Induction:** Use specific examples to reach general conclusions
- **Clustering**: Unsupervised discovery of natural groups in data

10

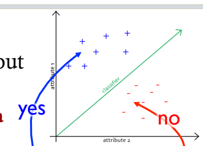
Major Paradigms of ML (2)

- **Analogy:** Find correspondences between different representations
- **Discovery**: Unsupervised, specific goal not given
- **Genetic algorithms:** *Evolutionary* search techniques, based on an analogy to *survival of the fittest*
- **Reinforcement:** Feedback (positive or negative reward) given at the end of a sequence of steps

11

The Classification Problem

- Extrapolate from **examples** to make accurate **predictions** about future data points
  - Examples are called **training data**

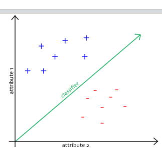


- Predict into **classes**, based on attributes ("**features**")
  - Example: it has tomato sauce, cheese, and no bread. Is it pizza?
  - Example: does this image contain a cat?

12

Supervised

- Goal: Learn an unknown function $f(X) = Y$, where
  - X is an input example
  - Y is the desired output. (f is the..?)
- **Supervised learning:** given a training set of (X, Y) pairs by a "teacher"

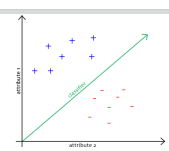


X			*Y*
bread	cheese	tomato sauce	**pizza**
¬ bread	¬ cheese	tomato sauce	**¬ not pizza**
bread	cheese	¬ tomato sauce	**pizza** (gross pizza)
lots more rows…			

"class labels" provided

Unsupervised

- Goal: Learn an unknown function $f(X) = Y$, where
  - X is an input example
  - Y is the desired output. (f is the..?)
- **Unsupervised learning:** only given Xs and some (eventual) feedback



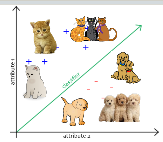
X		
bread	cheese	tomato sauce
¬ bread	¬ cheese	tomato sauce
bread	cheese	¬ tomato sauce
lots more rows…		

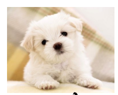
I think: pizza, ¬ pizza, ¬ pizza

67% right

Classification

- Classification or concept learning (aka "induction")
  - Given a set of examples of some concept/class/category:
    1. Determine if a given example is an instance of the concept (class member) or not
    2. If it **is**: **positive example**
    3. If it **is not**: **negative example**
    4. Or we can make a probabilistic prediction (e.g., using a Bayes net)

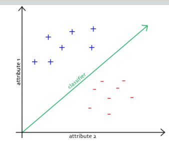




cat?

Supervised Concept Learning

- Given a training set of positive and negative examples of a concept
- Construct a description (model) that will accurately classify whether **future** examples are positive or negative



- I.e., learn estimate of function f given a training set:
  $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
  where each y_i is either + (positive) or - (negative), or a probability distribution over +/-

Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Supervised Learning

- Given training examples of inputs & outputs, produce "correct" outputs for new inputs
- Two main scenarios:
  - **Classification:** outputs typically labels (goodRisk/ badRisk, cat/notCat)
  - Learn a decision boundary that separates classes
  - **Regression** (aka "curve fitting" or "function approximation"): Learn a continuous input-output mapping from (possibly noisy) examples

Unsupervised Learning

Given only *unlabeled* data as input, learn some sort of structure, e.g.:

- Cluster your Facebook friends based on similarity of posts and friends
- Find sets of words whose meanings are related (e.g., doctor, hospital)
- Induce N topics and the words that are common in documents that are about that topic

Inductive Learning Framework

- Raw input data from sensors preprocessed to obtain **feature vector**, **X**, of **relevant** features for classifying examples
- Each **X** is a list of (attribute, value) pairs
- ***n*** attributes (a.k.a. features): fixed, positive, and finite
- Features have fixed, finite number # of possible values
  - Or continuous within some well-defined space, e.g., "age"
- Each example is a point in an *n*-dimensional feature space
  - X = [Person:Sue, EyeColor:Brown, Age:Young, Sex:Female]
  - X = [Cheese:*f*, Sauce:*t*, Bread:*t*]
  - X = [Texture:Fuzzy, Ears:Pointy, Purrs:Yes, Legs:4]

Inductive Learning as Search

- **Instance space, I,** is set of all possible examples
  - Defines the **language** for the training and test instances
  - Usually each instance $i \in I$ is a **feature vector**
  - Features are also sometimes called *attributes* or *variables*

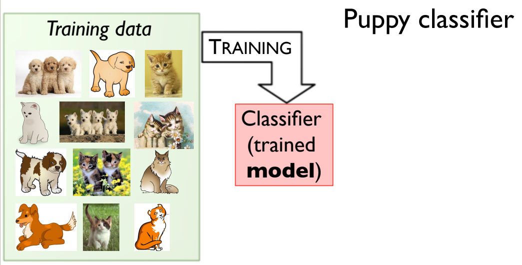
$$I: V_1 \times V_2 \times \ldots \times V_k,\ i = (v_1, v_2, \ldots, v_k)$$

- Class variable C gives an instance's class (to be predicted)

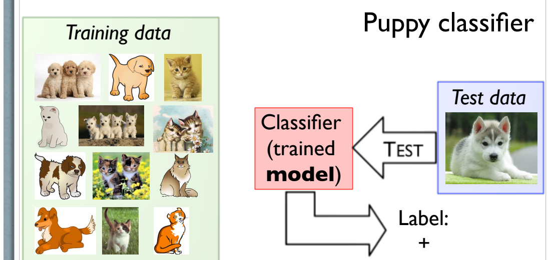
Inductive Learning as Search

- C gives an instance's class
- Model space M defines the possible **classifiers**
  - $M: I \to C$, $M = \{m_1, \ldots m_n\}$ (possibly infinite)
  - Model space is sometimes defined using same features as instance space (not always)
- Training data lets us search for a good (consistent, complete, simple) hypothesis in the model space
- The learned model is a *classifier*

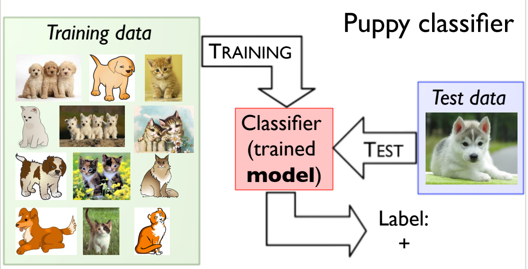
Inductive Learning Pipeline



Inductive Learning Pipeline



Inductive Learning Pipeline



Inductive Learning Pipeline

Training data, X

Texture	*Ears*	*Legs*	*Class*
Fuzzy	Round	4	+
Slimy	Missing	8	-
Fuzzy	Pointy	4	-
Fuzzy	Round	4	+
Fuzzy	Pointy	4	+
...			

TRAINING → Puppy classifier

Classifier (trained **model**) ← TEST ← *Test data* x_1 = <Fuzzy, Pointy, 4>

Label: +

Model Spaces (1)

- Decision trees
  - Partition the instance space **I** into axis-parallel regions
  - Labeled with class value
- Nearest-neighbor classifiers
  - Partition the instance space **I** into regions defined by centroid instances (or cluster of *k* instances)
- Bayesian networks
  - Probabilistic dependencies of class on attributes
  - Naïve Bayes: special case of BNs where class → each attribute

Model Spaces (2)

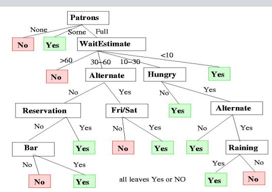
- Neural networks
  - Nonlinear feed-forward functions of attribute values
- Support vector machines
  - Find a separating plane in a high-dimensional feature space
- Associative rules (feature values → class)
- First-order logical rules

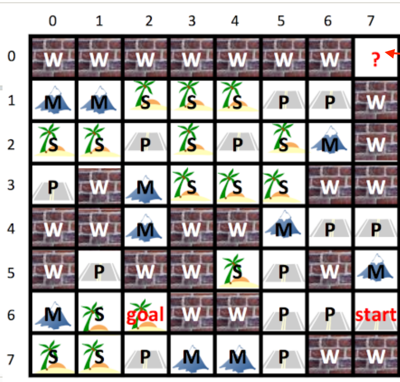
Decision Trees

- **Goal:** Build a 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 is an attribute (feature)
  - Each **leaf** node is a classification (+ or -)
    - Positive and negative data points
  - Each **arc** is one possible value of the attribute at the node from which the arc is directed
- Generalization: allow for >2 classes
  - e.g., {sell, hold, buy}

Decision Trees

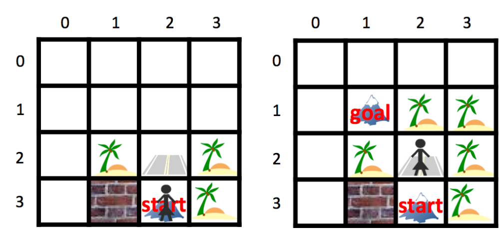
(Ch. 18.1–18.3)





Never observed; the agent can't see through walls

HW 4



timestep 1 *timestep 2*

Decision Trees (DTs)

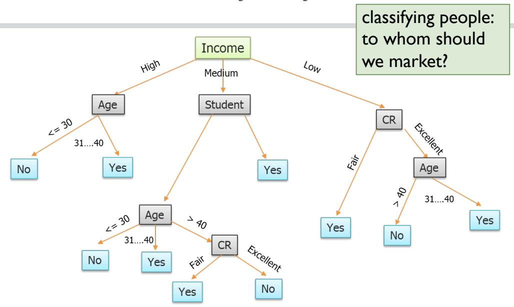
- A supervised learning method used for classification and regression
- Given a set of training tuples, learn model to predict one value from the others
  - Learned value typically a class (e.g. goodRisk)
- Resulting model is simple to understand, interpret, visualize and apply

Decision Tree Induction

- The Big Idea: build a tree of **decisions**, each of which splits training data into smaller groups
  - Very common machine learning technique!
- At each split, an attribute of the training data – a **feature** – is chosen to divide data into classes
- Goal: each leaf group in the tree consists entirely of one class
- Learning: creating that tree

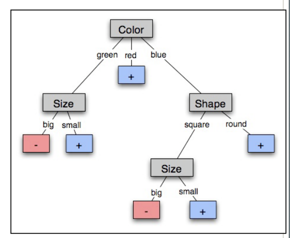
Will You Buy My Product?

classifying people: to whom should we market?

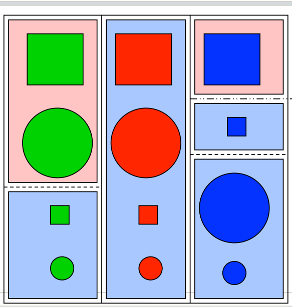


Learning Decision Trees

- Each **non-leaf** node is an attribute (feature)
- Each **arc** is one value of the attribute at the node it comes from
- Each **leaf** node is a classification (+ or -)



Learning a Concept



The red groups are negative examples, blue positive

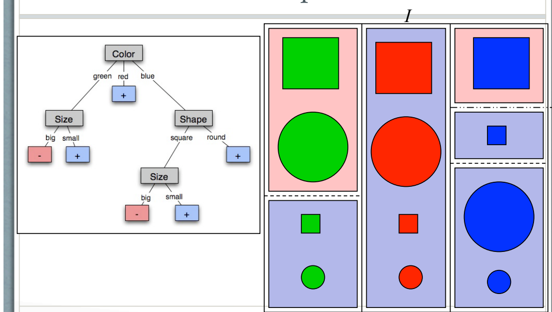
Features

- Size: large, small
- Color: red, green, blue
- Shape: square, circle

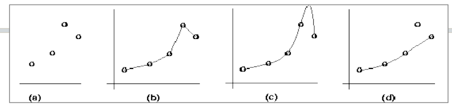
Training Data

Size	Color	Shape	class
Large	Green	Square	Negative
Large	Green	Circle	Negative
Small	Green	Square	Positive
Small	Green	Circle	positive
Large	Red	Square	Positive
Large	Red	Circle	Positive
Small	Red	Square	Positive
Small	Red	Circle	Positive
Large	Blue	Square	Negative
Small	Blue	Square	Positive
Large	Blue	Circle	Positive
Small	Blue	Circle	Positive

Decision Tree-Induced Partition – Example



Inductive Learning and Bias



- Want to learn a function **f(x) = y**
- Given sample (x,y) pairs, as in (a)
- There are several possible hypotheses (b-d)
- Preferring one shows the **bias** of our learning technique:
  - Prefer piece-wise functions? (b)
  - Prefer a smooth function? (c)
  - Prefer a simple function and treat outliers as noise? (d)

Preference Bias: Ockham's Razor

- A.k.a. Occam's Razor, Law of Economy, or Law of Parsimony
- Stated by William of Ockham (1285-1347/49):
  - *"Non sunt multiplicanda entia praeter necessitatem"*
  - "Entities are not to be multiplied beyond necessity"
- **"The simplest consistent explanation is the best."**
- Smallest decision tree that correctly classifies all training examples
- 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"

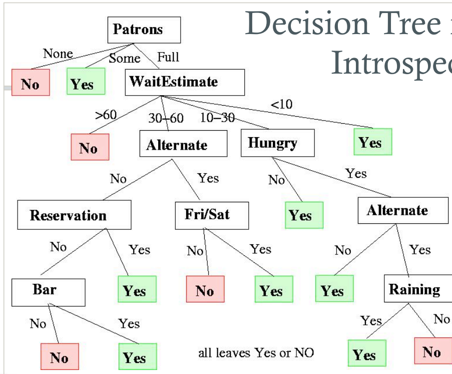
R&N's Restaurant Domain

- Model decision a patron makes when deciding whether to wait for a table
  - Two classes (outcomes): wait, leave
  - Ten attributes: Alternative available? ∃ Bar? Is it Friday? Hungry? How full is restaurant? How expensive? Is it raining? Do we have a reservation? What type of restaurant is it? What's purported waiting time?
- Training set of 12 examples
- ~ 7000 possible cases

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

Decision Tree from Introspection



Problem from R&N, table from Dr. Manfred Kerber @ Birmingham, with thanks – www.cs.bham.ac.uk/~mmk/Teaching/AI/I3.html

Issues

- It's like 20 questions:
- We can generate many decision trees depending on what attributes we ask about and in what order
- How do we decide?
- What makes one decision tree better than another: number of nodes? number of leaves? maximum depth?

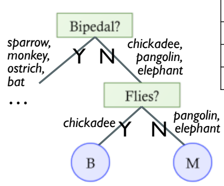
ID3/C4.5

- 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

Bird or Mammal?

1. Select attribute
2. Generate child nodes
3. Partition examples
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



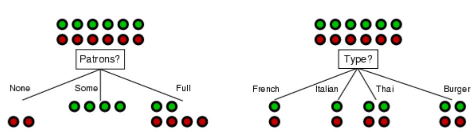
Test
mouse: <B:N, Fl:N, Fe:N>

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

Choosing an Attribute

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



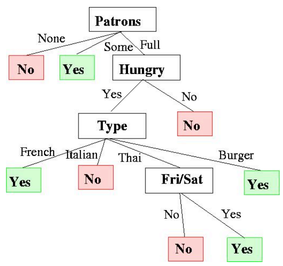
- Which is better: *Patrons?* or *Type?*
- **Why?**

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

ID3-induced Decision Tree



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

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:

  $P = (p_1, p_2 \ldots p_n)$
- 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.

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

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

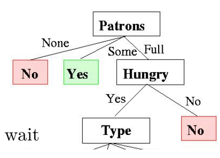
- 1 rule for each **path** in tree (from root to a leaf)
- Left-hand side: labels of nodes and arcs

  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

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)

70