CS 380: Artificial Intelligence

Lecture 12: Machine Learning

Some materials adapted from Russell & Norvig textbook slides: http://aima.cs.berkeley.edu

Summary so far:

- Rational Agents
- Problem Solving
  - Systematic Search
    - Uninformed
    - Informed
  - Local Search
  - Adversarial Search
- Logic and Knowledge Representation
  - Predicate Logic
  - First-order Logic
- Today: <u>Machine Learning</u>

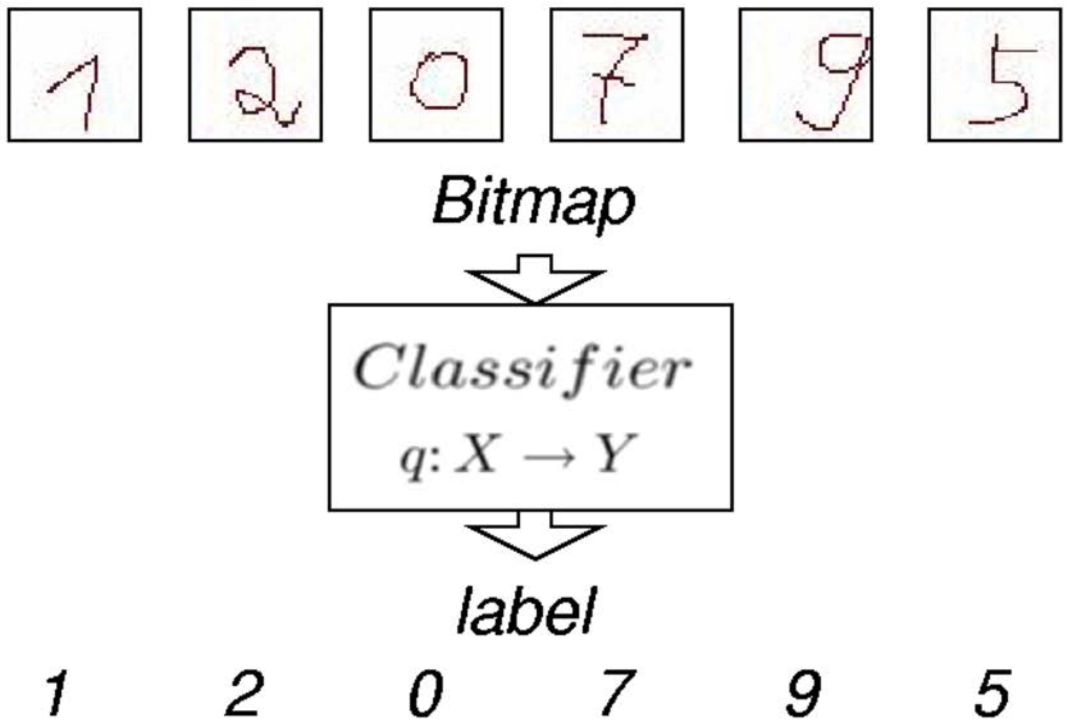
What is Learning?

Machine Learning

- Computational methods for computers to exhibit specific forms of learning. For example:
  - Learning from Examples:
    - Supervised learning
    - Unsupervised learning
  - Reinforcement Learning
  - Learning from Observation (demonstration/imitation)

Examples

- **Supervised Learning**: learning to recognize writing

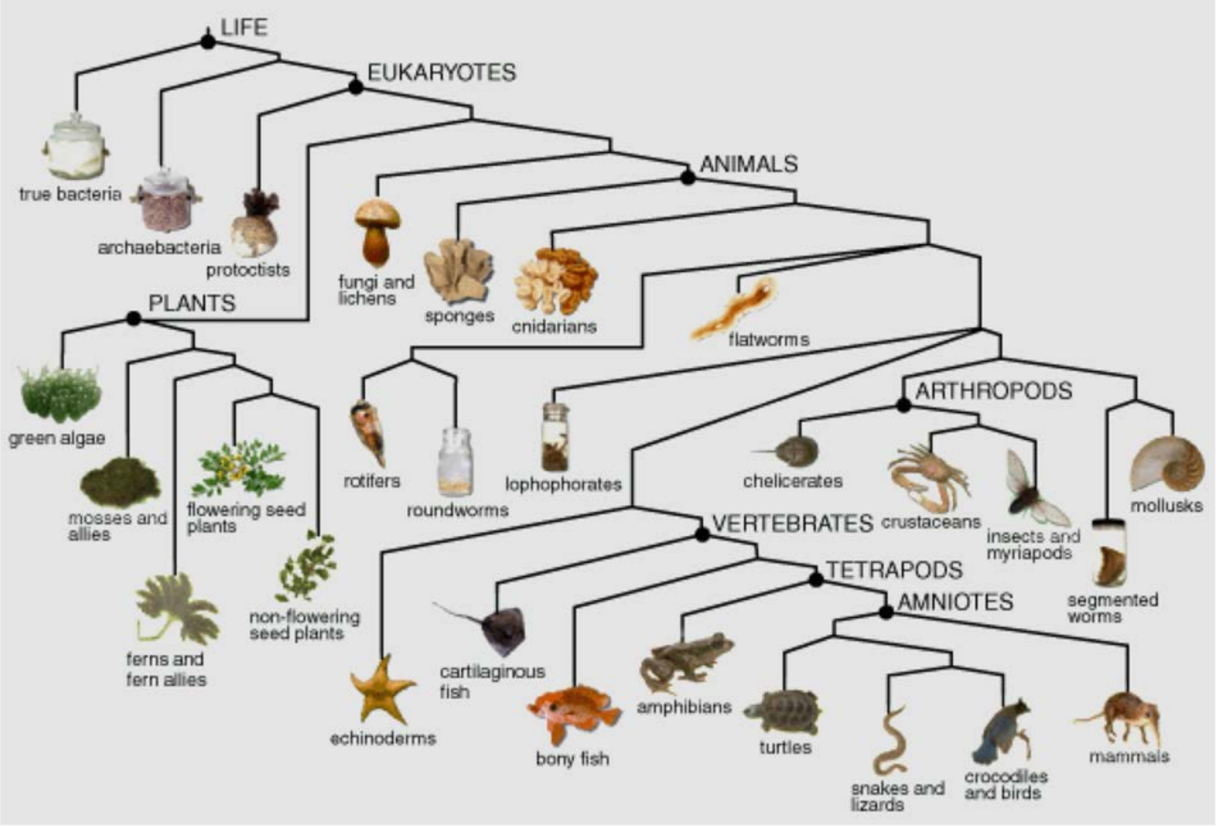


Examples

- **Supervised Learning**: image classification

Examples

- **Unsupervised Learning**: clustering observations into meaningful classes

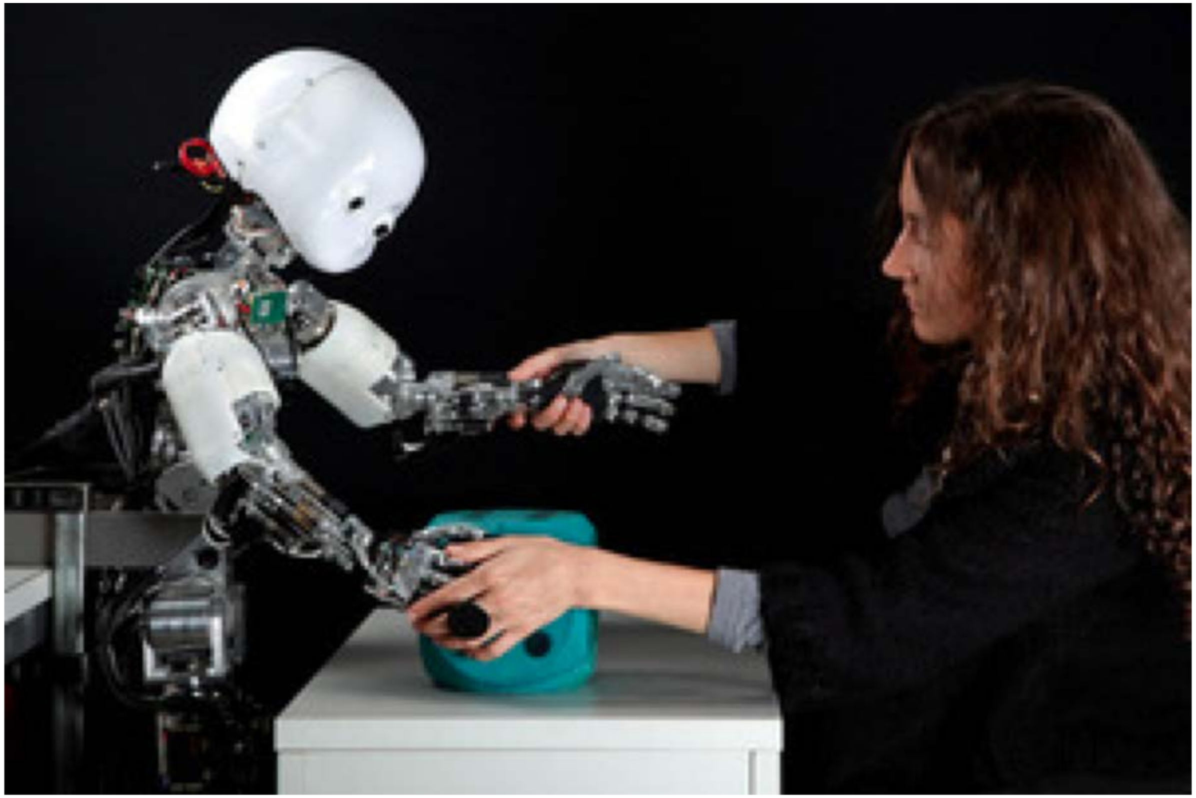


Examples

- **Reinforcement Learning**: learning to walk

Examples

- **Learning from demonstration**: performing tasks that other agents (or humans) can do



Examples

- **Reinforcement Learning**:
  - https://www.youtube.com/watch?v=hx_bgoTF7bs
  - https://www.youtube.com/watch?v=e27TUmMkOA0

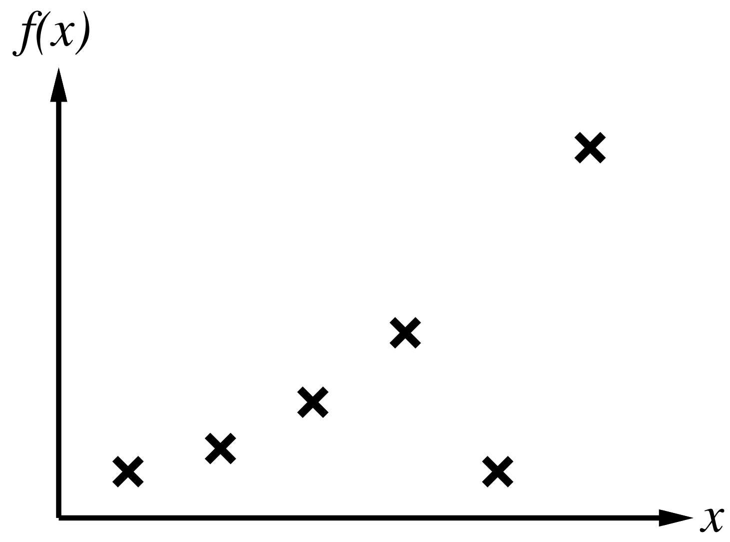
- **Learning from demonstration**:
  - https://www.youtube.com/watch?v=I9u3B6gv-RE

Inductive Learning

- Inductive learning tries to infer general concepts from simple examples
- Let's try to learn something in its simplest form: Learning a function from examples
  - **Desired**: f is a target function we want to learn — but we don't know what it is or how to compute it
  - **Given**: examples where an example = $< x , f(x) >$
  - **Problem**: find a hypothesis h such that $h \approx f$

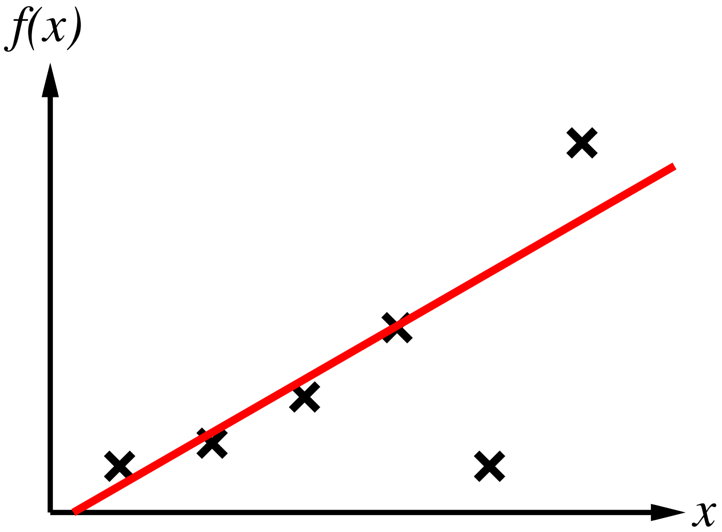
Inductive Learning

- Now, let's try to construct a hypothesis *h* such that $h \approx f$



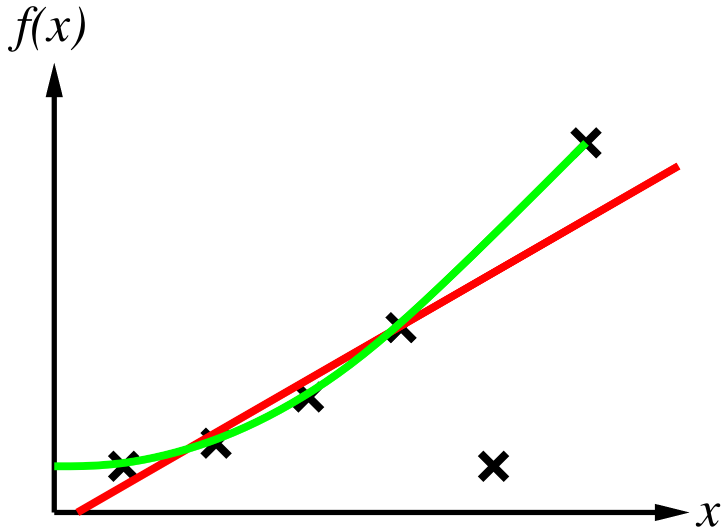
Inductive Learning

- Now, let's try to construct a hypothesis h such that $h \approx f$



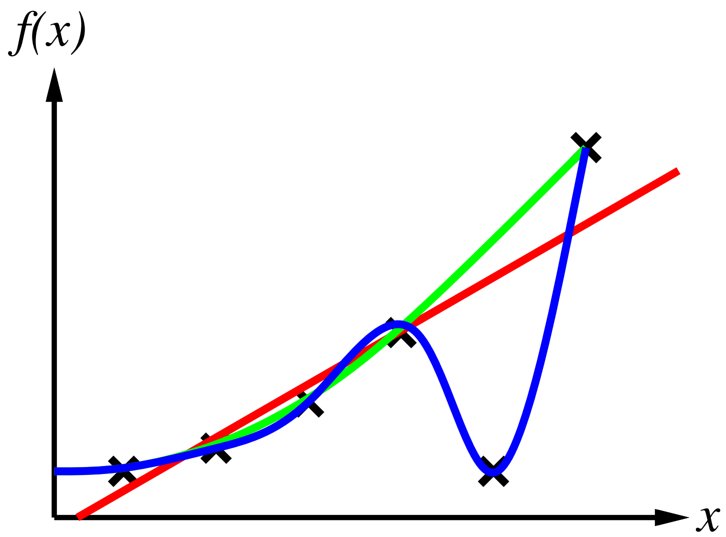
Inductive Learning

- Now, let's try to construct a hypothesis h such that $h \approx f$



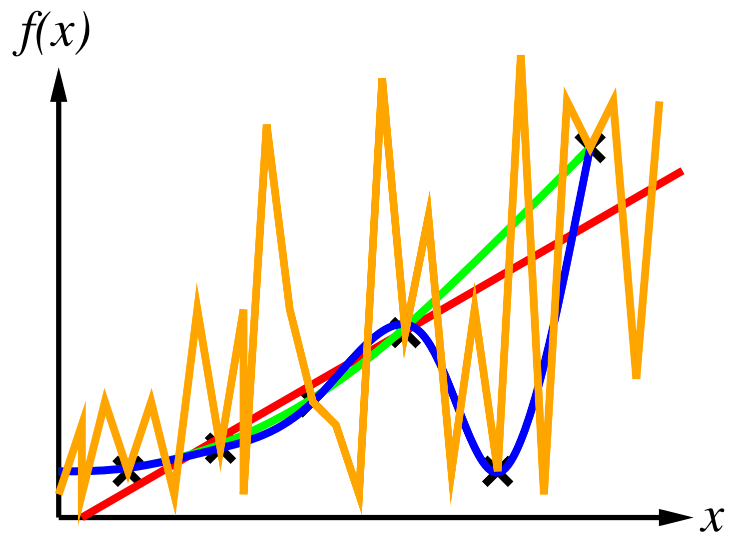
Inductive Learning

- Now, let's try to construct a hypothesis *h* such that $h \approx f$



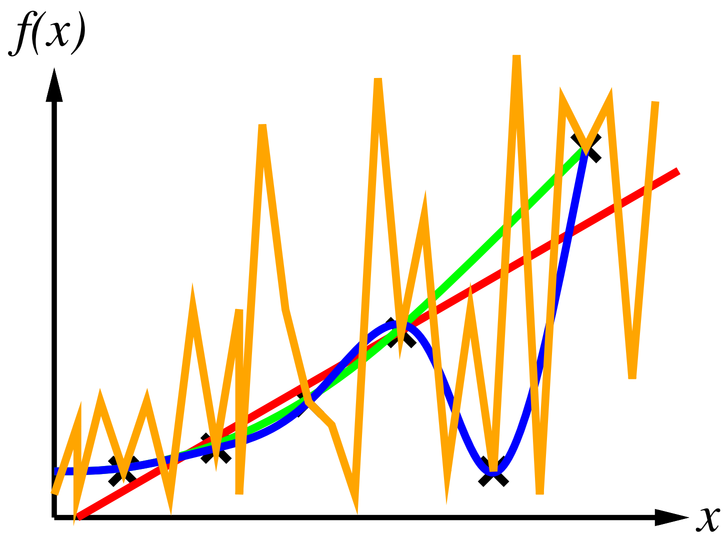
Inductive Learning

- Now, let's try to construct a hypothesis h such that $h \approx f$



Inductive Learning

- Now, let's try to construct a hypothesis h such that $h \approx f$



- **Occam's razor**: The simplest explanation tends to be the correct/best(?) one.

William of Occam (1287-1347)

Attribute-Based Representations

- Items (or examples) described by attribute values (Boolean, discrete, continuous, etc.)
- Example: Will I wait for a table at a restaurant?
  - What factors (attributes) would affect your decision?

Attribute-Based Representations

- Items (or examples) described by attribute values (Boolean, discrete, continuous, etc.)
- Example: Will I wait for a table at a restaurant?
  - Do I have an **Alternative**?
  - Is there a **Bar**?
  - Is it **Friday**?
  - Am I **Hungry**?
  - Are there **Patrons** in the restaurant?
  - What is the **Price**?
  - Is it **Raining**?
  - Do I have a **Reservation**?
  - What **Type** of food is it?
  - What's the **Estimated** wait time?

  → I will wait, **True** or **False**

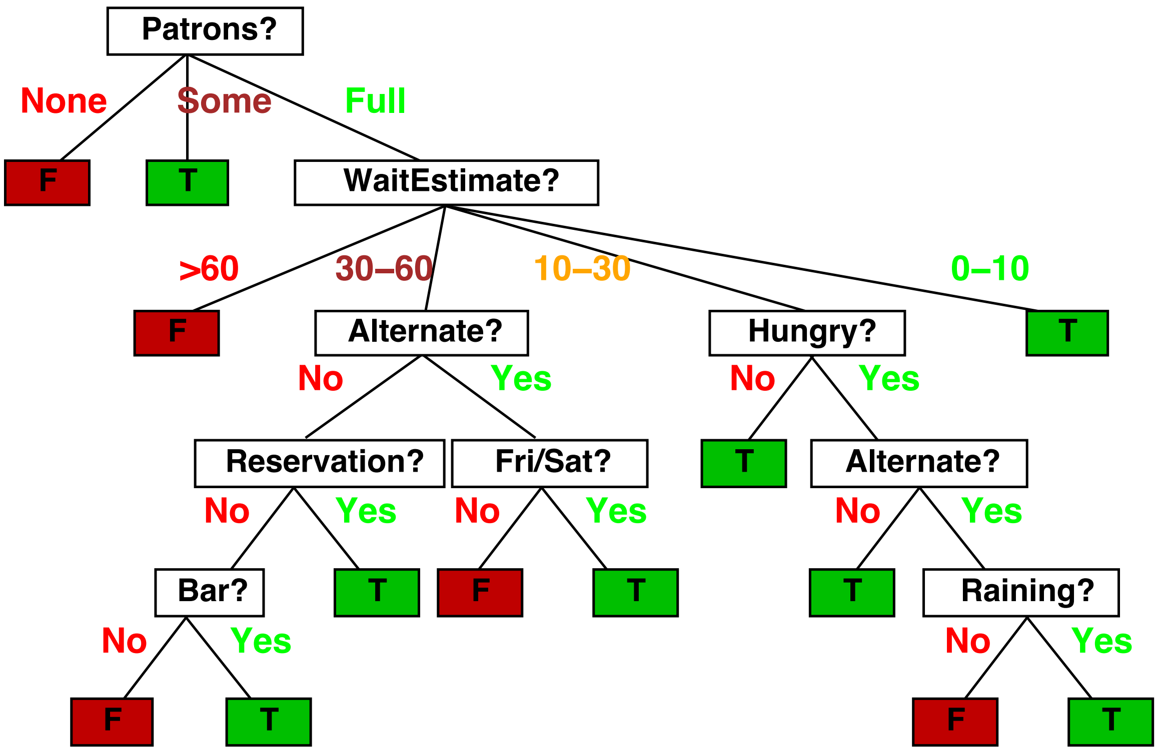
Attribute-Based Representations

- Items (or examples) described by attribute values (Boolean, discrete, continuous, etc.)
- Example: Will I wait for a table at a restaurant?

	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X1	T	F	F	T	Some	$$$	F	T	French	0–10	**T**
X2	T	F	F	T	Full	$	F	F	Thai	30–60	**F**
X3	F	T	F	F	Some	$	F	F	Burger	0–10	**T**
X4	T	F	T	T	Full	$	F	F	Thai	10–30	**T**
X5	T	F	T	F	Full	$$$	F	T	French	>60	**F**
X6	F	T	F	T	Some	$$	T	T	Italian	0–10	**T**
X7	F	T	F	F	None	$	T	F	Burger	0–10	**F**
X8	F	F	F	T	Some	$$	T	T	Thai	0–10	**T**
X9	F	T	T	F	Full	$	T	F	Burger	>60	**F**
X10	T	T	T	T	Full	$$$	F	T	Italian	10–30	**F**
X11	F	F	F	F	None	$	F	F	Thai	0–10	**F**
X12	T	T	T	T	Full	$	F	F	Burger	30–60	**T**

Decision Trees

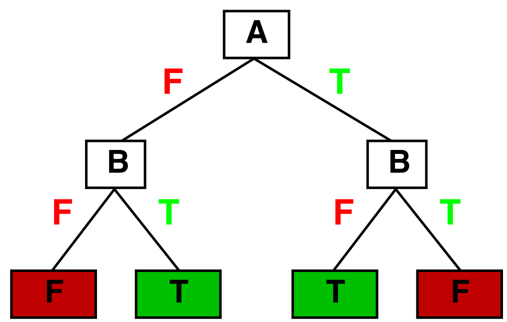
- A Decision Tree asks questions about the attributes of a given example to provide an answer
- Here is the “true” tree for whether or not to wait:



Decision Trees

- Decision trees are expressive: they can express any function of the input attributes
- For example, for Boolean functions, we can convert *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 w/ one path to leaf for each example (unless *f* nondeterministic in *x*)
- But it probably won't generalize to new examples!

Decision Trees

- How many distinct decision trees are there with n Boolean attributes?
  = number of Boolean functions
  = number of distinct truth tables with 2^n rows = $2^{2^\wedge n}$
  - E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees
- How many purely conjunctive hypotheses are there? (e.g., Hungry ∧ ¬Rain)
  - Each attribute can be positive, negative, or omitted → 3^n distinct conjunctive hypotheses
- A more expressive hypothesis space…
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent w/ training set
  - But may get worse predictions!

Decision-Tree Learning

- **Goal**: find a small tree consistent with examples
- **Idea**: (recursively) choose the "most significant" attribute as the root of a (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
        $best \leftarrow$ CHOOSE-ATTRIBUTE($attributes, examples$)
        $tree \leftarrow$ a new decision tree with root test $best$
        for each value v_i of $best$ **do**
            $examples_i \leftarrow$ {elements of $examples$ with $best = v_i$}
            $subtree \leftarrow$ DTL($examples_i, attributes - best$, MODE($examples$))
            add a branch to $tree$ with label v_i and subtree $subtree$
        return $tree$

ID3

- Developed by Quinlan (1986)
- ID3 = “Iterative Dichotomiser 3”
  - Create a decision tree that iteratively evaluates an item based on its attributes
  - Key idea: use *entropy* (or *information gain*) to make these decisions

ID3

- Consider how different attributes might split a set of training examples
  - Based on **Patrons**...

	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X1	T	F	F	T	Some	$$$	F	T	French	0–10	**T**
X2	T	F	F	T	Full	$	F	F	Thai	30–60	**F**
X3	F	T	F	F	Some	$	F	F	Burger	0–10	**T**
X4	T	F	T	T	Full	$	F	F	Thai	10–30	**T**
X5	T	F	T	F	Full	$$$	F	T	French	>60	**F**
X6	F	T	F	T	Some	$$	T	T	Italian	0–10	**T**
X7	F	T	F	F	None	$	T	F	Burger	0–10	**F**
X8	F	F	F	T	Some	$$	T	T	Thai	0–10	**T**
X9	F	T	T	F	Full	$	T	F	Burger	>60	**F**
X10	T	T	T	T	Full	$$$	F	T	Italian	10–30	**F**
X11	F	F	F	F	None	$	F	F	Thai	0–10	**F**
X12	T	T	T	T	Full	$	F	F	Burger	30–60	**T**

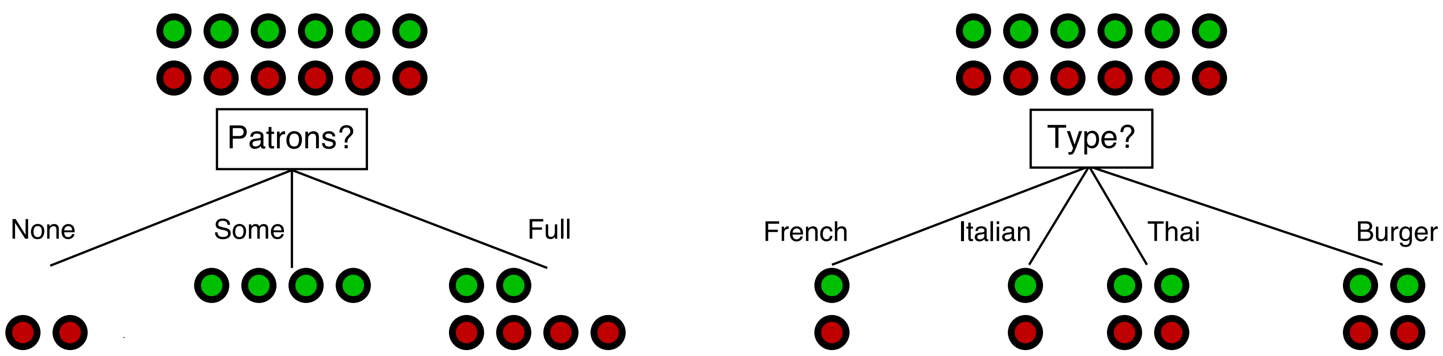
ID3

- Consider how different attributes might split a set of training examples
  - Based on **Type**...

	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X1	T	F	F	T	Some	$$$	F	T	French	0–10	**T**
X2	T	F	F	T	Full	$	F	F	Thai	30–60	**F**
X3	F	T	F	F	Some	$	F	F	Burger	0–10	**T**
X4	T	F	T	T	Full	$	F	F	Thai	10–30	**T**
X5	T	F	T	F	Full	$$$	F	T	French	>60	**F**
X6	F	T	F	T	Some	$$	T	T	Italian	0–10	**T**
X7	F	T	F	F	None	$	T	F	Burger	0–10	**F**
X8	F	F	F	T	Some	$$	T	T	Thai	0–10	**T**
X9	F	T	T	F	Full	$	T	F	Burger	>60	**F**
X10	T	T	T	T	Full	$$$	F	T	Italian	10–30	**F**
X11	F	F	F	F	None	$	F	F	Thai	0–10	**F**
X12	T	T	T	T	Full	$	F	F	Burger	30–60	**T**

ID3

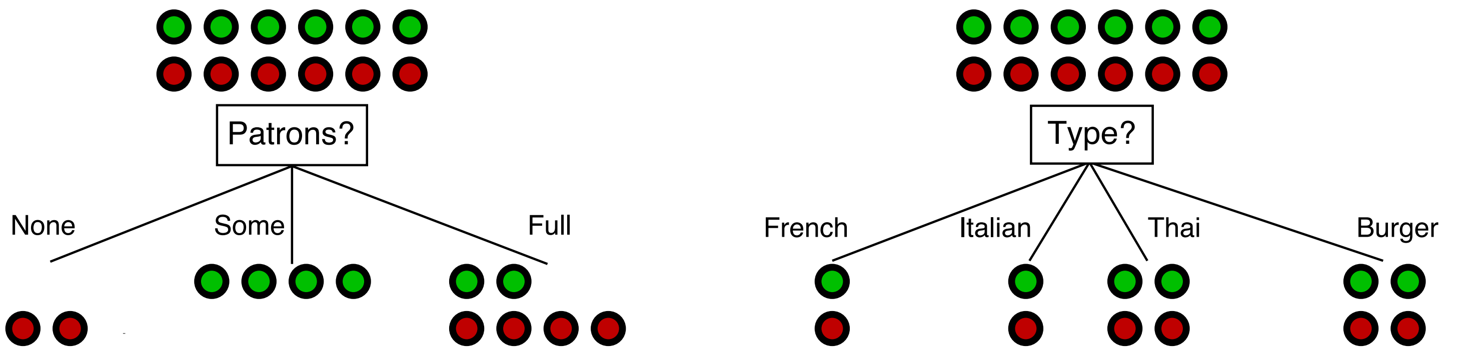
- Consider how different attributes might split a set of training examples



- Which of these attributes should we prefer?

ID3

- Consider how different attributes might split a set of training examples



- Which of these attributes should we prefer?
  – Better to have subsets that are all/mostly the same class
  – "**Patrons**" is a better choice here

Entropy

- "Information" is what answers questions in the context of decision trees
  - In essence, 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 $\langle 0.5, 0.5 \rangle$
  - Information in an answer when prior is $\langle P_1, \dots, P_n \rangle$ is

$$H(\langle P_1, \dots, P_n \rangle) = \sum_{i=1}^{n} -P_i \log_2 P_i$$

(also called the **entropy** of the prior)

Entropy

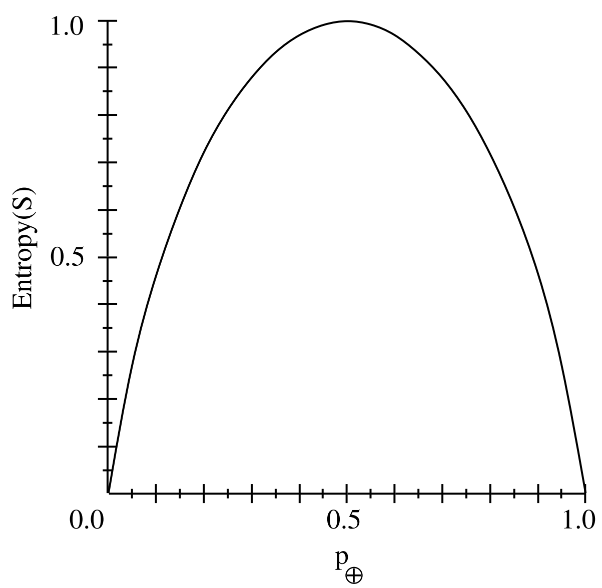
- Given a set of symbols *S*…
- Drawn from an alphabet *B*…
- **Entropy** = expected number of bits needed to encode the next symbol (amount of information that knowing one more symbol provides us)
  - If *S* is drawn at random from *B*, entropy is maximal
  - If *S* is always the same symbol from *B*, entropy is minimal
    - we know which symbol will come next, so there's no new information)

Entropy

- Entropy can be defined as:

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} -P_i \log_2 P_i$$

- Example for a binary variable:



ID3

- Suppose we have *p* positive and *n* negative examples at the root
  - $H(\left\langle \frac{p}{p+n}, \frac{n}{p+n} \right\rangle)$ bits needed to classify a new example
  - E.g., for the restaurant examples, *p* = *n* = 6, so 1 bit

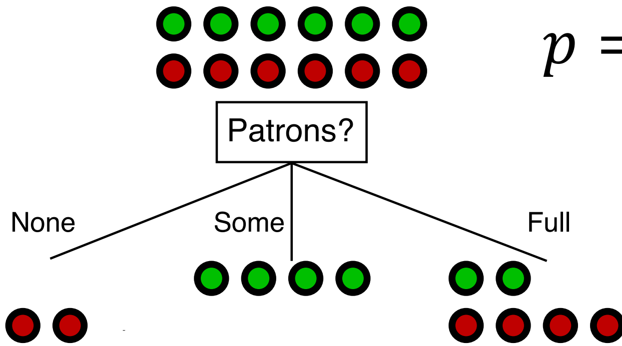
- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

ID3

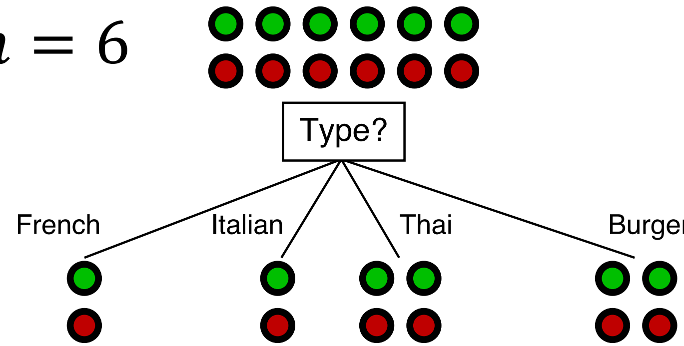
- Let E_i have p_i positive and n_i negative examples
  - $H(\langle\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\rangle)$ bits needed to classify a new example
  - the **expected** number of bits per example over all branches is

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

ID3



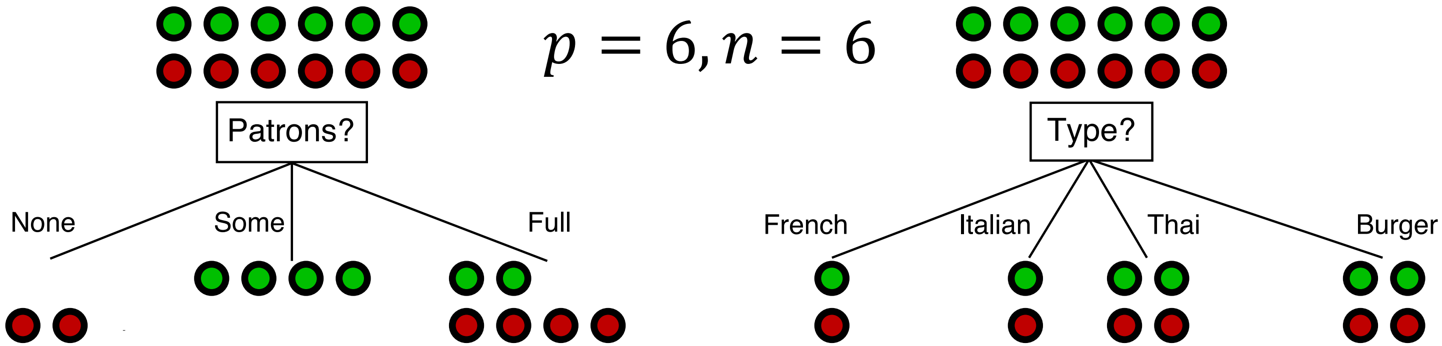
$p = 6, n = 6$



- For **Patrons**:
  - Subset E_1 has p_1 = 0 positive, n_1 = 2 negative
  - Subset E_2 has p_2 = 4 positive, n_2 = 0 negative
  - Subset E_3 has p_3 = 2 positive, n_3 = 4 negative

$$\sum_i \frac{p_i + n_i}{p + n} H\left(\left\langle \frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i} \right\rangle\right) = 0.459 \text{ bits}$$

ID3



$p = 6, n = 6$

- For **Type**:
  - All subsets have $p_i = n_i$

$$\sum_i \frac{p_i + n_i}{p + n} H\left(\left\langle \frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i} \right\rangle\right) = 1 \text{ bit}$$

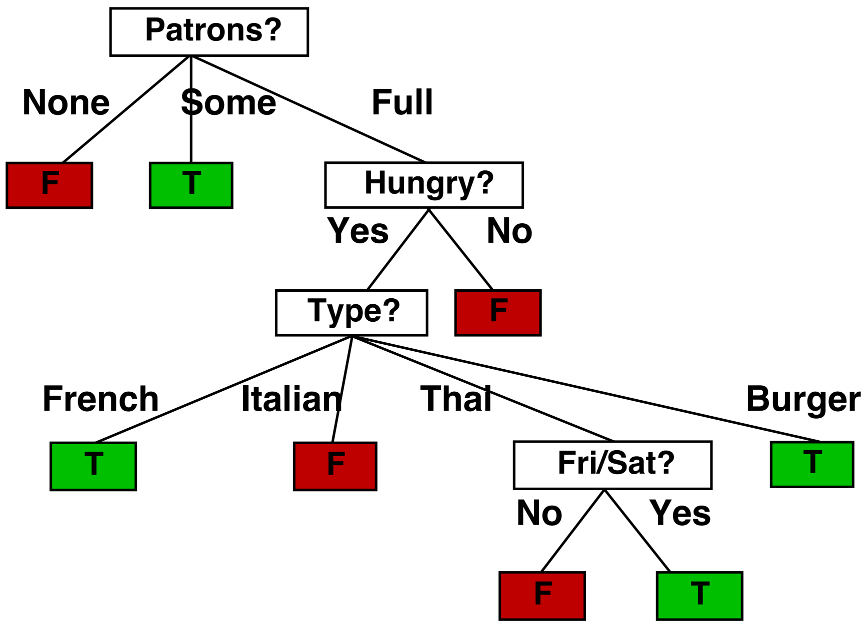
- So we choose the attribute that minimizes the remaining information needed, i.e., **Patrons**

ID3

- Alternatives to Entropy…
  - E.g., Information Gain is a common alternative
  - And there are others (Gain Ratio, GINI Index, RLDM, etc.)
- Alternative search strategies
  - E.g., Decision Forests (instead of 1 deep tree, use multiple shorter trees and aggregate their decisions)

ID3

- The ID3 tree after learning:



- A lot simpler than the “true” tree we saw earlier
- Why is simpler a good thing here?

Overfitting

- What if almost all examples fit a clear pattern, but there's one that doesn't fit?

	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X1	T	F	F	T	Some	$$$	F	T	French	0–10	**F**
X2	T	F	F	T	Full	$	F	F	Thai	30–60	**F**
X3	F	T	F	F	Some	$	F	F	Burger	0–10	**F**
X4	T	F	T	T	Full	$	F	F	Thai	10–30	**F**
X5	T	F	T	F	Full	$$$	F	T	French	>60	**F**
X6	F	T	F	T	Some	$$	T	T	Italian	0–10	**T**
X7	F	T	F	F	None	$	T	F	Burger	0–10	**T**
X8	F	F	F	T	Some	$$	T	T	Thai	0–10	**T**
X9	F	T	T	F	Full	$	T	F	Burger	>60	**T**
X10	T	T	T	T	Full	$$$	F	T	Italian	10–30	**F**
X11	F	F	F	F	None	$	F	F	Thai	0–10	**F**
X12	T	T	T	T	Full	$	F	F	Burger	30–60	**F**

Overfitting

- What if almost all examples fit a clear pattern, but there's one that doesn't fit?

	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X1	T	F	F	T	Some	$$$	F	T	French	0–10	**F**
X2	T	F	F	T	Full	$	F	F	Thai	30–60	**F**
X3	F	T	F	F	Some	$	F	F	Burger	0–10	**T**
X4	T	F	T	T	Full	$	F	F	Thai	10–30	**F**
X5	T	F	T	F	Full	$$$	F	T	French	>60	**F**
X6	F	T	F	T	Some	$$	T	T	Italian	0–10	**T**
X7	F	T	F	F	None	$	T	F	Burger	0–10	**T**
X8	F	F	F	T	Some	$$	T	T	Thai	0–10	**T**
X9	F	T	T	F	Full	$	T	F	Burger	>60	**T**
X10	T	T	T	T	Full	$$$	F	T	Italian	10–30	**F**
X11	F	F	F	F	None	$	F	F	Thai	0–10	**F**
X12	T	T	T	T	Full	$	F	F	Burger	30–60	**F**

Overfitting

- What happens if we have an example in the training set that is noise? (e.g. that is mislabeled)
  - ID3 will try to force it into the decision tree!
- This is an example of **Overfitting** — when what you've learned does very well on training examples, but much less well on other examples
  - In other words, it doesn't generalize
  - One of the most critical concepts in machine learning!
- For ID3, some strategies exist to avoid this
  - e.g., prevent leaves that have very few examples
  - e.g., stop building the tree when there would be little information gain in the next level

Overfitting

- More generally, what can you do?
  - Use a **training set** and a **test set**
    - Say, reserve 90% for training and 10% for testing
  - Use cross-validation
    - Repeatedly split your set into different training vs. test sets
    - E.g., Leave-one-out cross-validation
  - By not training on the data you're testing, you're trying to ensure generality

Next time…

- We will look at one particular type of Machine Learning called Reinforcement Learning…