CSCI 561
Foundation for Artificial Intelligence

17. Learning from Examples

Professor Wei-Min Shen
University of Southern California

Outline: learning from examples

- Learning agents
- Inductive learning
- Classification and support vector machines (SVM)
- Decision tree 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 study learning?

- Understand and improve efficiency of human learning
  - Use to improve methods for teaching and tutoring people (e.g., better computer-aided instruction)
- Discover new things or structure previously unknown
  - Examples: data mining, scientific discovery
- Fill in skeletal or incomplete specifications about a domain
  - Large, complex AI systems can’t be completely built by hand and require dynamic updating to incorporate new information
  - Learning new characteristics expands the domain or expertise and lessens the “brittleness” of the system
- Build agents that can adapt to users, other agents, and their environment

Two types of learning in AI

Deductive: Deduce rules/facts from already known rules/facts. (We have already dealt with this)

$$(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow C)$$

Inductive: Learn <u>new</u> rules/facts from a data set D.

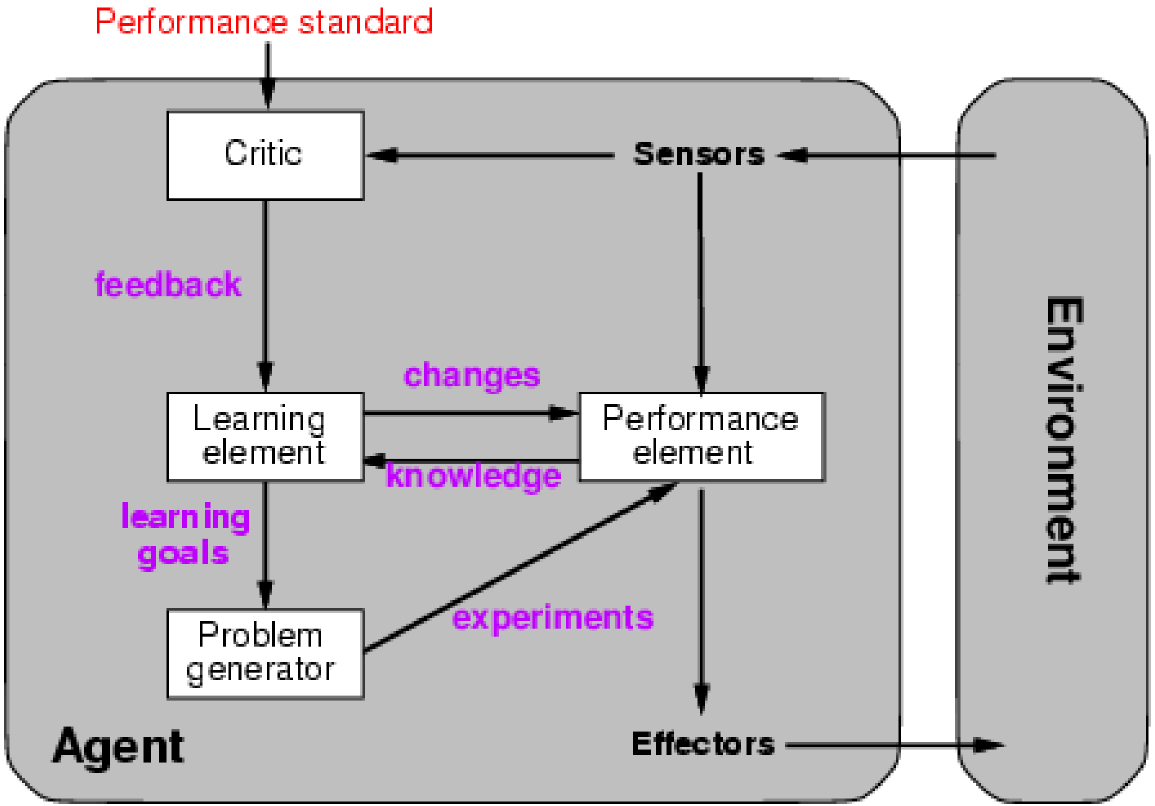
$$\mathcal{D} = \{\mathbf{x}(n), y(n)\}_{n=1\ldots N} \Rightarrow (A \Rightarrow C)$$

We will be dealing with the latter, *inductive* learning, now

Learning (Essential Features)

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

Learning agents



Learning element

- Design of a learning element is affected by
  - Which components of the performance element are to be learned
  - What feedback is available to learn these components
  - What representation is used for the components

- Type of feedback:
  - Supervised learning: correct answers for each example
  - Unsupervised learning: correct answers not given
  - Reinforcement learning: occasional rewards

Inductive learning

- Simplest form: learn a function from examples

f is the target function

An example is a pair (x, $f(x)$)

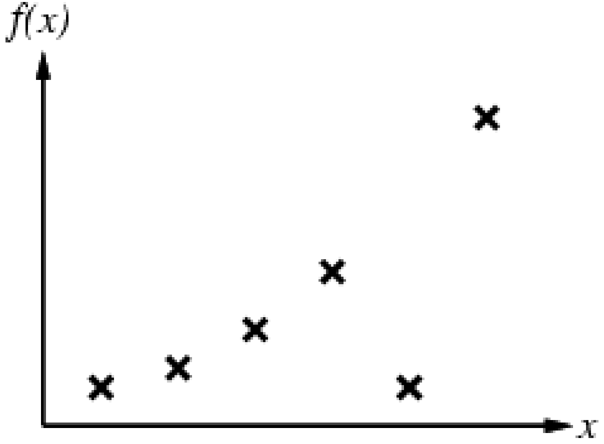
Problem: find a hypothesis h from a space H of possible functions
such that $h \approx f$
given a training set of examples

(This is a highly simplified model of real learning:
- Ignores prior knowledge
- Assumes examples are given)

Inductive learning method

- Construct/adjust h to agree with f on the training set of examples
- (h is consistent if it agrees with f on all the given examples)

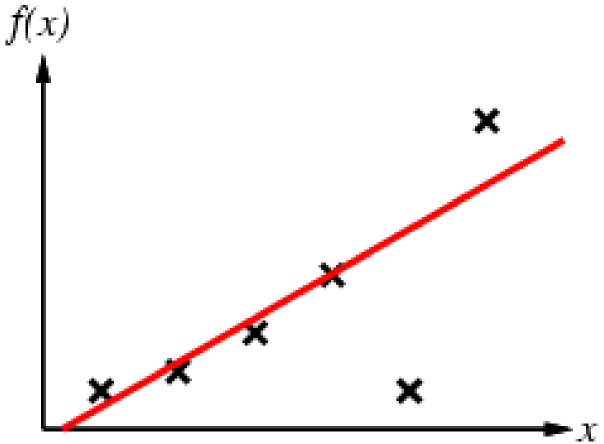
- E.g., curve fitting:



Inductive learning method

- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)

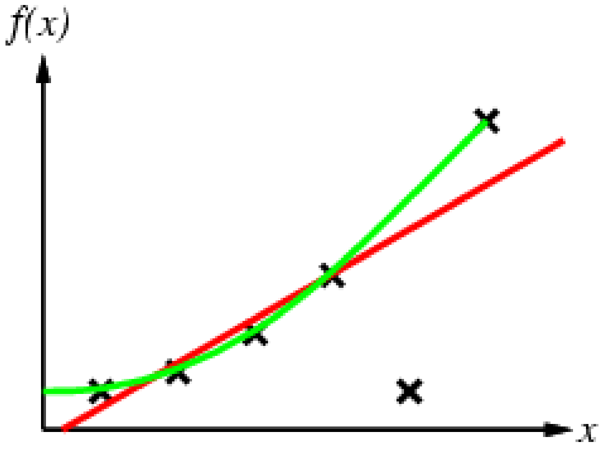
- E.g., curve fitting:



Inductive learning method

- Construct/adjust *h* to agree with f on the training set
- (*h* is consistent if it agrees with f on all the examples)

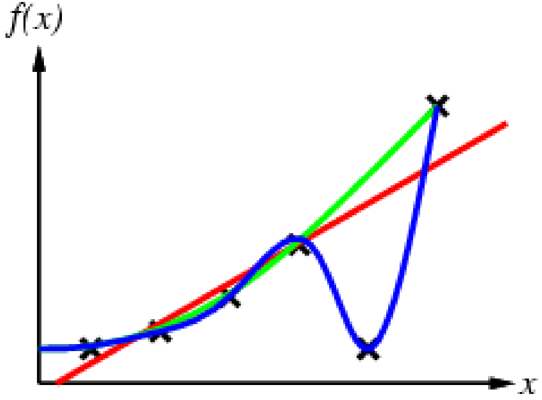
- E.g., curve fitting:



Inductive learning method

- Construct/adjust *h* to agree with f on the training set
- (*h* is consistent if it agrees with f on all the examples)

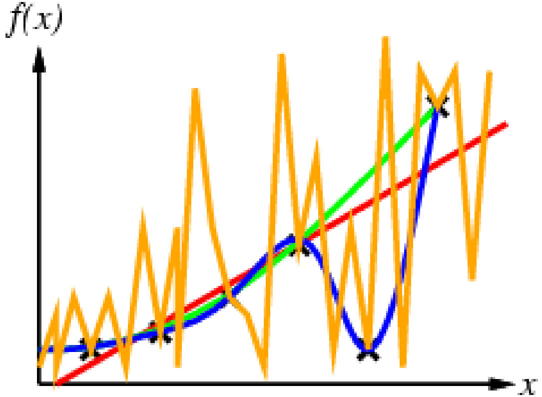
- E.g., curve fitting:



Inductive learning method

- Construct/adjust h to agree with f on the training set
- (h is consistent if it agrees with f on all the examples)

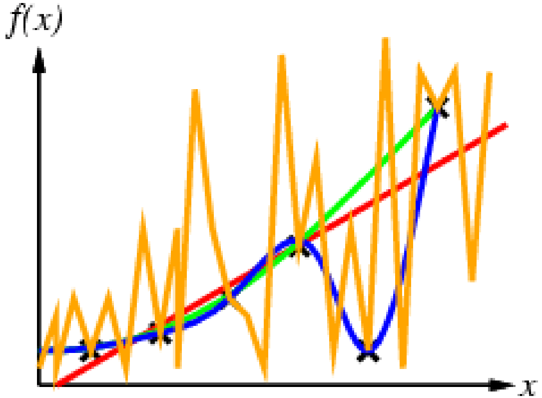
- E.g., curve fitting:



Inductive learning method

- Construct/adjust *h* to agree with f on the training set
- (*h* is consistent if it agrees with f on all the examples)

- E.g., curve fitting:



- Ockham's Razor: prefer the simplest hypothesis consistent with the data

Learning to classify

- In many problems we want to learn how to classify data into one of several possible categories.
  - E.g., face recognition, etc. Here are earthquake vs nuclear explosion:

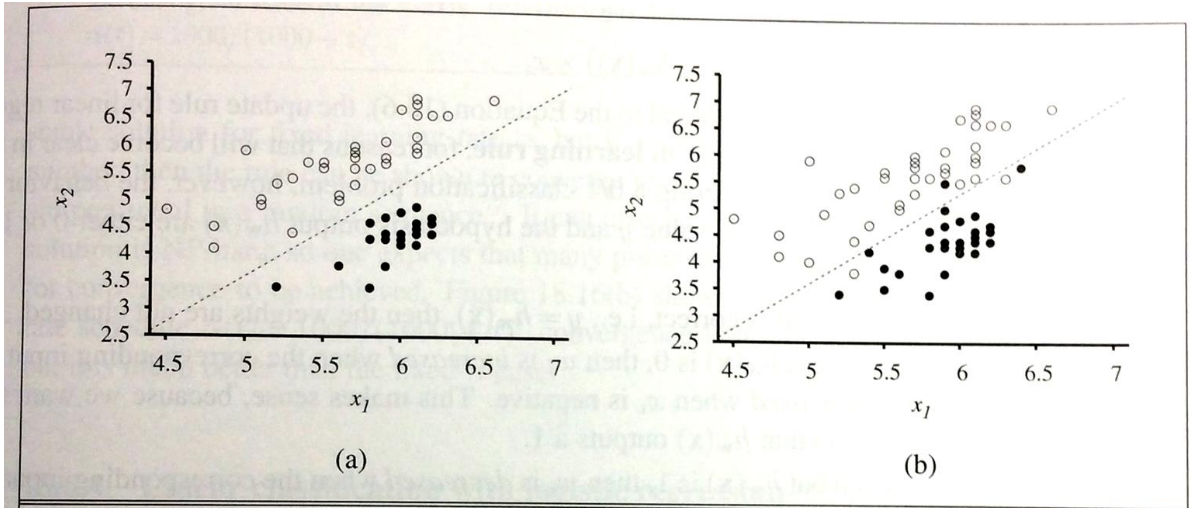


Figure 18.15 (a) Plot of two seismic data parameters, body wave magnitude x_1 and surface wave magnitude x_2, for earthquakes (white circles) and nuclear explosions (black circles) occurring between 1982 and 1990 in Asia and the Middle East (Kebeasy *et al.*, 1998). Also shown is a decision boundary between the classes. (b) The same domain with more data points. The earthquakes and explosions are no longer linearly separable.

Problem: how to best draw the line?

- Many methods exist. One of the most popular ones is the support vector machine (SVM): Find the maximum margin separator, i.e., the one that is as far as possible from any example point.

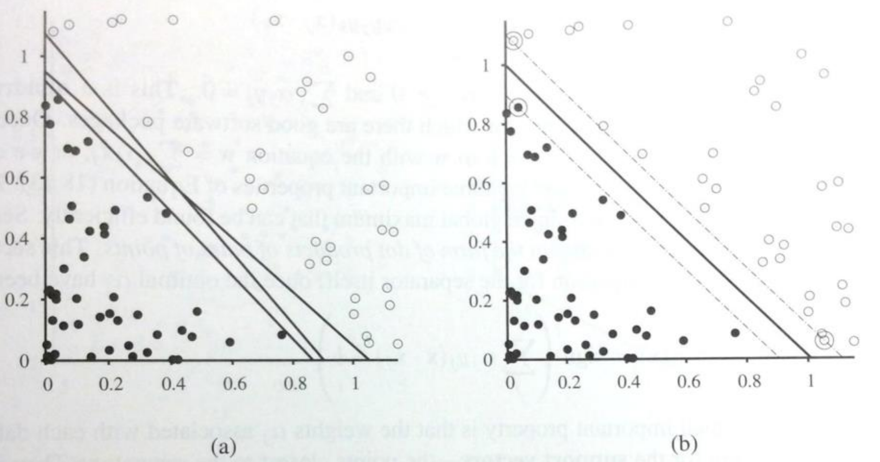


Figure 18.30 Support vector machine classification: (a) Two classes of points (black and white circles) and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the **margin** (area between dashed lines). The **support vectors** (points with large circles) are the examples closest to the separator.

Non-linear separability and SVM

- SVM can handle data that is not linearly separable using the so-called “kernel trick”: embed the data into a higher-dimensional space, in which it is linearly separable.

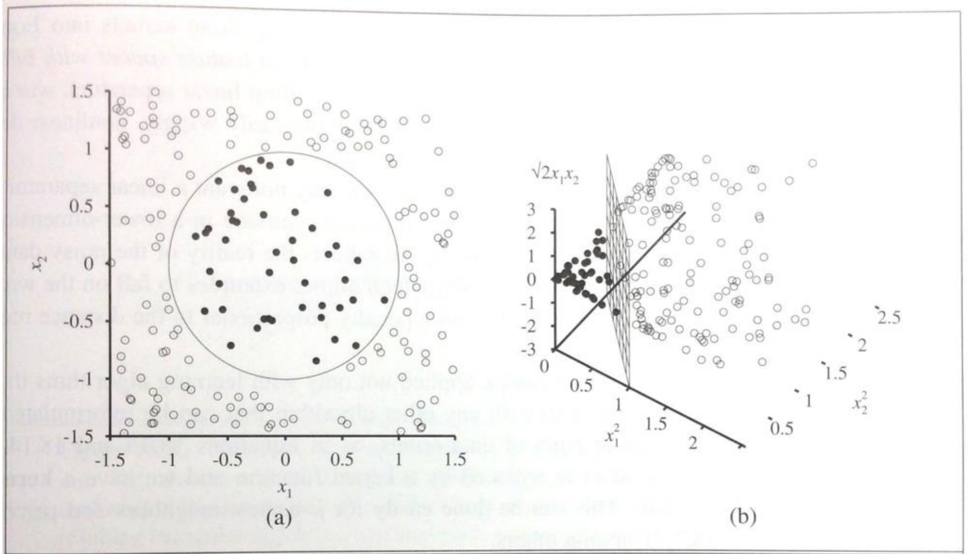


Figure 18.31 (a) A two-dimensional training set with positive examples as black circles and negative examples as white circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions. Figure 18.30(b) gives a closeup of the separator in (b).

Non-linear separability and SVM

- Kernel: remaps from the original 2 dimensions x1 and x2 to a new 3 dimensions: f1 = x_1^2, f2 = x_2^2, f3 = $\sqrt{2}\,x_1 x_2$

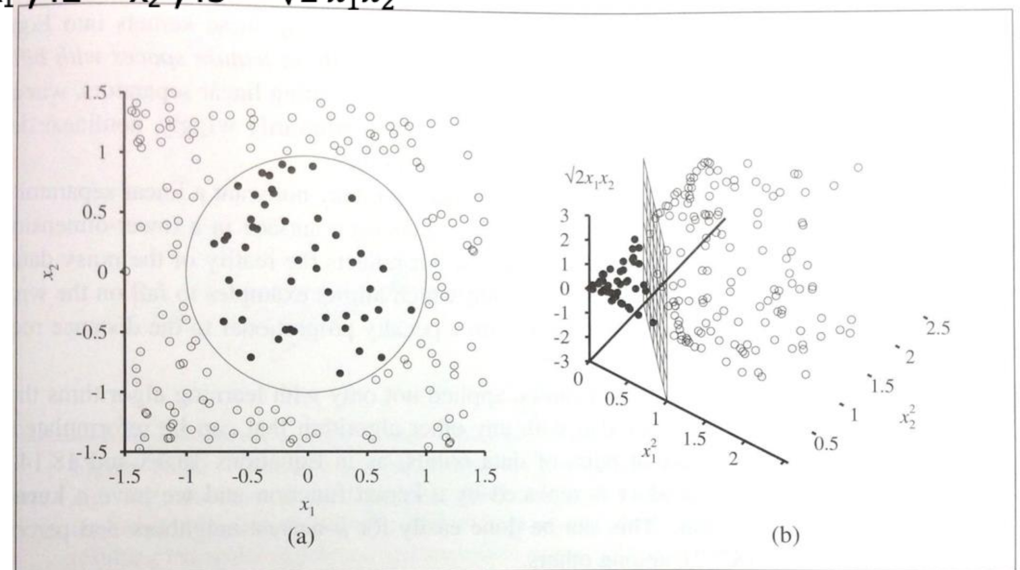


Figure 18.31 (a) A two-dimensional training set with positive examples as black circles and negative examples as white circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions. Figure 18.30(b) gives a closeup of the separator in (b).

(see textbook for details on how those new dimensions were chosen)

(Read the extra hand-out material for math details)

Learning decision trees

In some other problems, a single A vs. B classification is not sufficient. For example:

Problem: decide whether to wait for a table at a restaurant, based on the following attributes/features:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

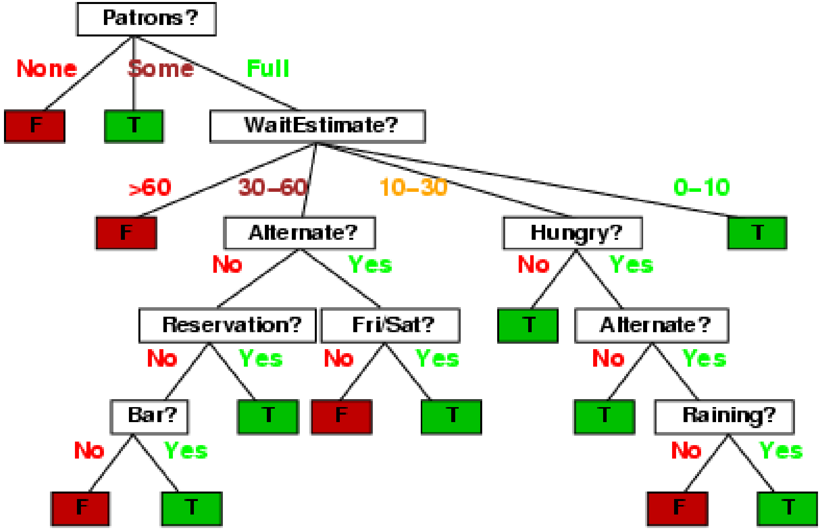
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

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

- Classification of examples is positive (T) or negative (F)

Decision trees

- One possible representation for hypotheses
- E.g., here is the “true” (designed manually by thinking about all cases) tree for deciding whether to wait:



- Could we learn this tree from examples instead of designing it by hand?

Inductive learning of decision tree

- **Simplest:** Construct a decision tree with one leaf for every example = memory based learning.
  Not very good generalization.

Inductive learning of decision tree

- **Simplest:** Construct a decision tree with one leaf for every example = memory based learning.
  Not very good generalization.
- **Advanced:** Split on each variable so that the purity of each split increases (i.e. either only yes or only no)
- Purity measured,e.g, with entropy

Inductive learning of decision tree

- **Simplest:** Construct a decision tree with one leaf for every example = memory based learning.
  Not very good generalization.
- **Advanced:** Split on each variable so that the purity of each split increases (i.e. either only yes or only no)
- Purity measured,e.g, with <u>entropy</u>

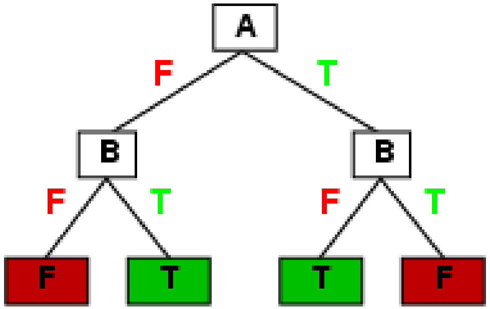
$$\boxed{\text{Entropy} = -P(yes)\ln[P(yes)] - P(no)\ln[P(no)]}$$

General form: $$\text{Entropy} = -\sum_i P(v_i)\ln[P(v_i)]$$

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

- Prefer to find more compact decision trees (Ockham's Razor)

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 possible trees

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

How many purely conjunctive hypotheses (e.g., *Hungry* ∧ ¬*Rain*)?
- Each attribute can be in (positive), in (negative), or out

  ⇒ 3^n distinct conjunctive hypotheses

The more expressive a hypothesis space is:
- increases chance that target function can be expressed
- increases number of hypotheses consistent with training set

  ⇒ may get worse predictions

ID3 Algorithm: Learning Decision Trees

- A greedy algorithm for decision tree construction developed by Ross Quinlan circa 1987

- Top-down construction of decision 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 selected attribute
  - Partition examples using the possible values of this attribute, and assign these subsets of the examples to the 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 are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the 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

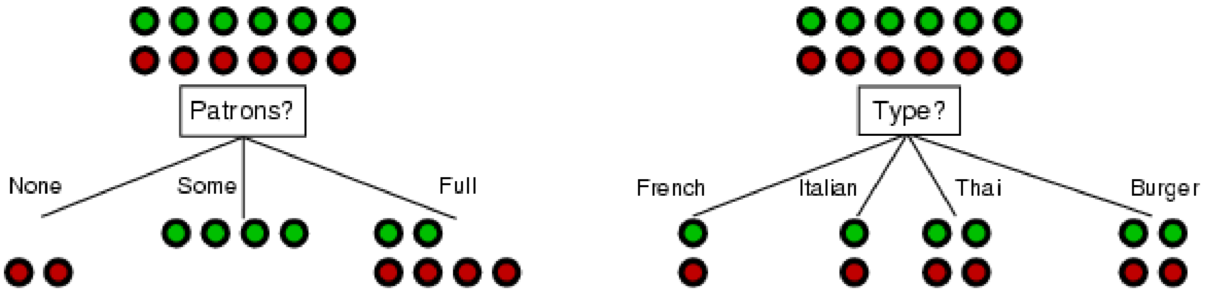
Decision tree learning

- Aim: find a small tree consistent with the training 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 ← 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

Using information theory

- To implement `Choose-Attribute` in the DTL algorithm

- Information Content (Entropy):

  $I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$ // $\log_2$, $\log_{10}$, or $\log_e$

- For a training set containing *p* positive examples and *n* negative examples:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

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) are shorter than less common ones (parliamentarian, foreshadowing)
  - In Morse code, common (probable) letters have shorter encodings
- Information is measured in 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.

Information Theory 101

- Information is measured in bits
- Information conveyed by message depends on its probability

- With n equally probable possible *messages*, the probability p of each is *1/n*
- Information conveyed by a 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:

$$I(P) = -\,[\; p_1 * \log(p_1) + \underbrace{p_2}_{\text{probability of msg 2}} * \underbrace{\log(p_2)}_{\text{information in msg 2}} + \ldots + p_n * \log(p_n) \;]$$

Information Theory II

- Information conveyed by distribution (a.k.a. *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*log(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

Information Gain

- A chosen attribute A divides the training set E into subsets $E_1, \ldots, E_v$ according to their values for A, where A has v distinct values.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Information Gain (IG) or reduction in entropy from the attribute test:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- Choose the attribute with the largest IG

Information gain

For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

Consider the attributes *Patrons* and *Type* (and others too):

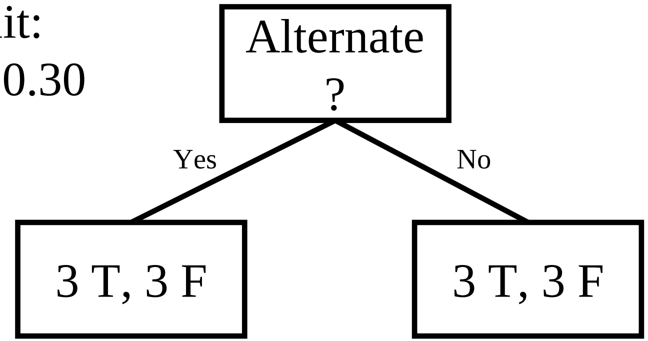
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Decision tree learning example

Before split:
entropy = 0.30



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

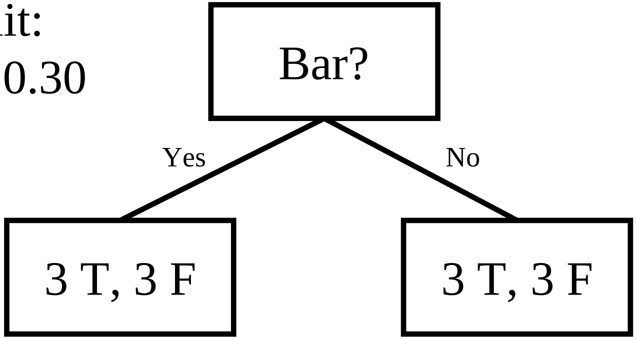
After split: $\text{Entropy} = \frac{6}{12}\left[-\left(\frac{3}{6}\right)\ln\left(\frac{3}{6}\right)-\left(\frac{3}{6}\right)\ln\left(\frac{3}{6}\right)\right]+\frac{6}{12}\left[-\left(\frac{3}{6}\right)\ln\left(\frac{3}{6}\right)-\left(\frac{3}{6}\right)\ln\left(\frac{3}{6}\right)\right]=0.30$

Entropy decrease = 0.30 – 0.30 = 0

NOTE: Please replace “ln” by “$\log_{10}$” in the above statement.

Decision tree learning example

Before split:
entropy = 0.30



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

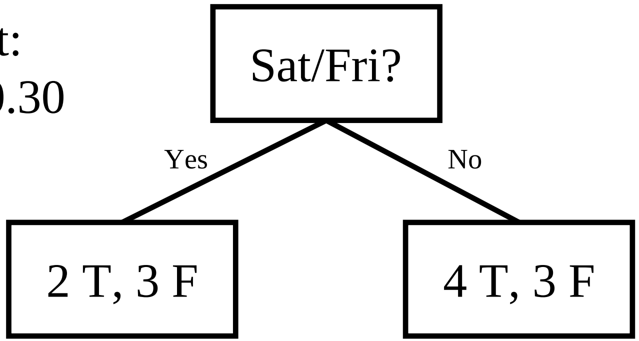
After split:

$$\text{Entropy} = \frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)\right]+\frac{6}{12}\left[-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)-\left(\tfrac{3}{6}\right)\ln\left(\tfrac{3}{6}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example

Before split:
entropy = 0.30



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

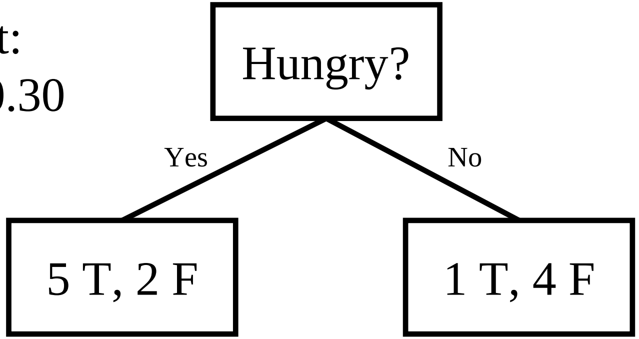
After split:

$$\text{Entropy} = \frac{5}{12}\left[-\left(\tfrac{2}{5}\right)\ln\left(\tfrac{2}{5}\right)-\left(\tfrac{3}{5}\right)\ln\left(\tfrac{3}{5}\right)\right]+\frac{7}{12}\left[-\left(\tfrac{4}{7}\right)\ln\left(\tfrac{4}{7}\right)-\left(\tfrac{3}{7}\right)\ln\left(\tfrac{3}{7}\right)\right]=0.29$$

Entropy decrease = 0.30 – 0.29 = 0.01

Decision tree learning example

Before split:
entropy = 0.30



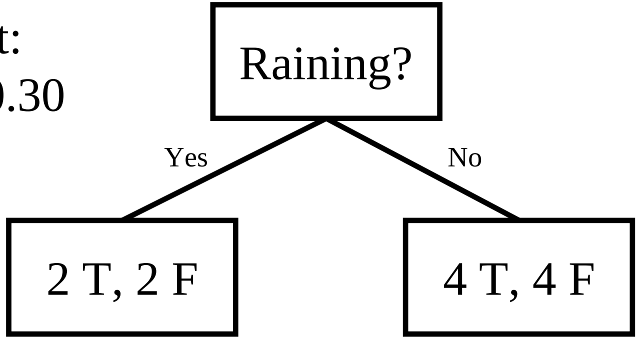
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	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

After split: $$\text{Entropy} = \frac{7}{12}\left[-\left(\tfrac{5}{7}\right)\ln\left(\tfrac{5}{7}\right)-\left(\tfrac{2}{7}\right)\ln\left(\tfrac{2}{7}\right)\right]+\frac{5}{12}\left[-\left(\tfrac{1}{5}\right)\ln\left(\tfrac{1}{5}\right)-\left(\tfrac{4}{5}\right)\ln\left(\tfrac{4}{5}\right)\right]=0.24$$

Entropy decrease = 0.30 – 0.24 = 0.06

Decision tree learning example

Before split:
entropy = 0.30



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

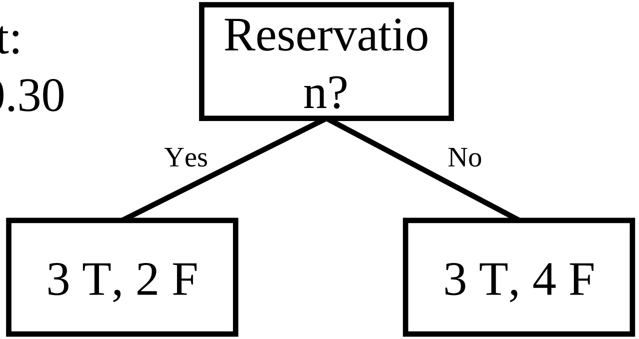
After split:

$$\text{Entropy} = \frac{4}{12}\left[-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)\right]+\frac{8}{12}\left[-\left(\tfrac{4}{8}\right)\ln\left(\tfrac{4}{8}\right)-\left(\tfrac{4}{8}\right)\ln\left(\tfrac{4}{8}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example

Before split:
entropy = 0.30



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

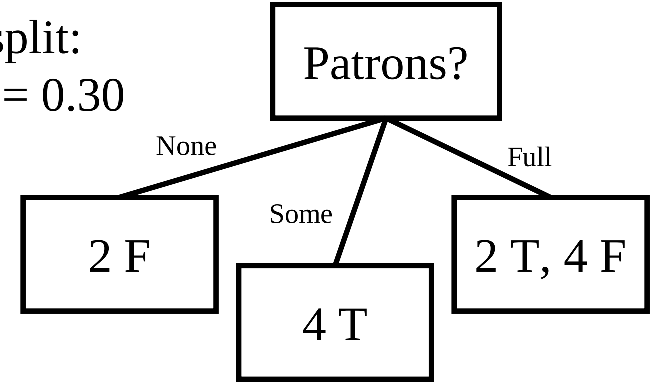
After split:

$$\text{Entropy} = \frac{5}{12}\left[-\left(\tfrac{3}{5}\right)\ln\left(\tfrac{3}{5}\right)-\left(\tfrac{2}{5}\right)\ln\left(\tfrac{2}{5}\right)\right]+\frac{7}{12}\left[-\left(\tfrac{3}{7}\right)\ln\left(\tfrac{3}{7}\right)-\left(\tfrac{4}{7}\right)\ln\left(\tfrac{4}{7}\right)\right]=0.29$$

Entropy decrease = 0.30 – 0.29 = 0.01

Decision tree learning example

Before split:
entropy = 0.30



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

After split:

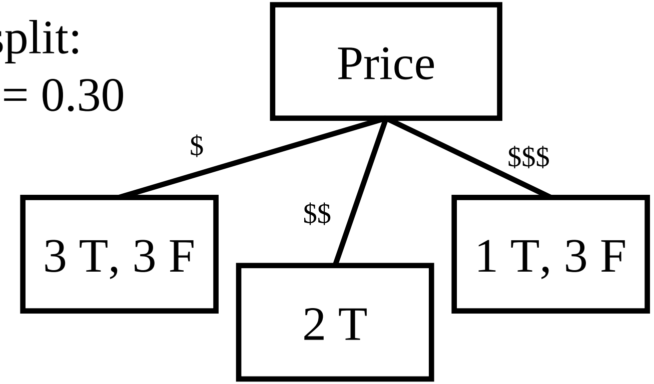
$$\text{Entropy} = \frac{2}{12}\left[-\left(\tfrac{0}{2}\right)\ln\left(\tfrac{0}{2}\right) - \left(\tfrac{2}{2}\right)\ln\left(\tfrac{2}{2}\right)\right] + \frac{4}{12}\left[-\left(\tfrac{4}{4}\right)\ln\left(\tfrac{4}{4}\right) - \left(\tfrac{0}{4}\right)\ln\left(\tfrac{0}{4}\right)\right]$$

$$+\frac{6}{12}\left[-\left(\tfrac{2}{6}\right)\ln\left(\tfrac{2}{6}\right) - \left(\tfrac{4}{6}\right)\ln\left(\tfrac{4}{6}\right)\right] = 0.14$$

Entropy decrease = 0.30 – 0.14 = 0.16

Decision tree learning example

Before split:
entropy = 0.30



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

After split:

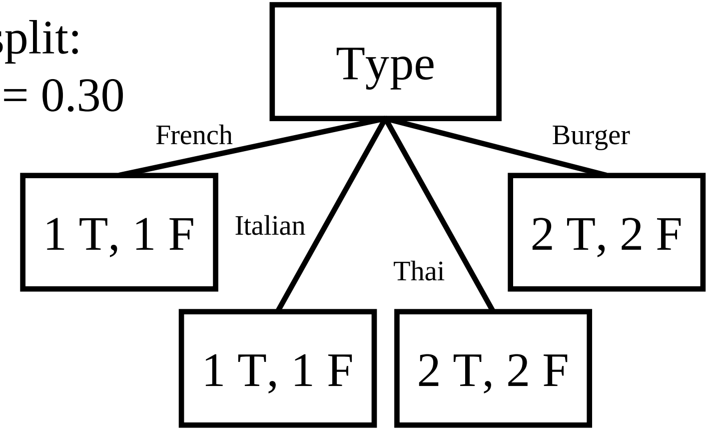
$$\text{Entropy} = \frac{6}{12}\left[-\left(3/6\right)\ln\left(3/6\right) - \left(3/6\right)\ln\left(3/6\right)\right] + \frac{2}{12}\left[-\left(2/2\right)\ln\left(2/2\right) - \left(0/2\right)\ln\left(0/2\right)\right]$$

$$+\frac{4}{12}\left[-\left(1/4\right)\ln\left(1/4\right) - \left(3/4\right)\ln\left(3/4\right)\right] = 0.23$$

Entropy decrease = 0.30 – 0.23 = 0.07

Decision tree learning example

Before split:
entropy = 0.30



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

After split:

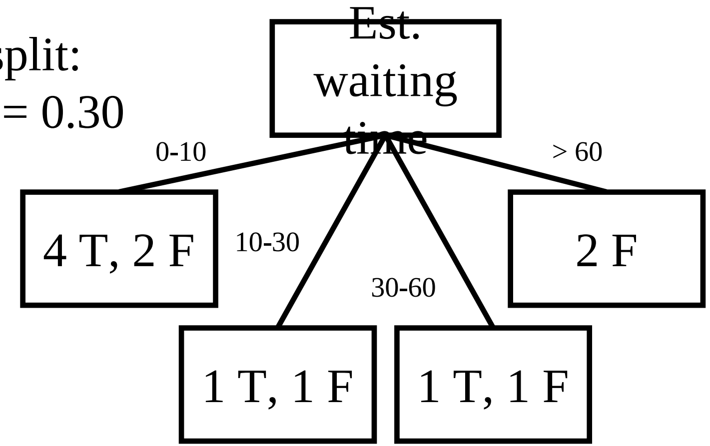
$$\text{Entropy} = \frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)\right]$$

$$+\frac{4}{12}\left[-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)\right]+\frac{4}{12}\left[-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)-\left(\tfrac{2}{4}\right)\ln\left(\tfrac{2}{4}\right)\right]=0.30$$

Entropy decrease = 0.30 – 0.30 = 0

Decision tree learning example

Before split:
entropy = 0.30



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

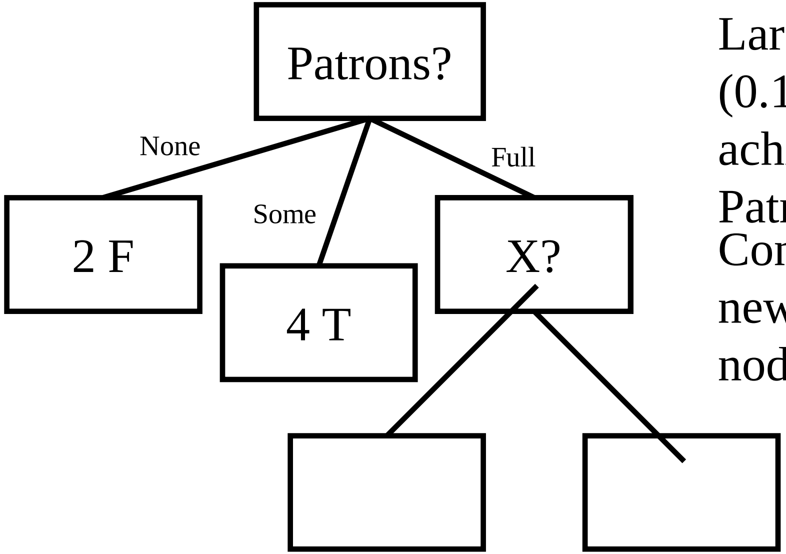
After split:

$$\text{Entropy} = \frac{6}{12}\left[-\left(\tfrac{4}{6}\right)\ln\left(\tfrac{4}{6}\right)-\left(\tfrac{2}{6}\right)\ln\left(\tfrac{2}{6}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)\right]$$

$$+\frac{2}{12}\left[-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)-\left(\tfrac{1}{2}\right)\ln\left(\tfrac{1}{2}\right)\right]+\frac{2}{12}\left[-\left(\tfrac{0}{2}\right)\ln\left(\tfrac{0}{2}\right)-\left(\tfrac{2}{2}\right)\ln\left(\tfrac{2}{2}\right)\right]=0.24$$

Entropy decrease = 0.30 – 0.24 = 0.06

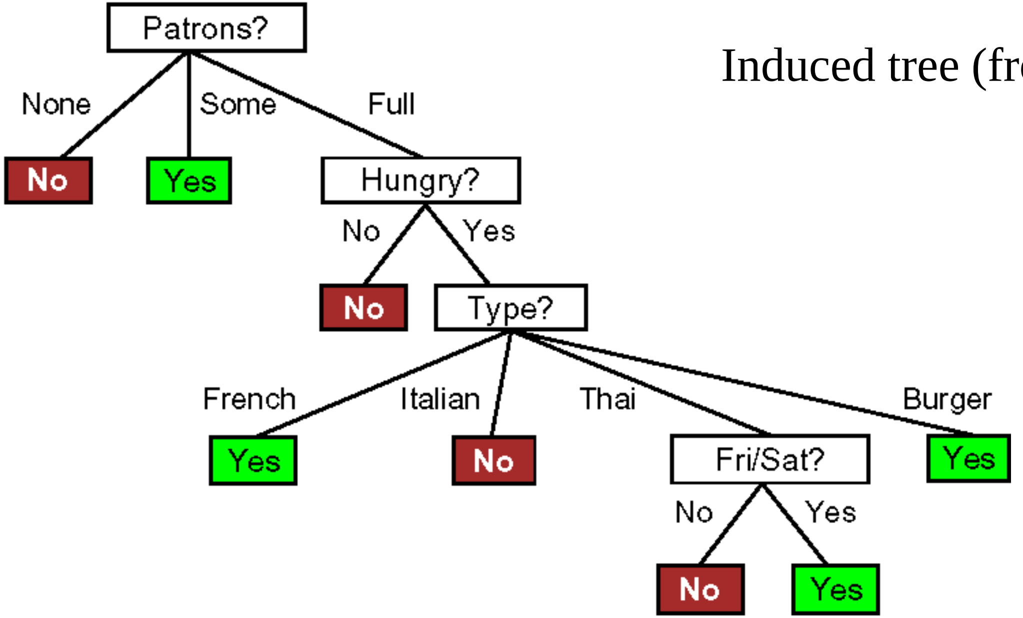
Decision tree learning example



Largest entropy decrease (0.16)
achieved by splitting on Patrons.

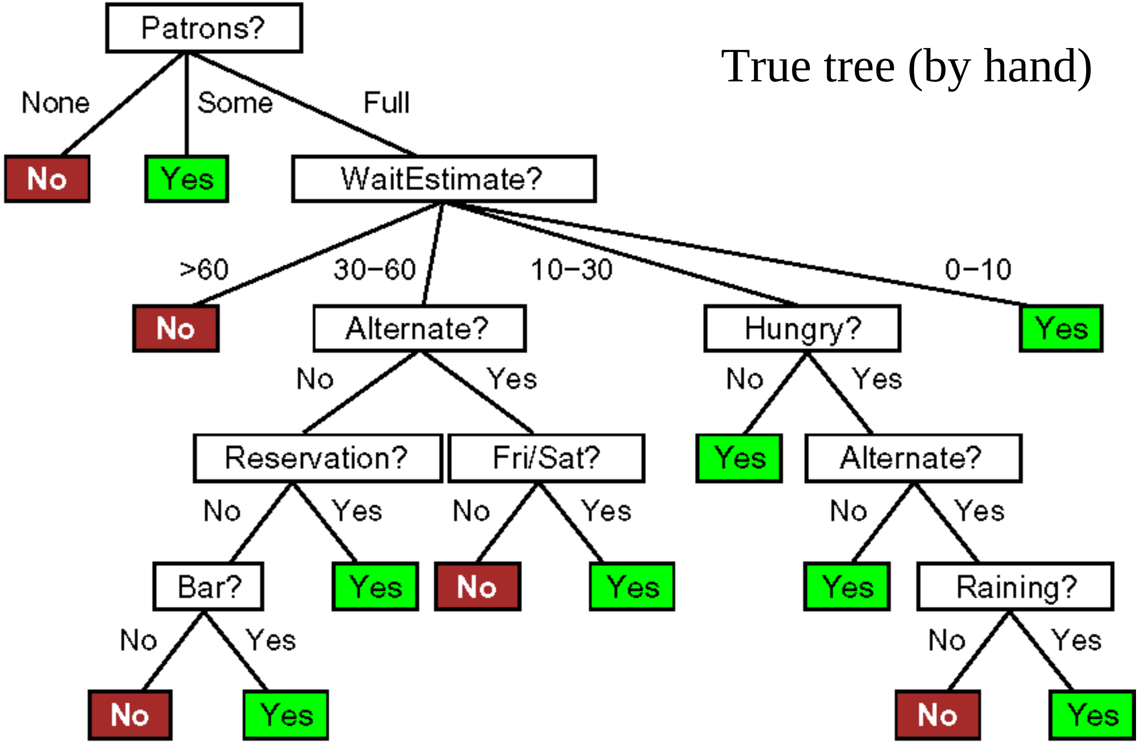
Continue like this, making new splits, always purifying nodes.

Decision tree learning example



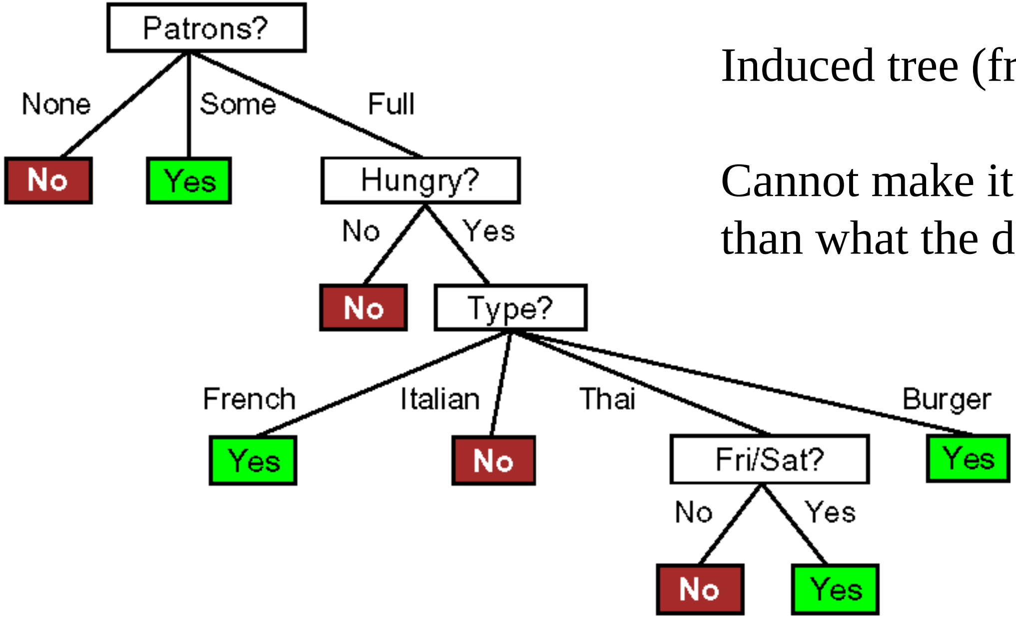
Induced tree (from examples)

Decision tree learning example



True tree (by hand)

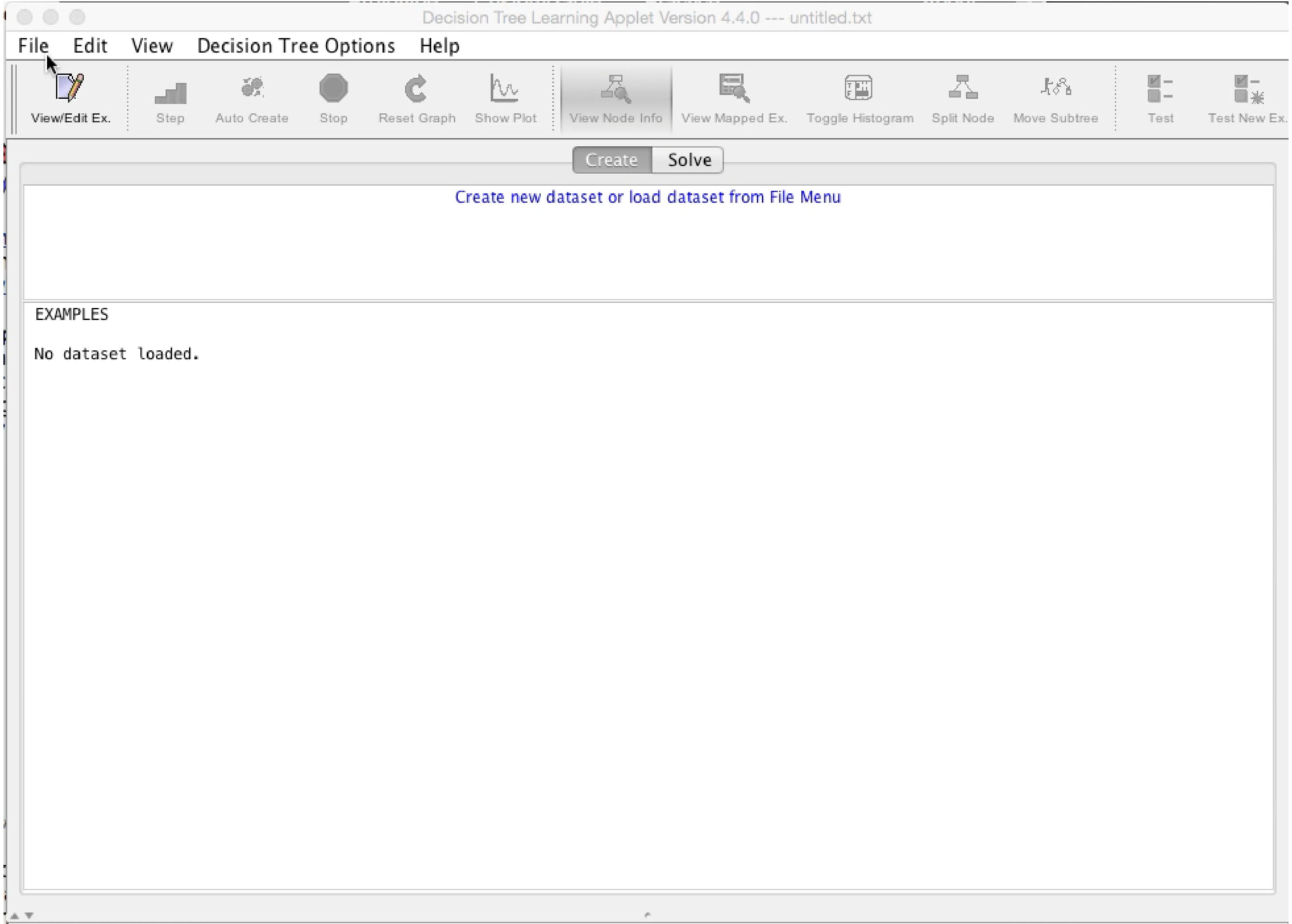
Decision tree learning example



Induced tree (from examples)

Cannot make it more complex than what the data supports.

Demo


Decision Tree Learning Applet Version 4.4.0 --- untitled.txt
File
Edit
View
Decision Tree Options
Help
View/Edit Ex.
Step
Auto Create
Stop
Reset Graph
Show Plot
View Node Info
View Mapped Ex.
Toggle Histogram
Split Node
Move Subtree
Test
Test New Ex.
Create
Solve
Create new dataset or load dataset from File Menu
EXAMPLES
No dataset loaded.

How do we know it is correct?

How do we know that $h \approx f$?

(Hume's Problem of Induction)

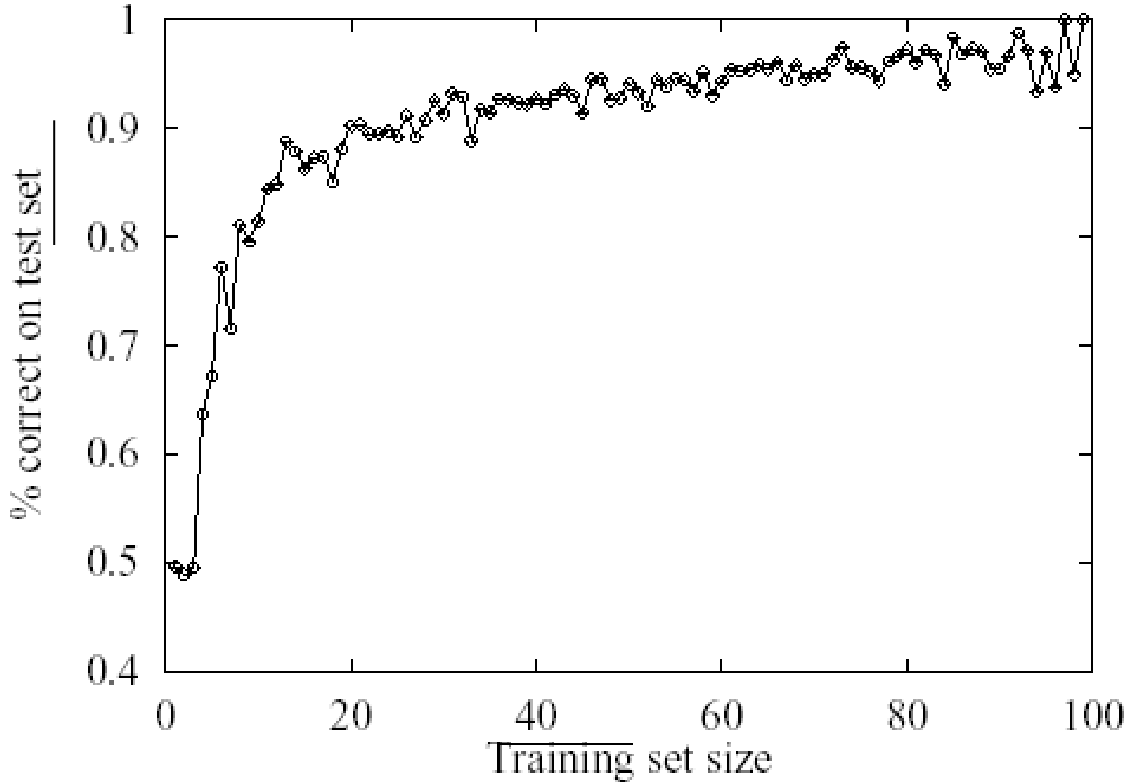
- Try h on a new test set of examples (cross validation)

...and assume the "principle of uniformity", i.e. the result we get on this test data should be indicative of results on future data. Causality is constant.

Inspired by a slide by V. Pavlovic

Learning curve for the decision tree algorithm on 100 randomly generated examples in the restaurant domain.

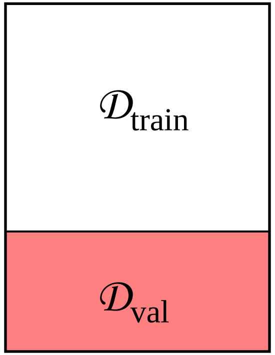
The graph summarizes 20 trials.



Cross-validation

Use a “validation set”.

$$E_{gen} \approx E_{val}$$



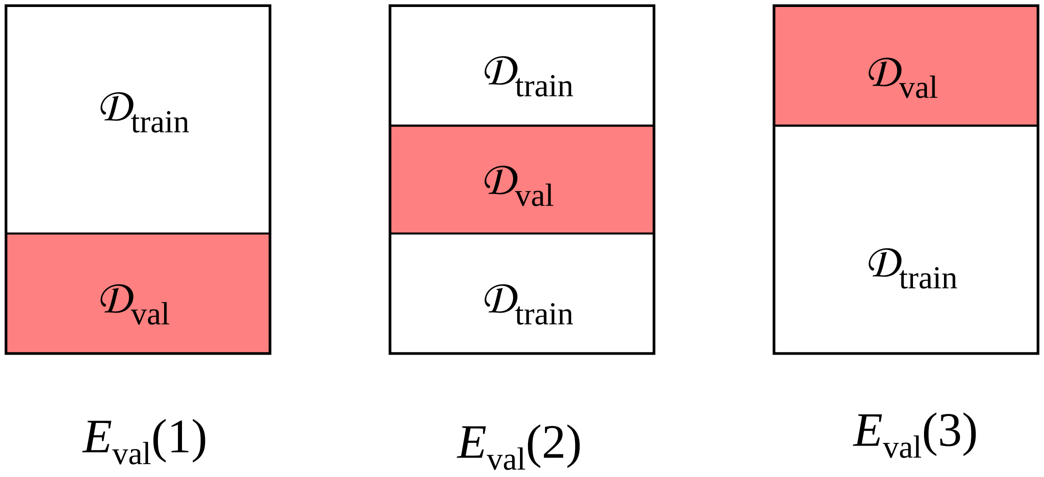
E_{val}

Split your data set into two parts, one for training your model and the other for validating your model.
The error on the validation data is called “validation error” (E_{val})

K-Fold Cross-validation

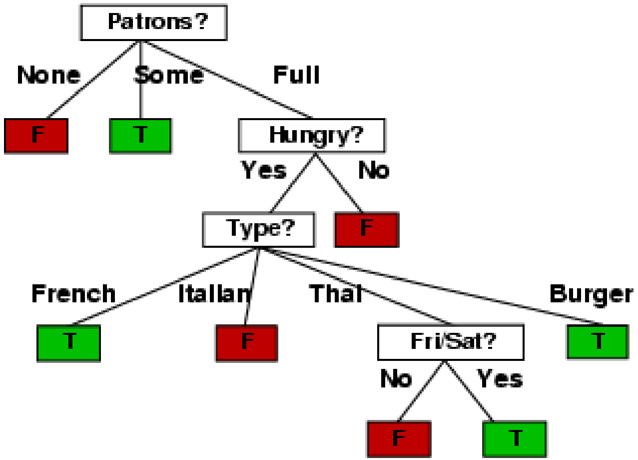
More accurate than using only one validation set.

$$E_{gen} \approx \langle E_{val} \rangle = \frac{1}{K} \sum_{k=1}^{K} E_{val}(k)$$



Example contd.

- Decision tree learned from the 12 examples:



- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set