From inductive inference to machine learning

adapted from AIMA slides

Russel&Norvig:Artificial Intelligence: a modern approach

AIMA: Inductive inference

Outline

- ♢ Bayesian inferences with multiple models
- ♢ Bayesian learning
- ♢ Maximum *a posteriori* and maximum likelihood learning
- ♢ Bayes net learning
  - ML parameter learning with complete data
  - linear regression
- ♢ Inductive learning
- ♢ Decision tree learning
- ♢ Measuring learning performance

Bayesian inference with multiple models

Assume multiple models $M_i = (S_i, \theta_i)$ with prior $p(M_i)$ $i = 1, \ldots, M$.

The inference $p(Q = q|E = e)$ can be performed as follows:

$$p(q|e) = \Sigma_{i=1,\ldots,M} p(q, M_i|e) = \Sigma_{i=1,\ldots,M} p(q|M_i, e)p(M_i|e)$$

Note that $p(M_i|e)$ is a posterior over models with evidence e:

$$p(M_i|e) = \frac{p(e|M_i)p(M_i)}{p(e)} \propto p(e|M_i)p(M_i)$$

i.e., the evidence e reweight our beliefs in multiple models.

The inference is performed by **Bayesian Model Averaging** (BMA). Epicurus' (342(?) B.C. - 270 B.C.) **principle of multiple explanations** which states that one should keep all hypotheses that are consistent with the data.

Bayesian model averaging with data

Beside models, assume N multiple complete observations D_N.

The standard inference $p(Q = q|E = e, D_N)$ is defined as:

$$p(q|e, D_N) = \Sigma_{i=1,\ldots,M} p(q, M_i|e, D_N) = \Sigma_{i=1,\ldots,M} p(q|M_i, e, D_N) p(M_i|e, D_N)$$

Because $p(q|M_i, e, D_N) = p(q|M_i, e)$ and $p(M_i|e, D_N) \approx p(M_i|D_N)$:

$$p(q|e, D_N) \approx \Sigma_{i=1,\ldots,M} p(q|M_i, e) p(M_i|D_N)$$

where again $p(M_i|D_N)$ is a posterior after observations D_N:

$$p(M_i|D_N) = \frac{p(D_N|M_i)p(M_i)}{p(e)} \propto \underbrace{p(D_N|M_i)}_{likelihood} \underbrace{p(M_i)}_{prior}.$$

i.e., our rational foundation, probability theory, automatically includes and normatively defines learning from observations as standard Bayesian inference!

Full Bayesian learning

View learning as Bayesian updating of a probability distribution over the hypothesis space

H is the hypothesis variable, values $h_1, h_2, \ldots$, prior $\mathbf{P}(H)$ jth observation d_j gives the outcome of random variable D_j training data $\mathbf{d} = d_1, \ldots, d_N$

Given the data so far, each hypothesis has a posterior probability:

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

where $P(\mathbf{d}|h_i)$ is called the likelihood

Predictions use a likelihood-weighted average over the hypotheses:

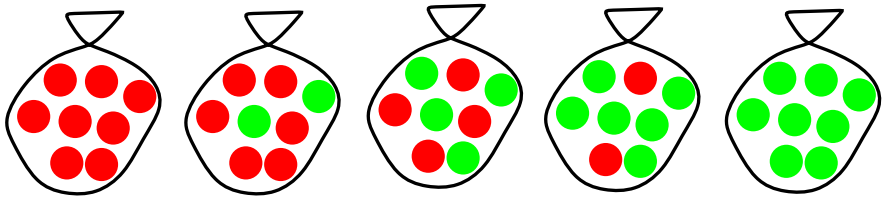
$$\mathbf{P}(X|\mathbf{d}) = \Sigma_i \, \mathbf{P}(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = \Sigma_i \, \mathbf{P}(X|h_i)P(h_i|\mathbf{d})$$

No need to pick one best-guess hypothesis!

Example

Suppose there are five kinds of bags of candies:

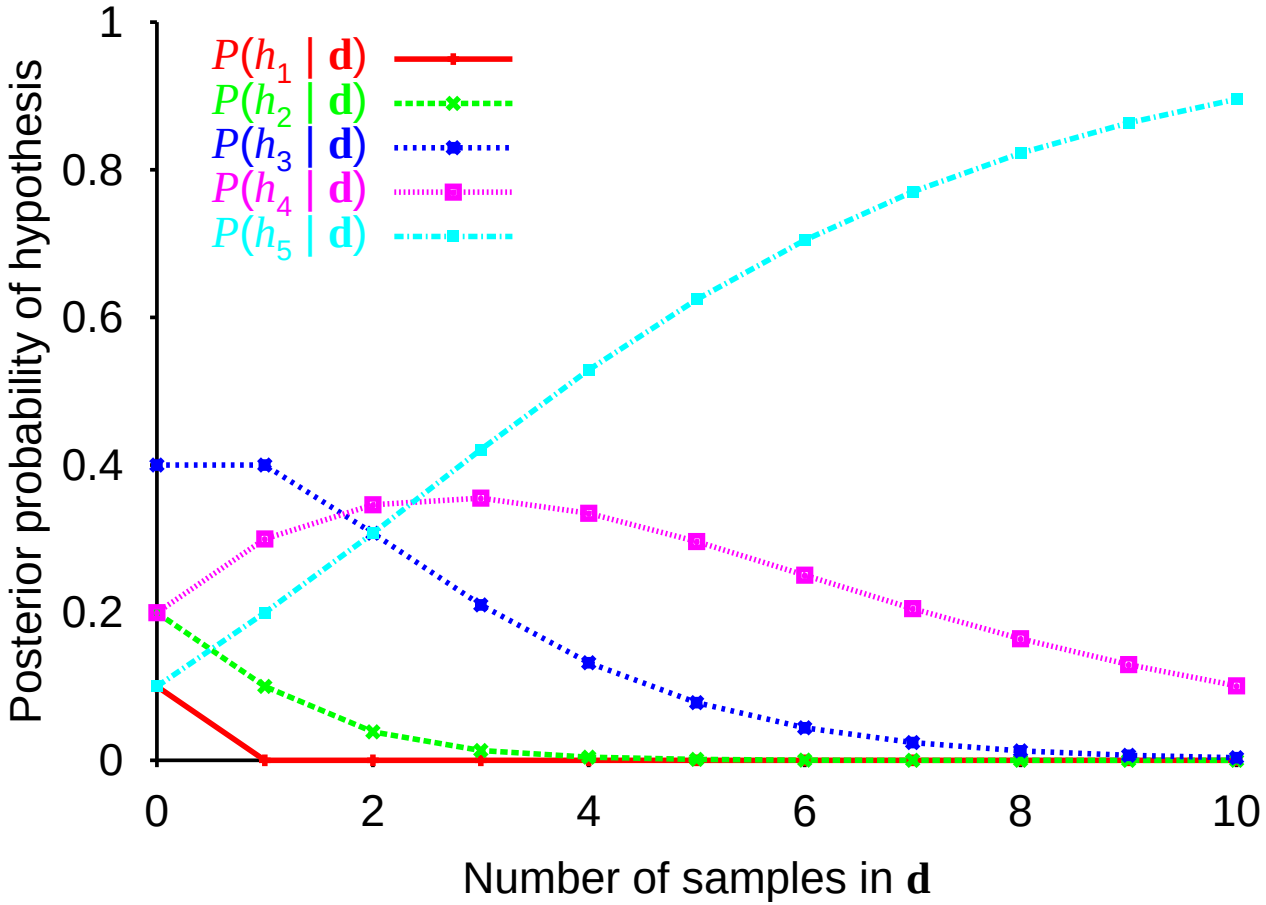
10% are h_1: 100% cherry candies
20% are h_2: 75% cherry candies + 25% lime candies
40% are h_3: 50% cherry candies + 50% lime candies
20% are h_4: 25% cherry candies + 75% lime candies
10% are h_5: 100% lime candies



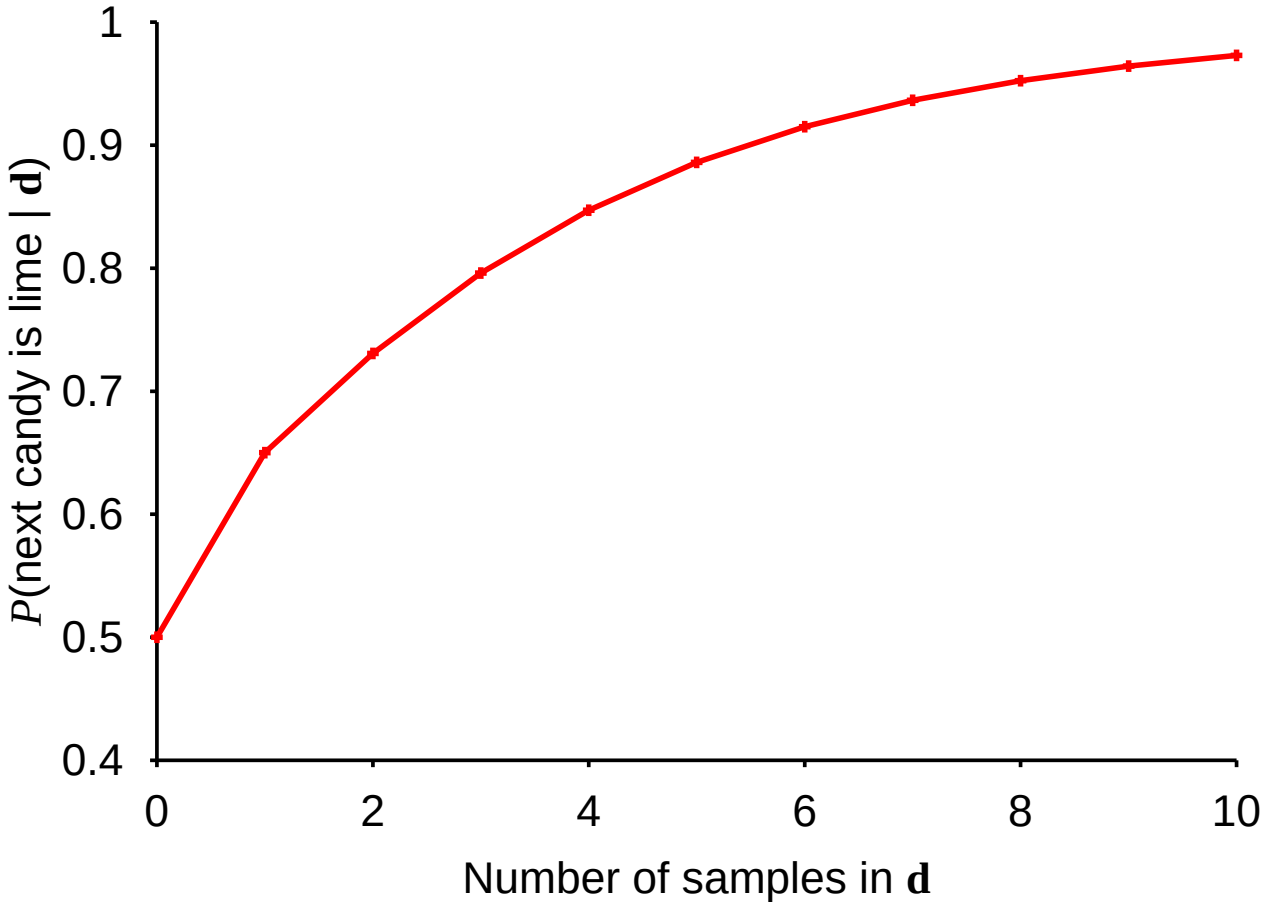
Then we observe candies drawn from some bag: 

What kind of bag is it? What flavour will the next candy be?

Posterior probability of hypotheses



Prediction probability



MAP approximation

Summing over the hypothesis space is often intractable
(e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)

Maximum a posteriori (MAP) learning: choose h_{MAP} maximizing $P(h_i|\mathbf{d})$

I.e., maximize $P(\mathbf{d}|h_i)P(h_i)$ or $\log P(\mathbf{d}|h_i) + \log P(h_i)$

Log terms can be viewed as (negative of)

bits to encode data given hypothesis + bits to encode hypothesis

This is the basic idea of minimum description length (MDL) learning

For deterministic hypotheses, $P(\mathbf{d}|h_i)$ is 1 if consistent, 0 otherwise

$\Rightarrow$ MAP = simplest consistent hypothesis (cf. science)

ML approximation

For large data sets, prior becomes irrelevant

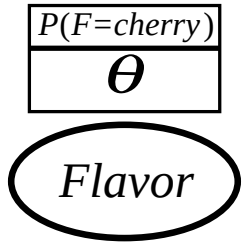
Maximum likelihood (ML) learning: choose h_{ML} maximizing $P(\mathbf{d}|h_i)$

I.e., simply get the best fit to the data; identical to MAP for uniform prior
(which is reasonable if all hypotheses are of the same complexity)

ML is the "standard" (non-Bayesian) statistical learning method

ML parameter learning in Bayes nets

Bag from a new manufacturer; fraction θ of cherry candies?
Any θ is possible: continuum of hypotheses h_θ
θ is a parameter for this simple (binomial) model.



Suppose we unwrap N candies, c cherries and $\ell = N - c$ limes
These are i.i.d. (independent, identically distributed) observations,

$$P(\mathbf{d}|h_\theta) = \prod_{j=1}^{N} P(d_j|h_\theta) = \theta^c \cdot (1-\theta)^\ell$$

Maximize this w.r.t. θ—which is easier for the log-likelihood:

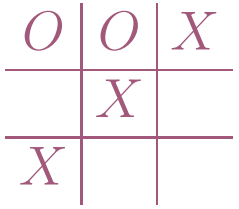
$$L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = \sum_{j=1}^{N} \log P(d_j|h_\theta) = c\log\theta + \ell\log(1-\theta)$$

$$\frac{dL(\mathbf{d}|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c+\ell} = \frac{c}{N}$$

Inductive learning (a.k.a. Science)

Simplest form: learn a function from examples (**tabula rasa**)

f is the target function

An example is a pair x, $f(x)$, e.g.,  , $+1$

Problem: find a(n) hypothesis h such that $h \approx f$ given a training set of examples.

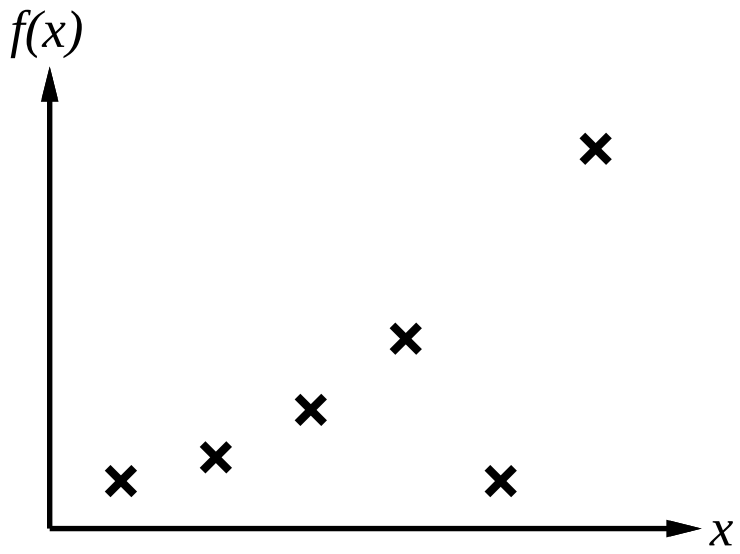
(This is a highly simplified model of real learning:

- **Ignores prior knowledge**
- **Assumes a deterministic, observable "environment"**
- **Assumes examples are given**
- **Assumes that the agent wants to learn f—why?)**

Inductive learning method

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

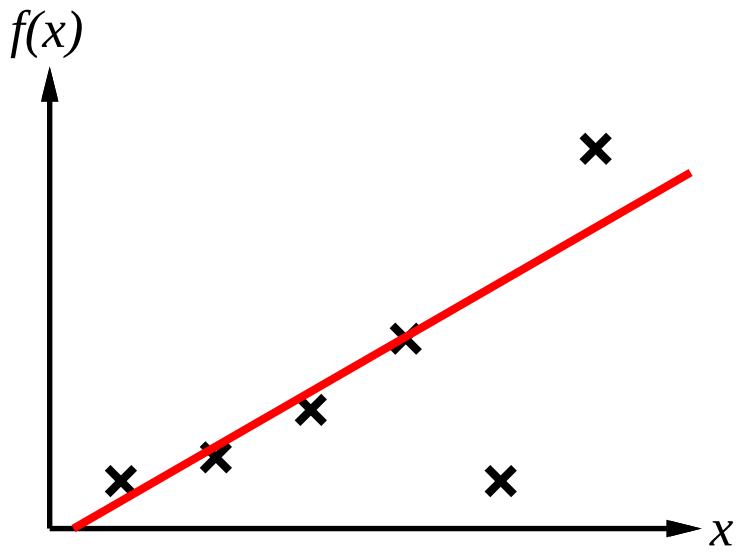
E.g., curve fitting:



Inductive learning method

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

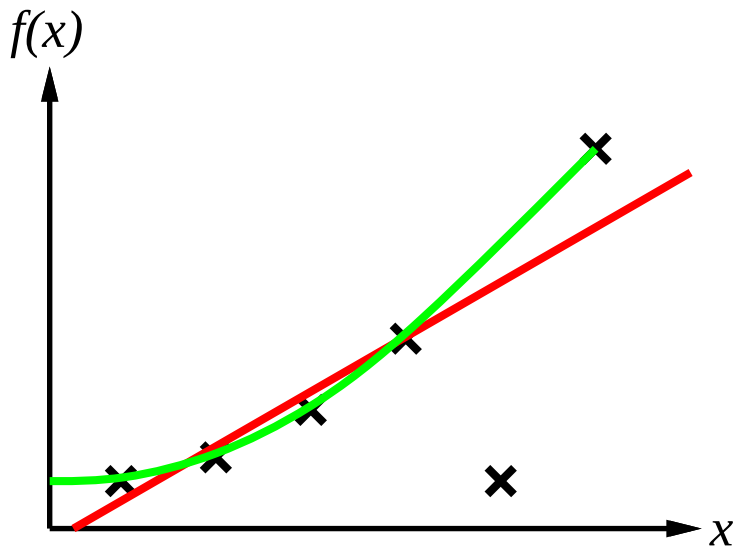
E.g., curve fitting:



Inductive learning method

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

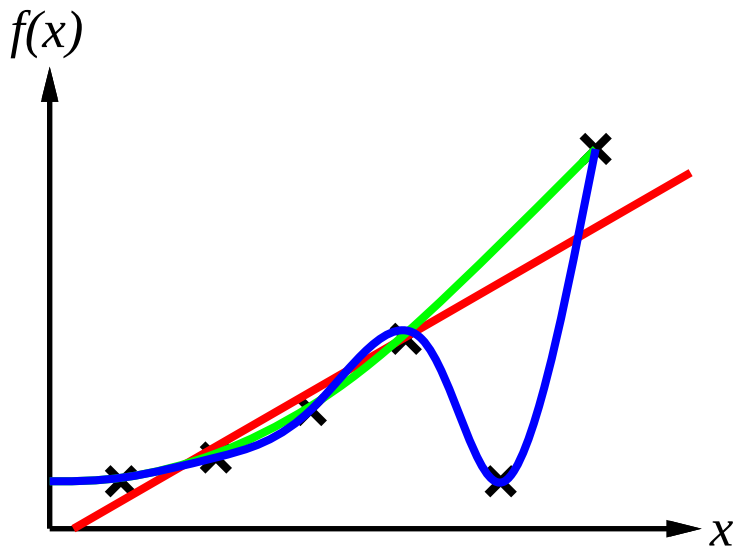
E.g., curve fitting:



Inductive learning method

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

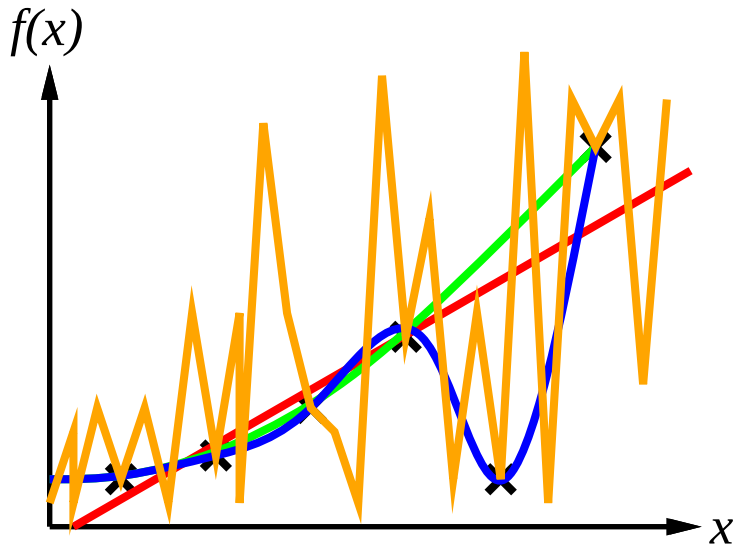
E.g., curve fitting:



Inductive learning method

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

E.g., curve fitting:



Ockham's razor

Ockham's razor: balance consistency and simplicity

The principle of Occam's razor (1285 - 1349, sometimes spelt Ockham). Occam's razor states that **when inferring causes entities should not be multiplied beyond necessity**. This is widely understood to mean: Among all hypotheses consistent with the observations, choose the simplest.

In terms of a prior distribution over hypotheses, this is the same as giving simpler hypotheses higher a priori probability, and more complex ones lower probability.

Attribute-based representations

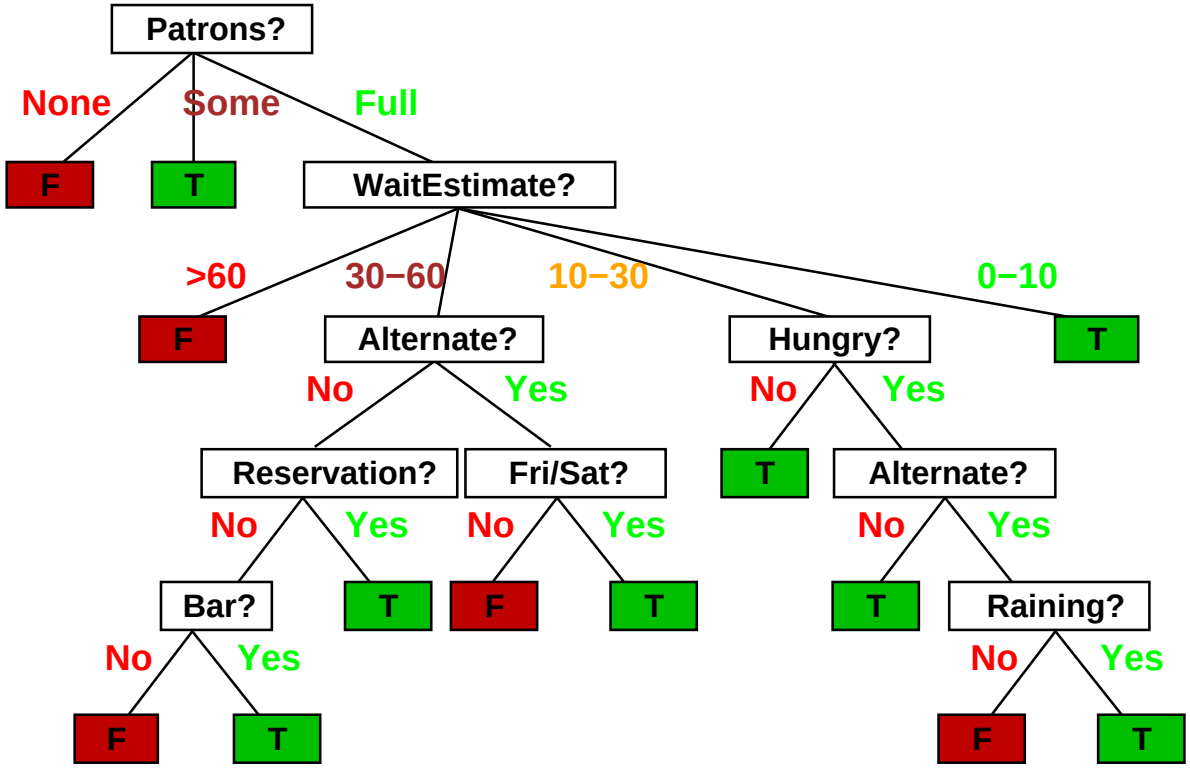
Examples described by attribute values (Boolean, discrete, continuous, etc.), e.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*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*

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

Decision trees

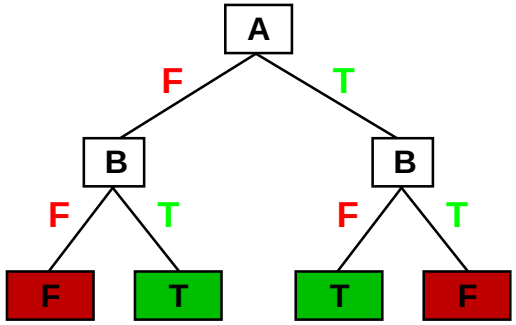
Common representation for protocols, e.g. here is the "true" tree for deciding whether to wait:



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\rightarrow$ 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.

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows

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}

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

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., $Hungry \wedge \neg Rain$)??

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., $Hungry \wedge \neg Rain$)??

Each attribute can be in (positive), in (negative), or out
$\Rightarrow$ 3^n distinct conjunctive hypotheses

More expressive hypothesis space
- increases chance that target function can be expressed
- increases number of hypotheses consistent w/ training set
  $\Rightarrow$ may get worse predictions

Decision tree learning

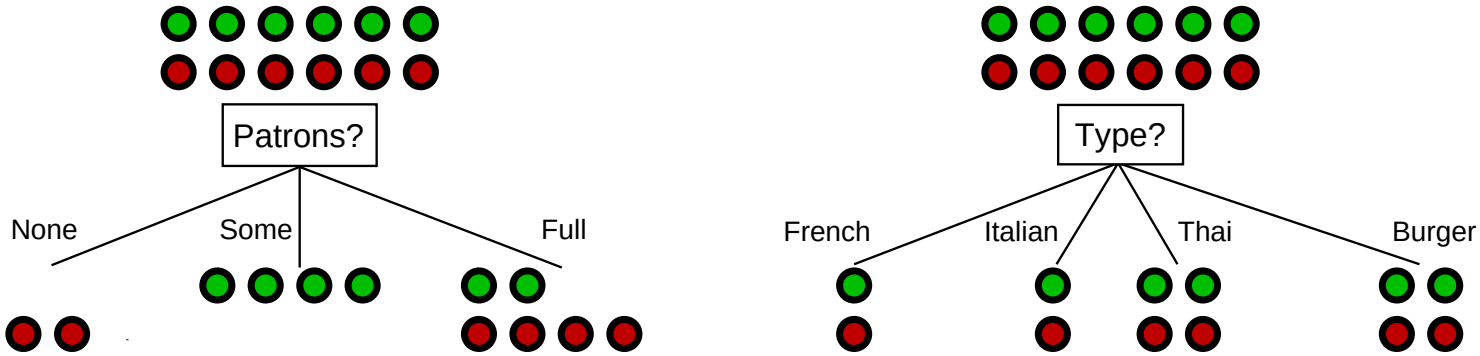
Aim: find a small tree consistent with the training examples

Idea: recursively choose "most significant" attribute to branch

```
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—gives **information** about the classification

Information

Information answers questions

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 $\langle P_1, \ldots, P_n \rangle$ is

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

(also called entropy of the prior)

Information contd.

Suppose we have p positive and n negative examples at the root
$\Rightarrow$ $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example
E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

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

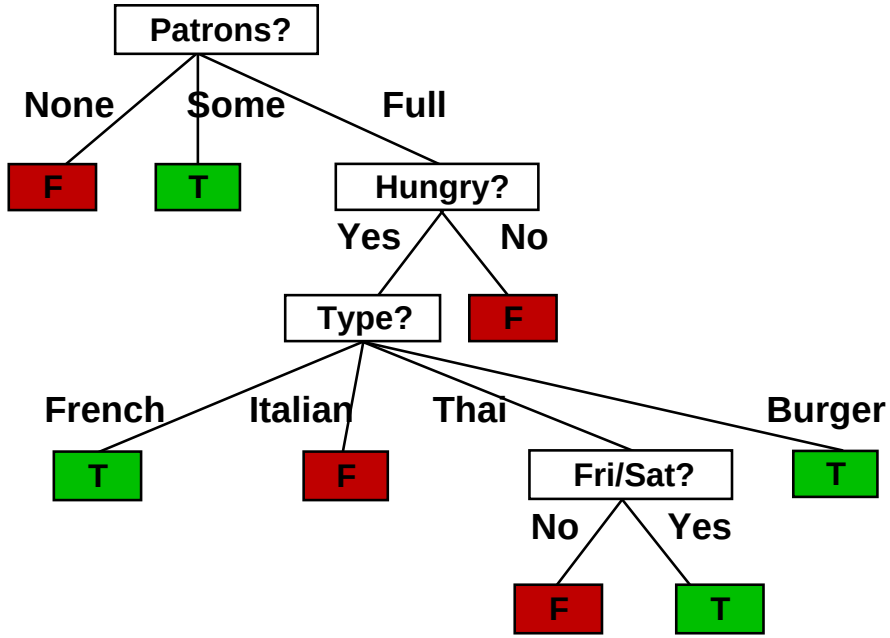
Let E_i have p_i positive and n_i negative examples $\Rightarrow$ $H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$ bits needed to classify a new example $\Rightarrow$ **expected** number of bits per example over all branches is

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

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit $\Rightarrow$ choose the attribute that minimizes the remaining information needed

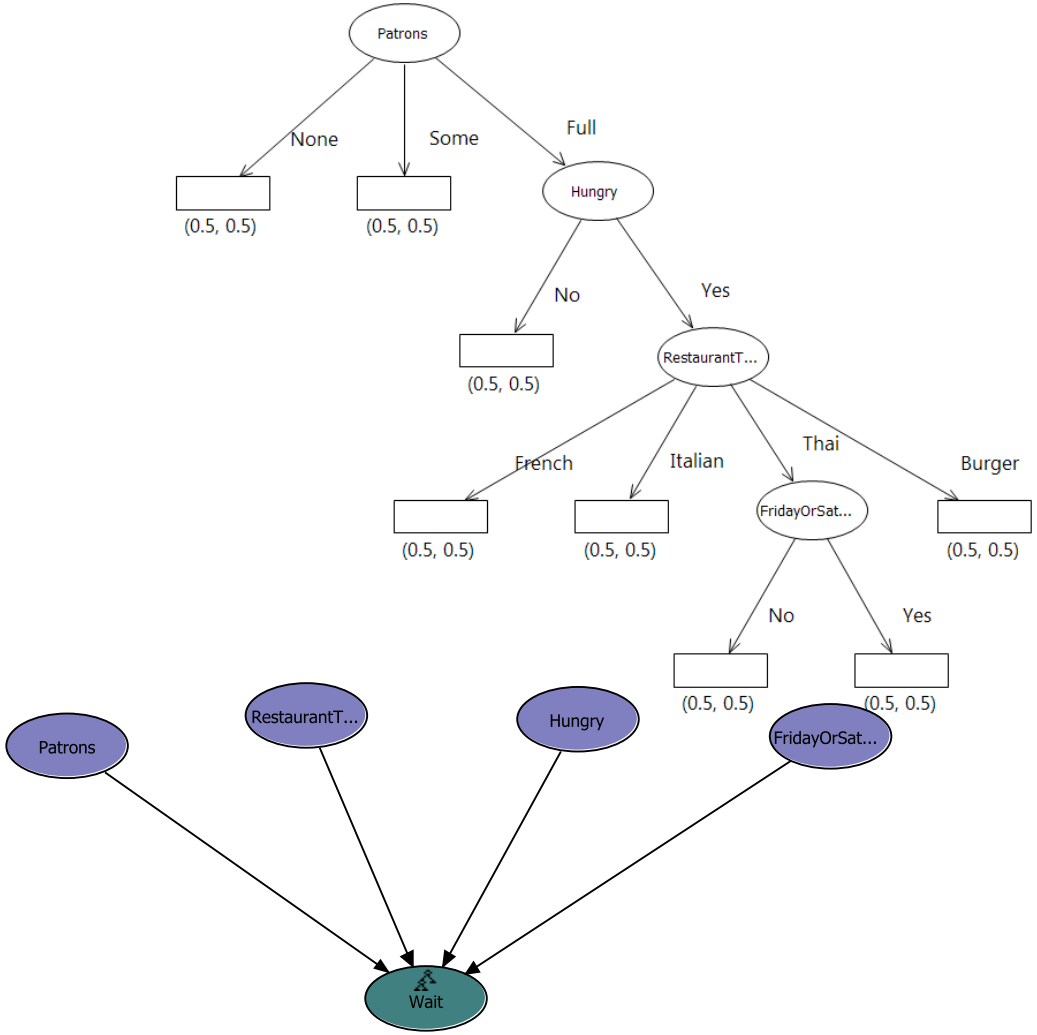
Example contd.

Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Decision tree as local conditional model



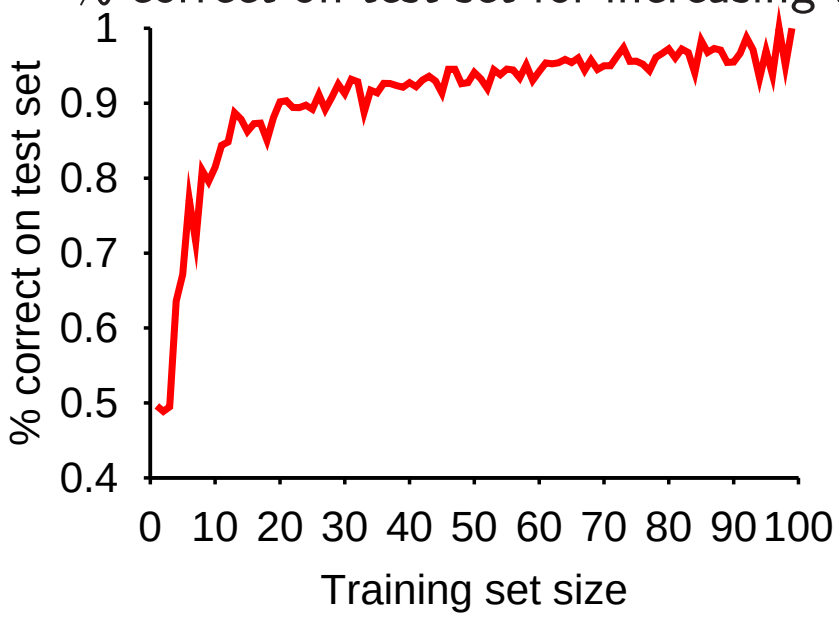
Performance measurement

How do we know that $h \approx f$? (Hume's **Problem of Induction**)

1) Use theorems of computational/statistical learning theory

2) Try h on a new test set of examples (use **same distribution over example space** as training set)

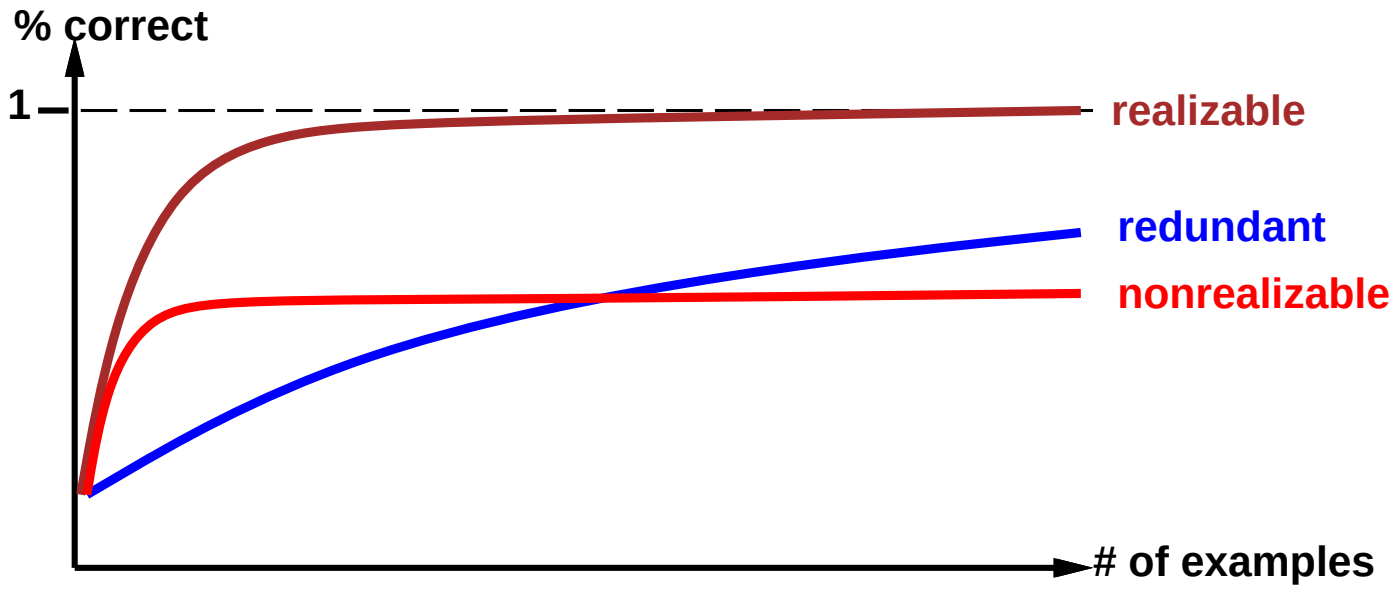
Learning curve = % correct on test set for increasing training set size



Performance measurement contd.

Learning curve depends on

- realizable (can express target function) vs. non-realizable
  non-realizability can be due to missing attributes
  or restricted hypothesis class (e.g., thresholded linear function)
- redundant expressiveness (e.g., loads of irrelevant attributes)



Summary

Learning needed for unknown environments, lazy designers

Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation

Full Bayesian learning gives best possible predictions but is intractable

MAP learning balances complexity with accuracy on training data

Maximum likelihood assumes uniform prior, OK for large data sets

For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples

Decision tree learning using information gain

Learning performance = prediction accuracy measured on test set