CS5010
Artificial Intelligence Principles:
Lecture 4

Learning from Observations

Chapter 18, Sections 1–3

Outline

$\Diamond$ Learning agents

$\Diamond$ Inductive learning

$\Diamond$ Decision tree learning

$\Diamond$ Measuring learning performance

Learning

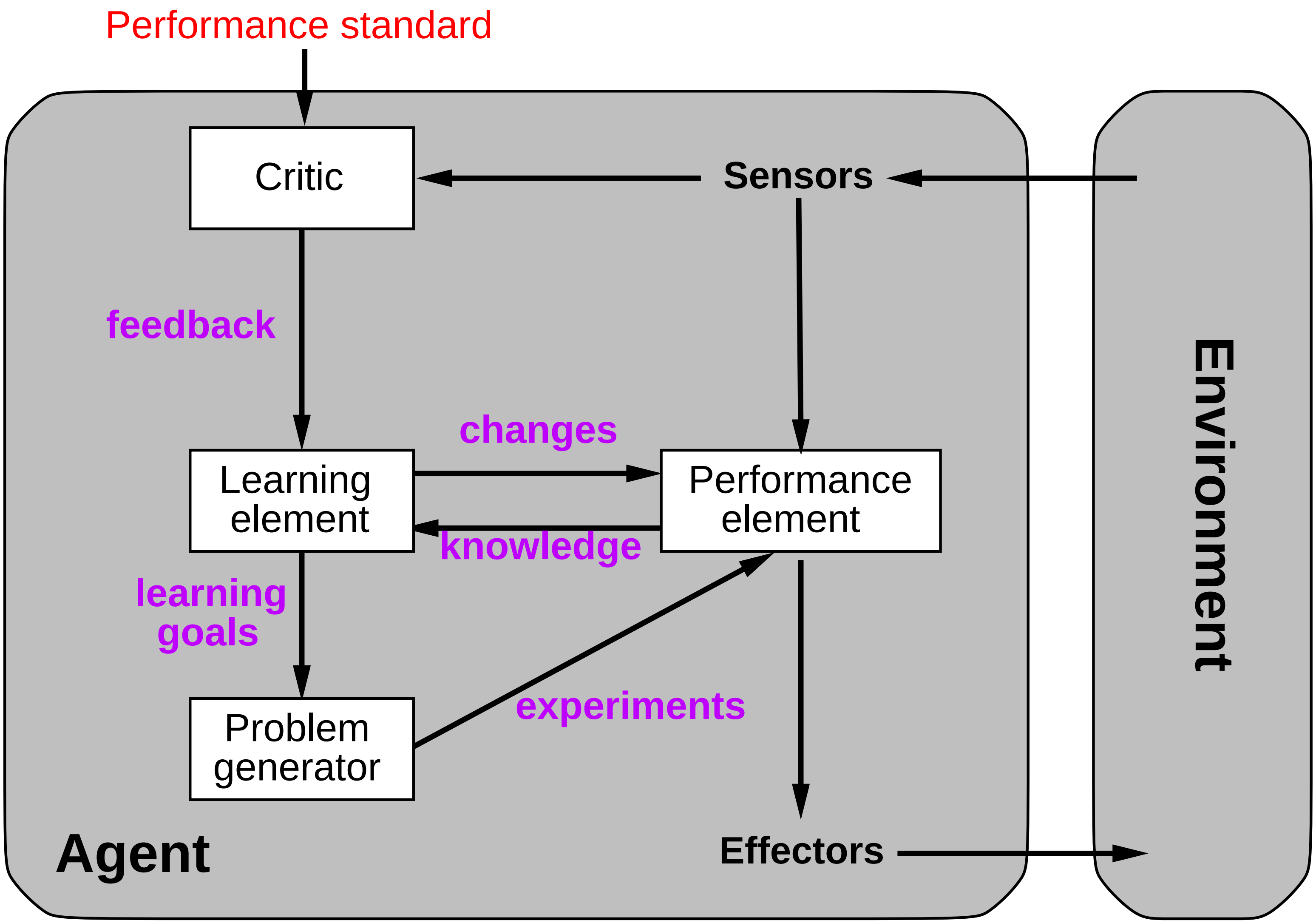
Learning is essential for unknown environments,
i.e., when designer lacks omniscience

Learning is useful as a system construction method,
i.e., expose the agent to reality rather than trying to write it down

Learning modifies the agent's decision mechanisms to improve performance

See Lecture 3 to review the derivation of this scheme

Learning agents



Learning element

Design of learning element is dictated by

- ♢ what type of performance element is used
- ♢ which functional component is to be learned
- ♢ how that functional compoent is represented
- ♢ what kind of feedback is available

Example scenarios:

Performance element	Component	Representation	Feedback
Alpha−beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor−state axioms	Outcome
Utility−based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept−action fn	Neural net	Correct action

Supervised learning: correct answers for each instance
Reinforcement learning: occasional rewards

RL: exploration and exploitation of knowledge learned by repeated trials of maximising the reward.

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., $\begin{array}{c|c|c} O & O & X \\ \hline & X & \\ \hline X & & \end{array}$, $+1$

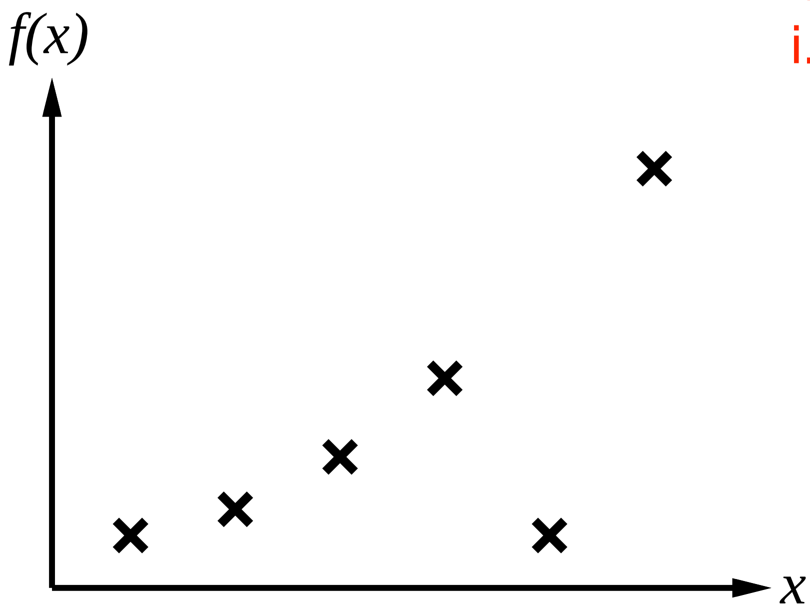
Problem: find a(n) hypothesis h
such that $h \approx f$
given a training set of examples

(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: (or linear regression)



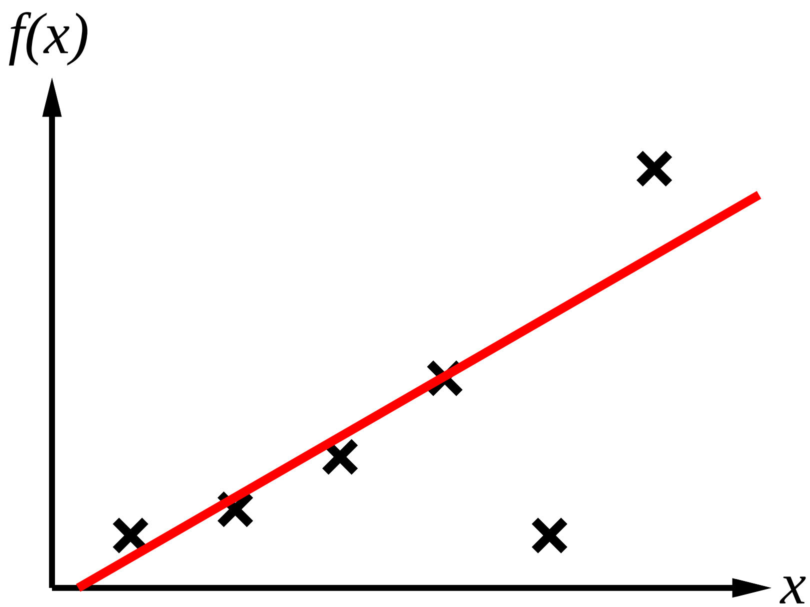
Model of the form y = f(x) + e where x is a vector of attributes, y is a target value and e is an error term based on residuals, i.e. vertical difference between a data point and the model curve

Test the model by suppling a new observation x value and calculating prediction y. If these are close then e is small

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:



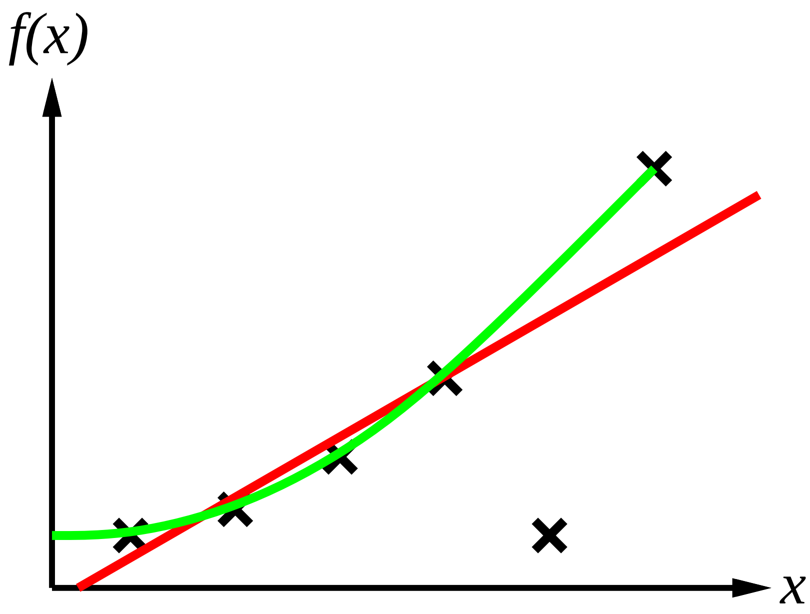
y = mx + c + e
where m is the slope, c is the intercept and e is the error between predictions and observations

e would be based on sum of squared residuals

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:



y is quadratic in x

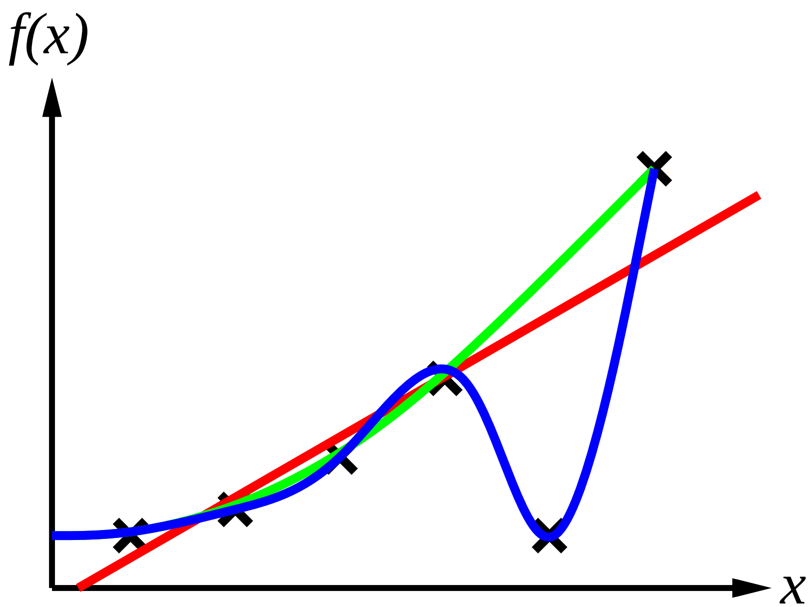
This model captures the overall growth with increasing x, but treats one value as an outlier

Error is coefficient of determination (again)

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:



y is cubic in x

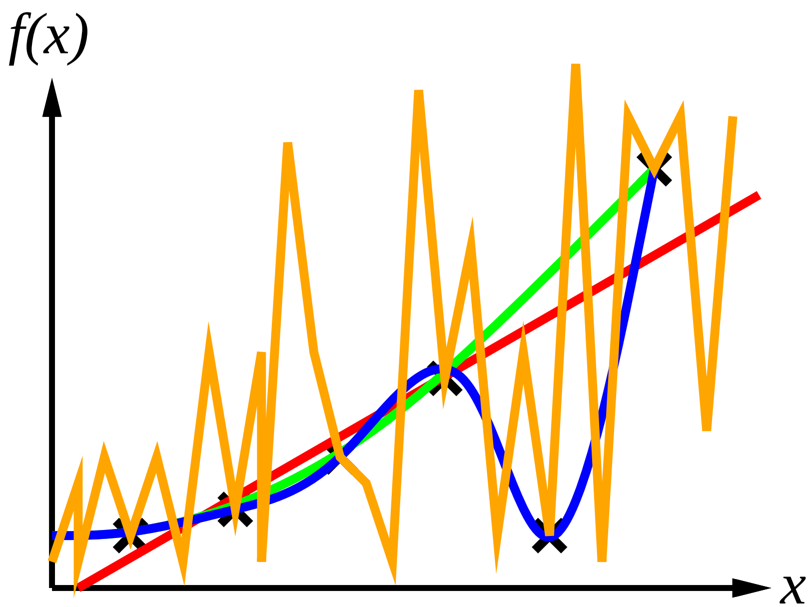
This model is close to all input data and has growth/ decline/growth behaviour

Error is coefficient of determination (again)

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:



y is high degree or even non-smooth in x

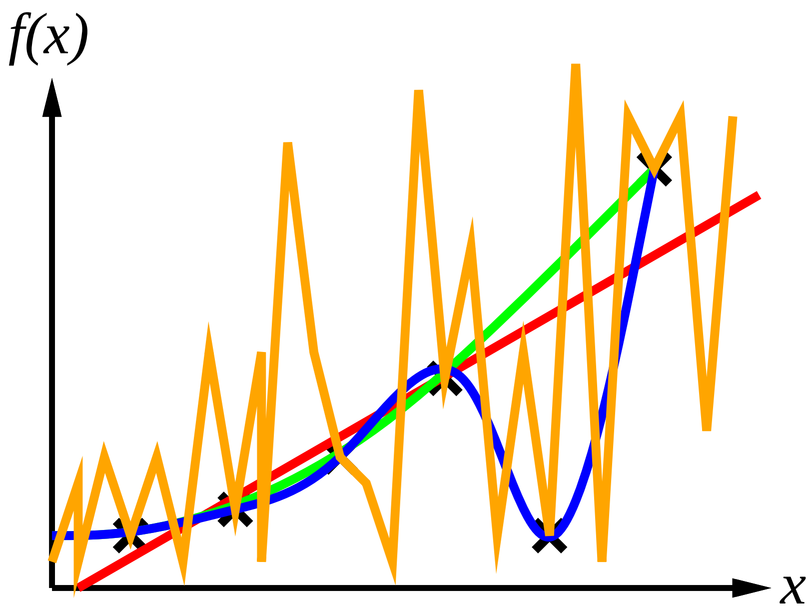
This model is close to all input data but varies wildly between known values

Error is coefficient of determination (again)

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:



Goodness of fit has increased at each stage

Simplicity has decreased at each stage

Which model will have least error when new x values are supplied?
i.e. lowest generalisation error

Ockham's razor: maximize a combination of consistency and simplicity

Attribute = factor = predictor variable = covariate = independent variable = explanatory variable = …

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)

Target = response = dependent variable = outcome variable = …

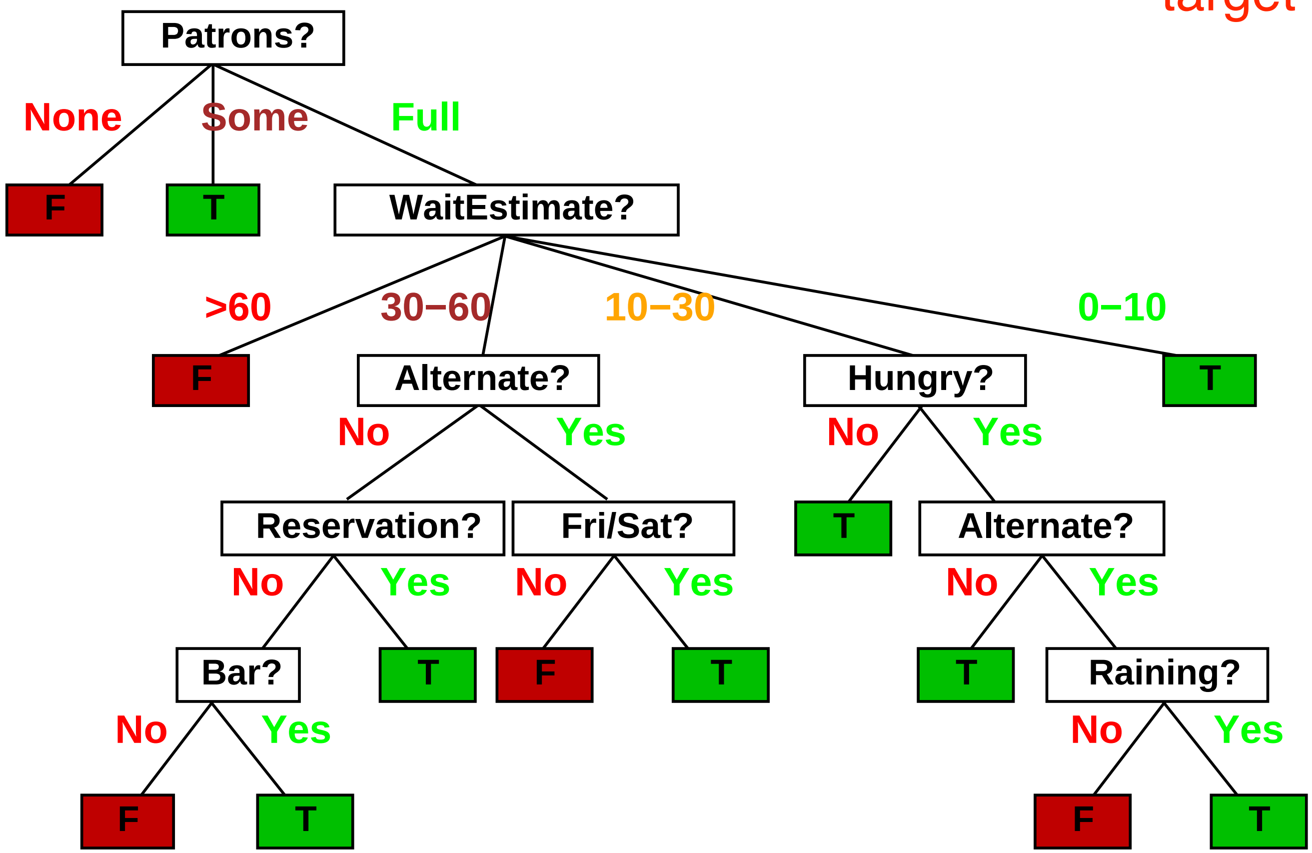
We know the Target for each data instance, hence supervised learning

Some continuous factors have been made discrete. How (if) this is done changes the learning environment

Decision trees

One possible representation for hypotheses

E.g., here is the "true" tree for deciding whether to wait:



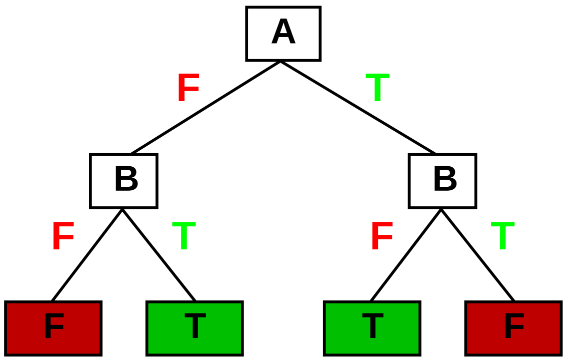
Model of the form y = f(X) + e where X is a matrix of attributes, y is a vector of target values and e is an error term

Test the model by running a new instance down the tree. If the predicted outcome matches the observed outcome, e = 0 for that instance

Expressiveness

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

Prefer to find more **compact** decision trees

Recall: Deterministic means that the current state and a chosen action completely specify the next state.
If not deterministic, then stochastic

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

So generating and testing candidate trees is intractable

We need to restrict the types of tree that we'll consider

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

A conjunction is a Boolean expression of the form A ∧ B

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 as root of (sub)tree

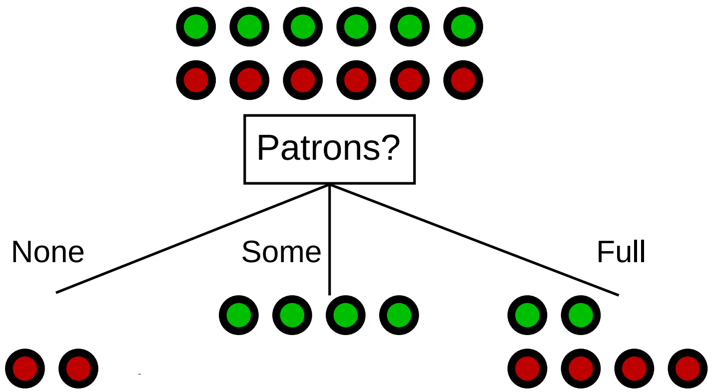
function DTL(*examples, attributes, default*) **returns** a decision tree

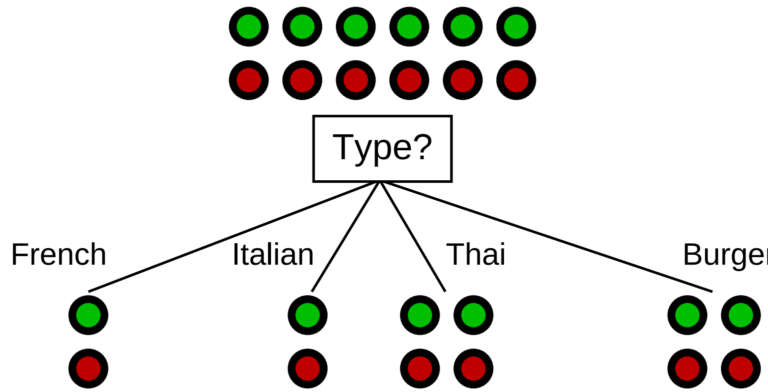
    if *examples* is empty **then return** *default*
    else if all *examples* have the same classification **then return** the classification
    else if *attributes* is empty **then return** MODE(*examples*)
    else
        best ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        tree ← a new decision 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*

Recursive partitioning: the feature space is recursively split into regions containing observations with similar response values.

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

If the name Claude Shannon is new to you, then maybe pause this lecture and look up Information Theory. Shannon is - with Alan Turing - a figure of huge importance in Computer Science log in my annotations is log base 2. Use the change of base formula to convert to base 2 from base 10 or base e

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

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

https://planetcalc.com/2476/

(also called entropy of the prior)

H(0.9,0.1) = -0.9log(0.9) - 0.1log(0.1) = 0.469 bits

H(0.5,0.5) = -0.5log(0.5) - 0.5log(0.5) = 1 bit

H(1,0) = -log(1) - 0log(0) = 0 bits

H(1/3,1/2,1/12,1/12) = (1/3)(1.58) + (1/2)(1) + (2/12)(3.58) = 1.626 bits

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

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

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

For Patrons?: Enone = (2/12)H(0/2,2/2) = 0
Esome = (4/12)H(4/4,0/4) = 0
Efull = (6/12)H(2/6,4/6) = 1/2(0.9813) = 0.459
Sum these three to get 0.459 bits

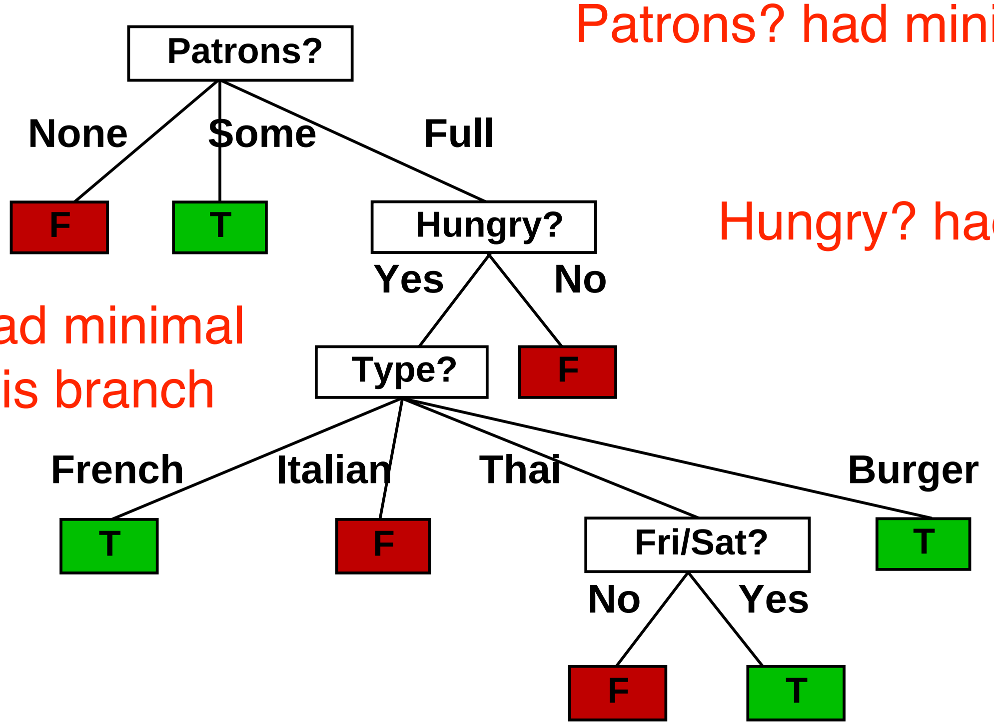
For Type?: H = 1 for each branch,
so 2/12 + 2/12 + 4/12 + 4/12 = 1

0.459 < 1 so choose Patrons?

Example contd.

Decision tree learned from the 12 examples:

Patrons? had minimal E amongst all attributes



Hungry? had minimal E for this branch

Type? had minimal E for this branch

Continue until there is no randomness i.e every leaf node is pure

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

Caveat: this is simpler than a real model for which we grow to a tolerance and then prune to get good generalisation error

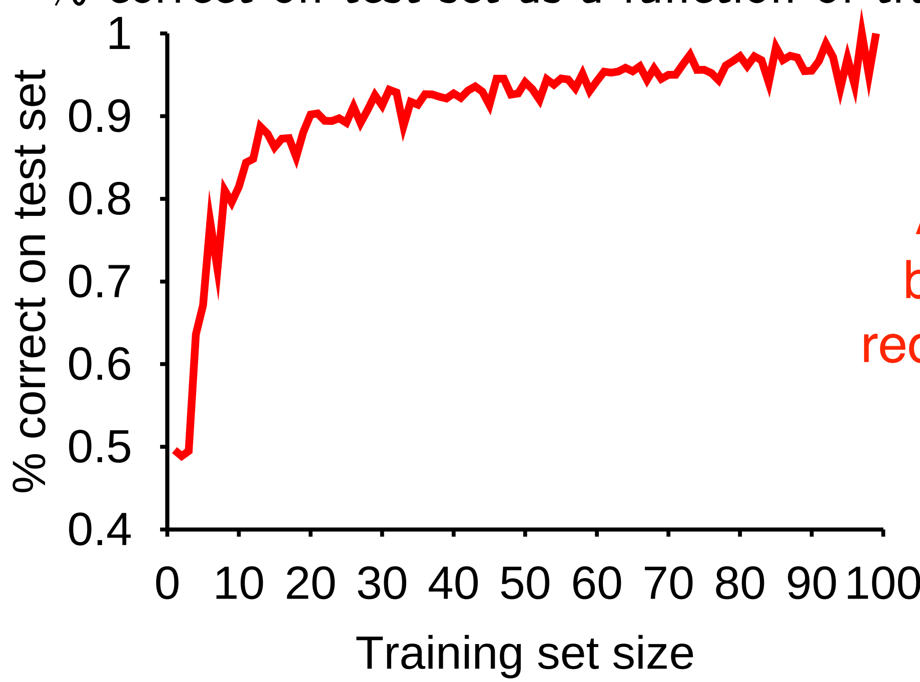
Performance measurement

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 as a function of training set size



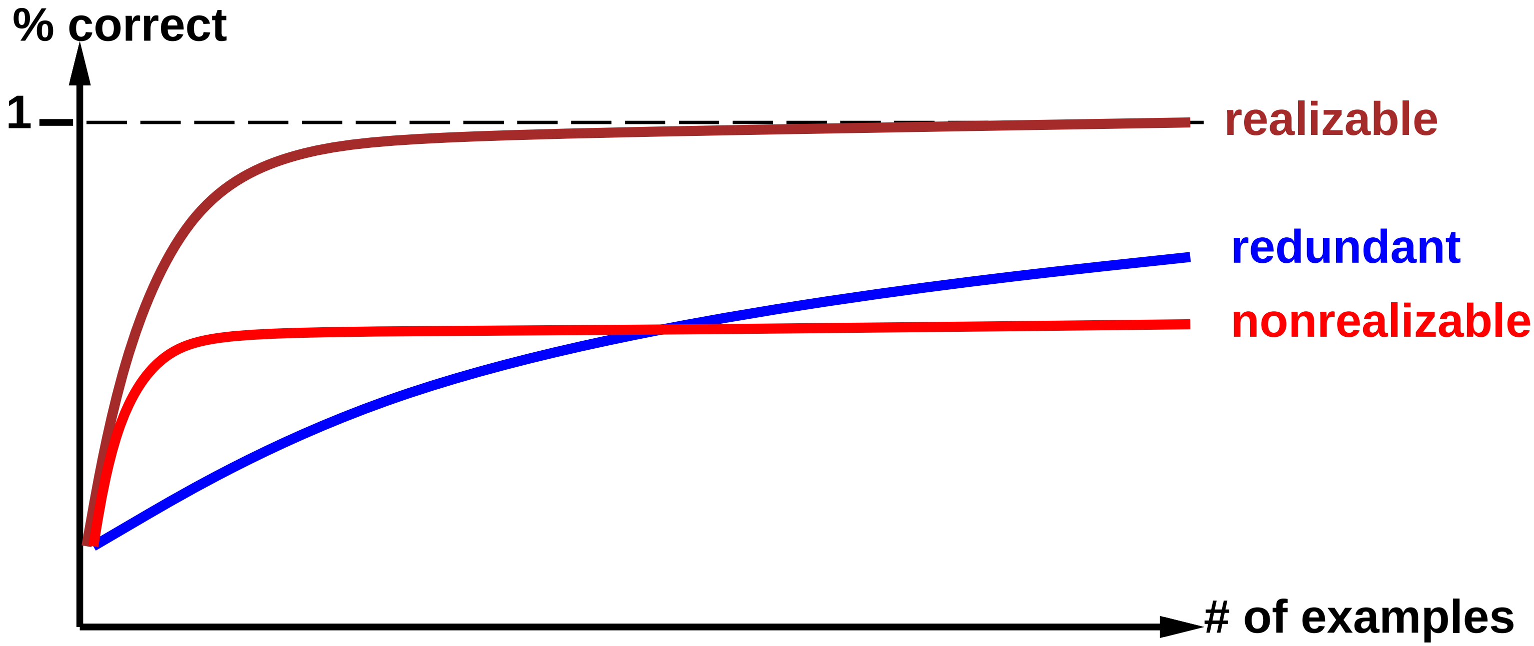
As more high quality data becomes available we can reduce error in our predictions

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 agent = performance element + learning element

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

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

Ex 18.5 (maybe 18.3 in your edition): “Suppose we generate a training set …”

Ex 18.8 (maybe 18.6 in your edition): 3 binary inputs and one binary output