Introduction to Computational intelligence

Learning from examples

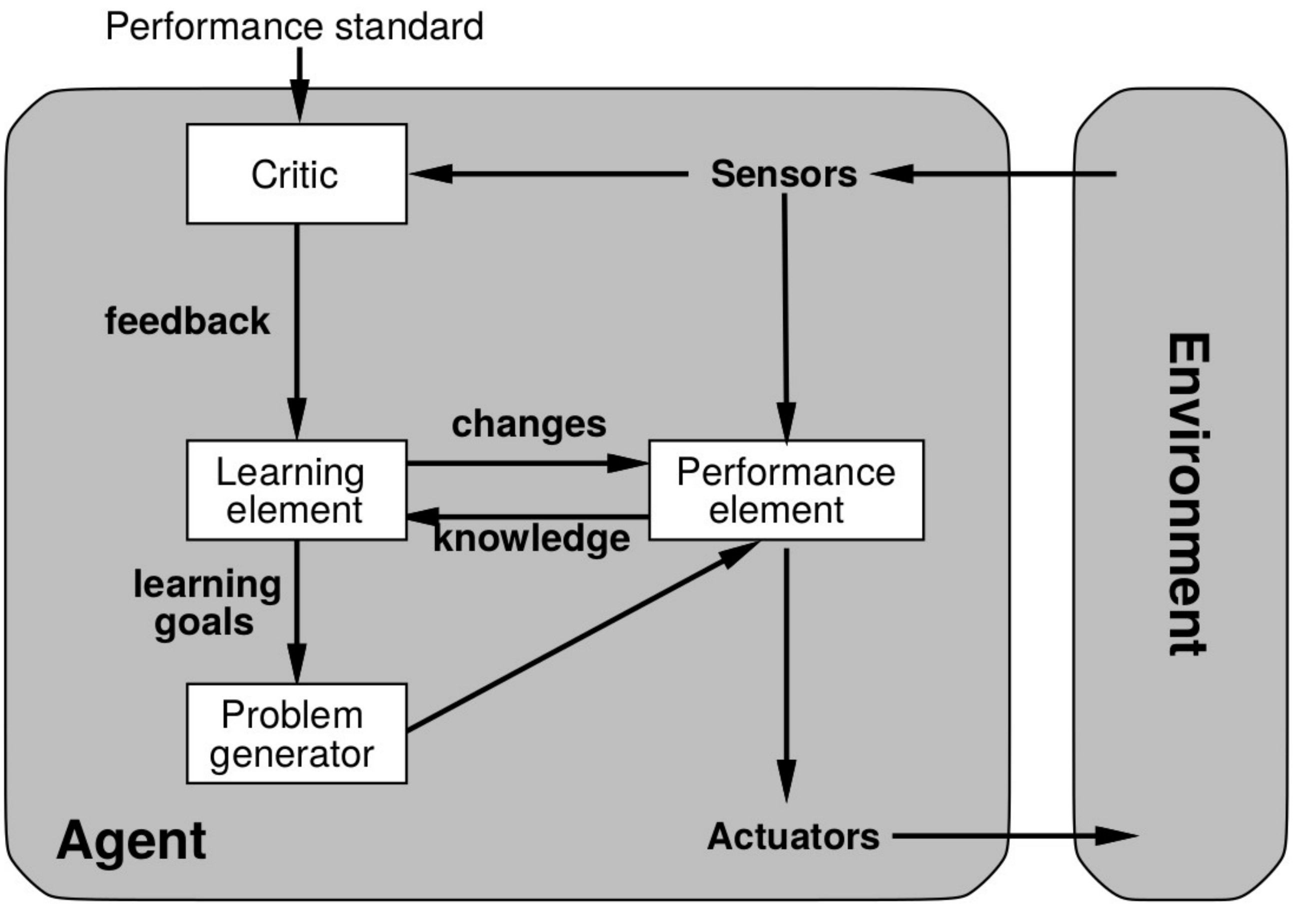
Igor Farkaš
Centre for Cognitive Science
Comenius University in Bratislava

Based on Russel & Norvig: Artificial Intelligence: a Modern Approach, 3rd ed., Prentice Hall, 2010.

Learning agents

- Agent is learning if it improves its performance on future tasks after making observations about the world.
- Why learning? Three main reasons:
  - designers cannot anticipate all possible situations that the agent might find itself in;
  - designers cannot anticipate all changes over time
  - sometimes human programmers have no idea how to program a solution themselves.
- Learning can range from a very simple to a very complex scenario.

Learning agent



Forms of learning

- Any component of an agent can be improved by learning from data.
- Improvements and techniques used to make them depend on four major factors:

  (1) component to be improved, (2) prior knowledge,
  (3) representation of data and learning, (4) feedback from environment.

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

Components (of agents) to be learned

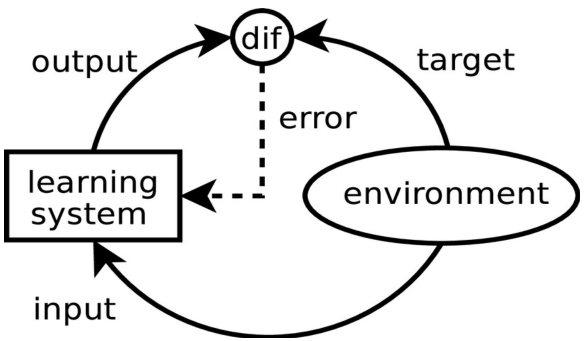
- Direct mapping from conditions on current state to actions.
- A means to infer relevant properties of the world from the percept sequence.
- Information about the way the world evolves and about the results of possible actions the agent can take.
- Utility information indicating the desirability of world states.
- Action-value information indicating the desirability of actions.
- Goals that describe states whose achievement maximizes the agent's utility.

Representation and prior knowledge

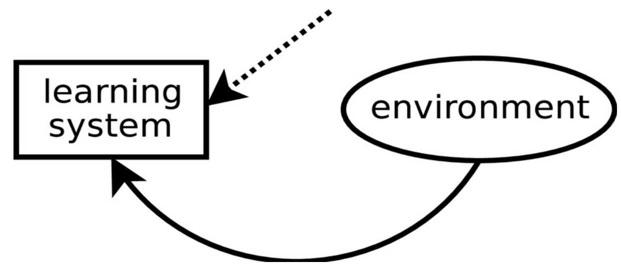
- Examples: propositional logic, first-order logic, Bayesian networks, neural networks… We focus on **factored representation**.

Feedback

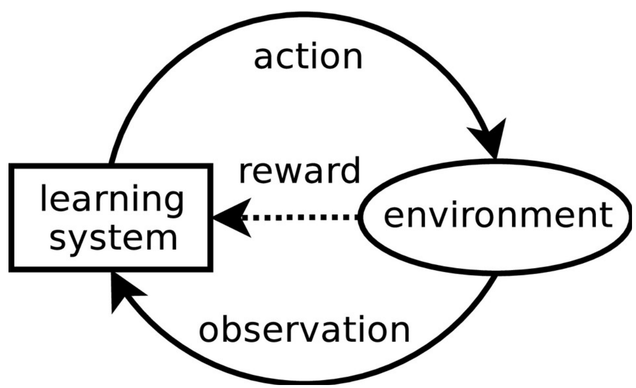
supervised (with teacher)



unsupervised (self-organized)



reinforcement learning



Inductive learning

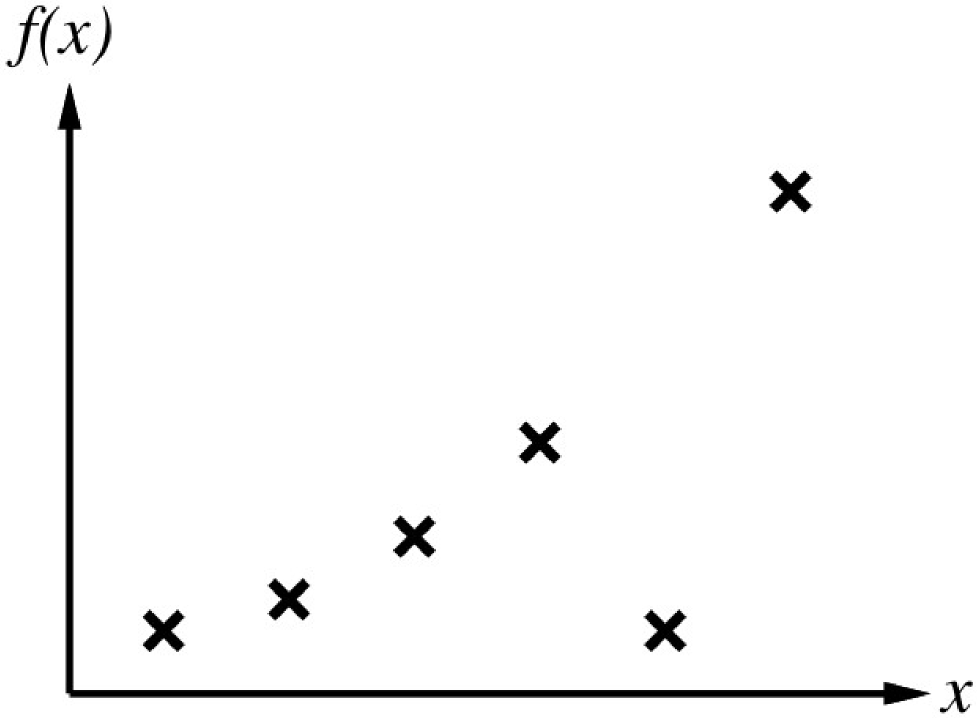
- We focus now on supervised learning
- Example of input-target pair: $\{x, f(x)\}$

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

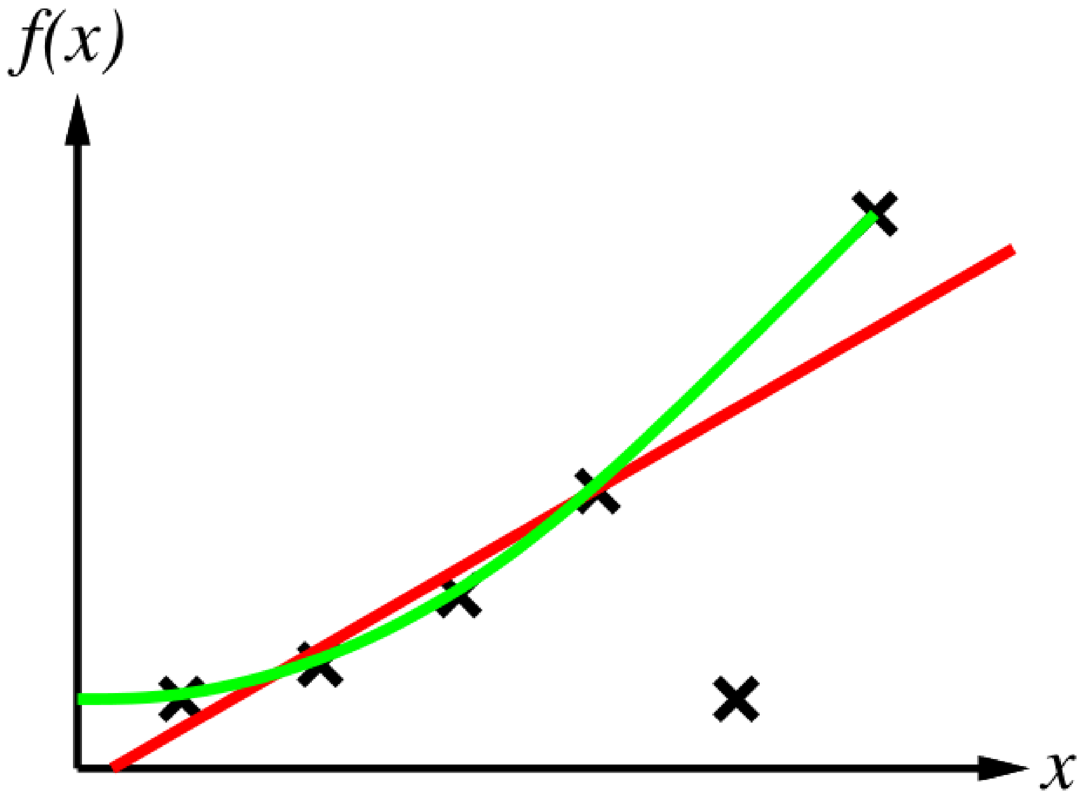
- Assume training set: $\{(x_1,f(x_1)), (x_2,f(x_2)),\ldots,(x_n,f(x_n))\}$
- Problem: find a hypothesis h such that $h \approx f$ given a training set of examples
- Assumptions (simplification of real learning):
  – ignores prior knowledge
  – deterministic, observable environment
  – examples are given
  – the agent wants to learn f (why?)

Example: curve fitting

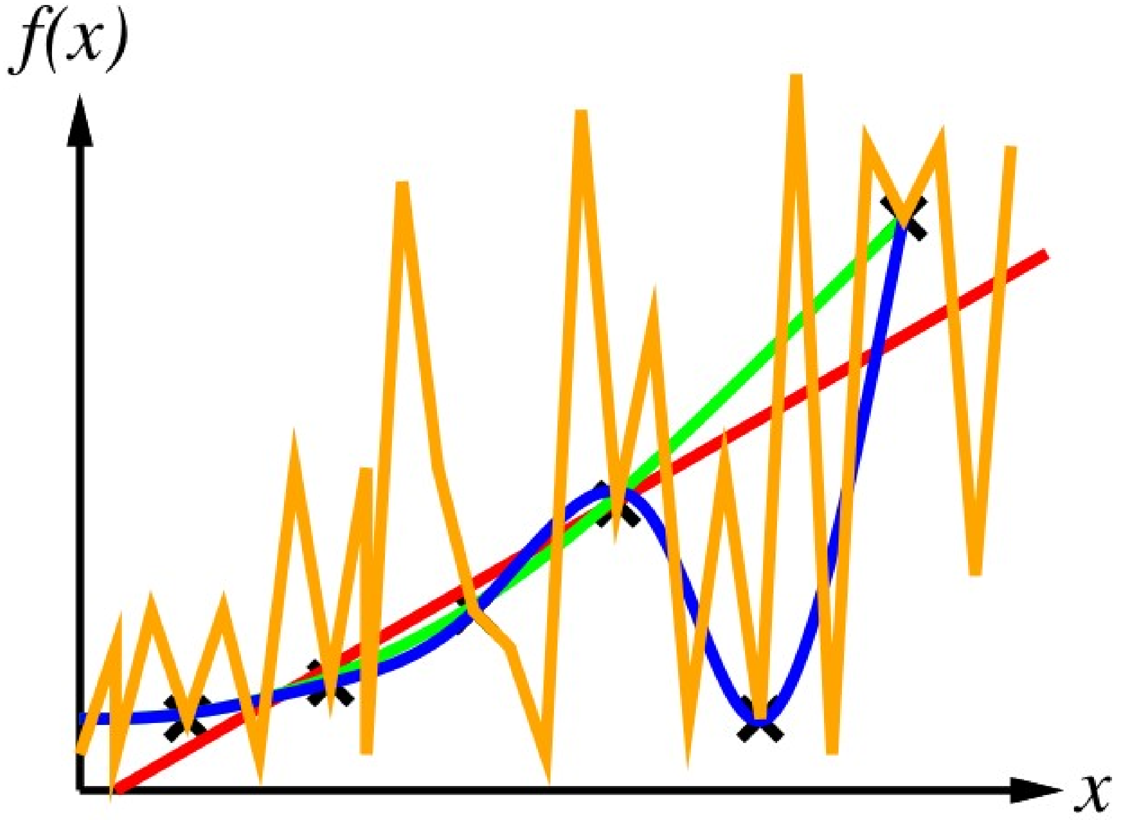
- Construct / adjust *h* to agree with *f* on training set
  (*h* is consistent if it agrees with f on all examples)



Example: curve fitting



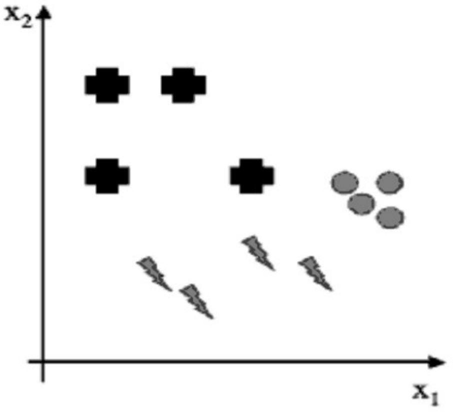
Example: curve fitting

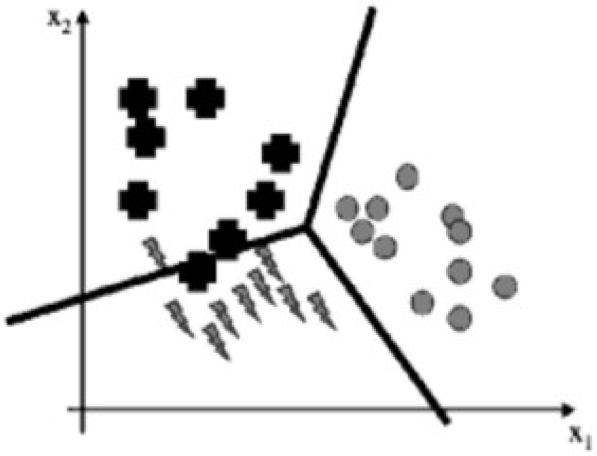


Ockham's razor: maximize a combination of consistency and simplicity

Example: Input classification

x_1	x_2	Class
0.1	1	1
0.15	0.2	2
0.48	0.6	3
0.1	0.6	1
0.2	0.15	2
0.5	0.55	3
0.2	1	1
0.3	0.25	2
0.52	0.6	3
0.3	0.6	1
0.4	0.2	2
0.52	0.5	3





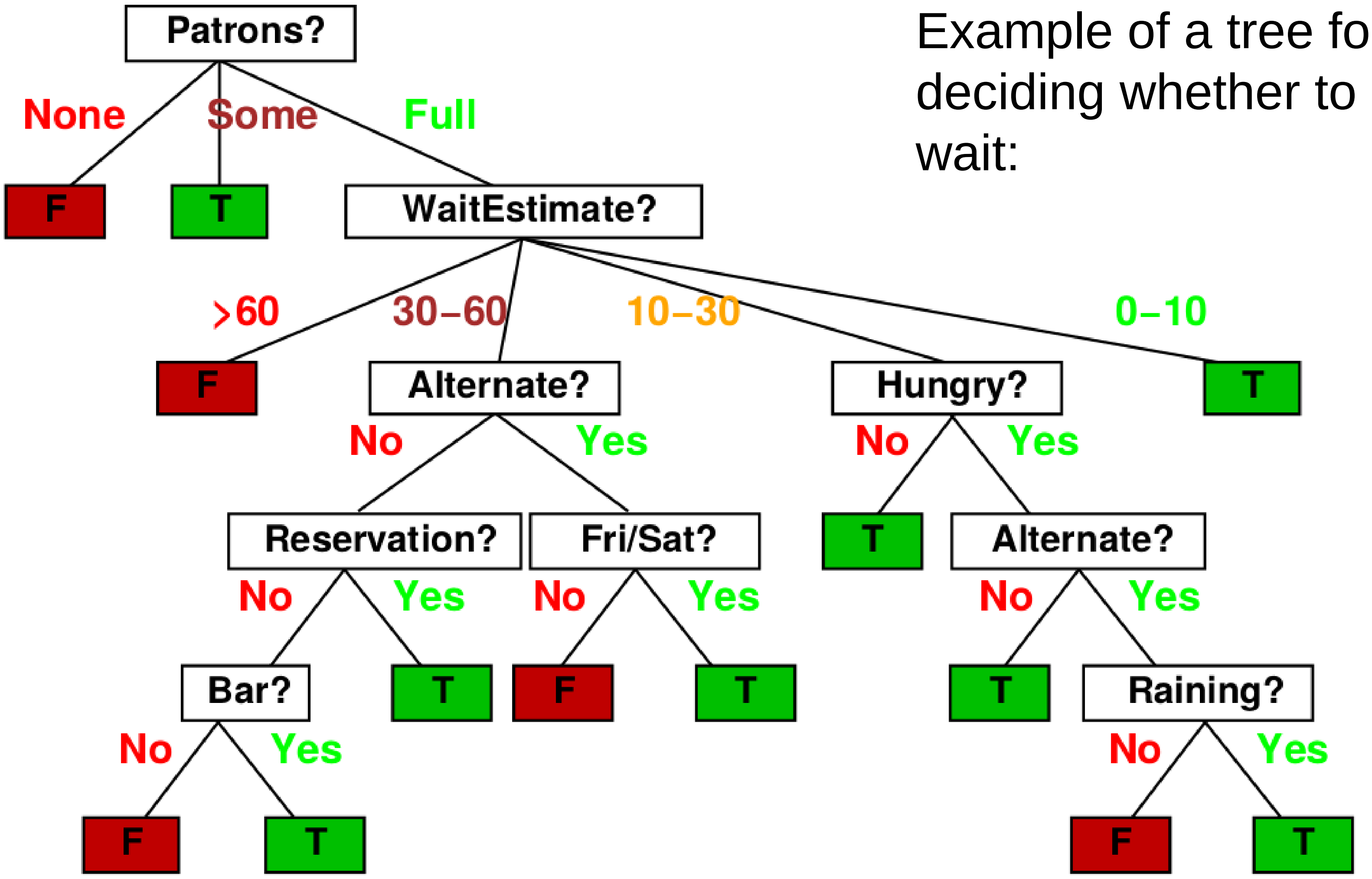
Decision Tree: attribute-based representations

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 (DT)

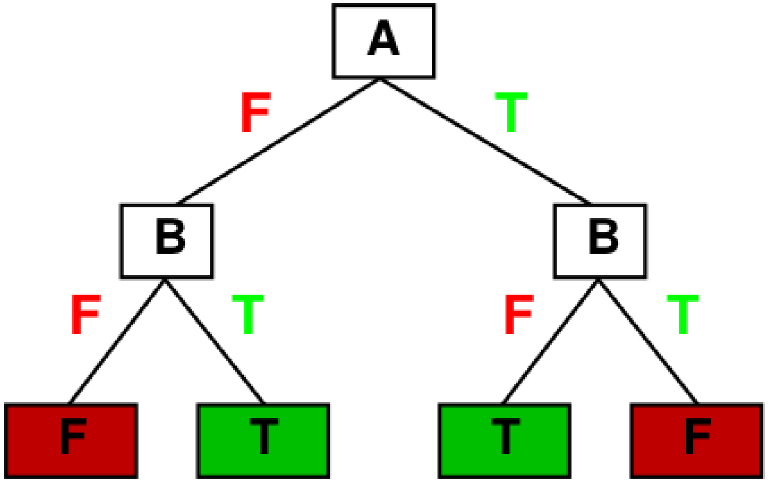
Example of a tree for deciding whether to wait:



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



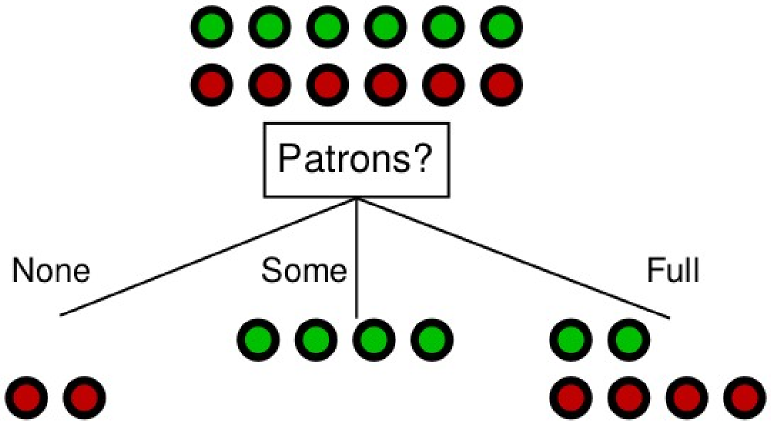
Trivially, there is a consistent DT for any training set with one path to leaf for each example (*f* is deterministic) but it probably won't generalize to new examples.
Prefer to find more compact decision trees.

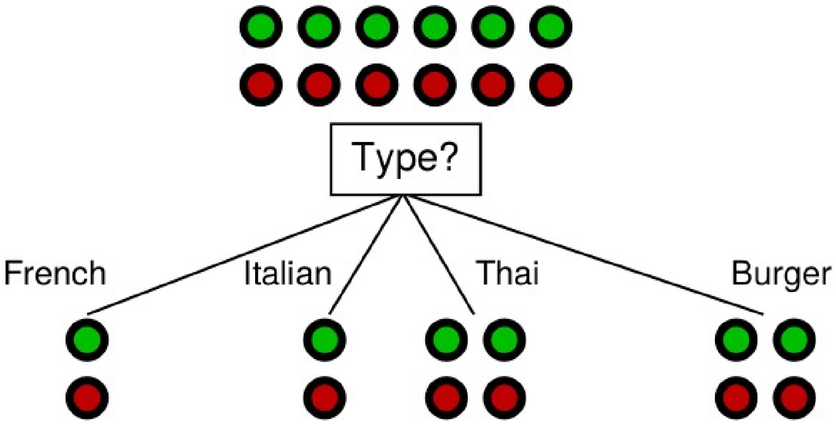
Information and entropy

- Important concept: Information – can be quantified
- 1 bit of information: learning about the outcome of flipping a fair coin
- Acquisition of information (information gain) corresponds to a reduction in entropy.
- Entropy – fundamental quantity in information theory, a measure of uncertainty, or “surprise.” (Shannon & Weaver, 1949)
- How can these concepts be used in building an optimal decision tree?
- There exist many DTs (=> huge hypothesis space)
  - Which one to use?
  - Procedure: always choose the “most significant” attribute as root of (sub)tree.

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”.



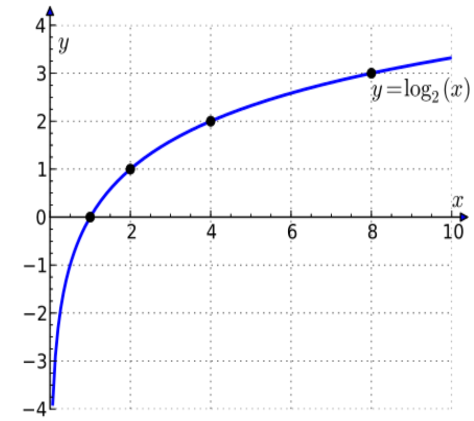


Which attribute is better (i.e. provides more information about the decision)?

More on entropy

- Entropy (*H*) of a random variable *V* with possible values v_i, each with probability $P(v_i)$, for *i* = 1,2,…,*n,* is defined as

$$H(V) = -\sum_{i=1}^{n} P(v_i)\log_2(P(v_i))$$



- *H* can be interpreted as the average quantity of information, or "surprise", inherent to the variable's possible outcomes.
- *H*(fair-coin) = – 0.5*log2(0.5) – 0.5*log2(0.5) = 1 bit
- *IG*(tail) = *IG*(head) = – 1*log2(0.5) = 1 bit
- For unfair coin, e.g. *P*(head) = 0.3 => *P*(tail) = 0.7: *H* = 0.880, and
- Information gain after each observation:

  IG(head) = – 1*log2(0.3) = 1.737

  IG(tail) = – 1*log2(0.7) = 0.514

Entropy in decision tree task

- Let's define $B(q)$ as the entropy of a Boolean random variable that is true with probability q:

  $$B(q) = -(q \log_2(q) + (1-q) \log_2(1-q))$$

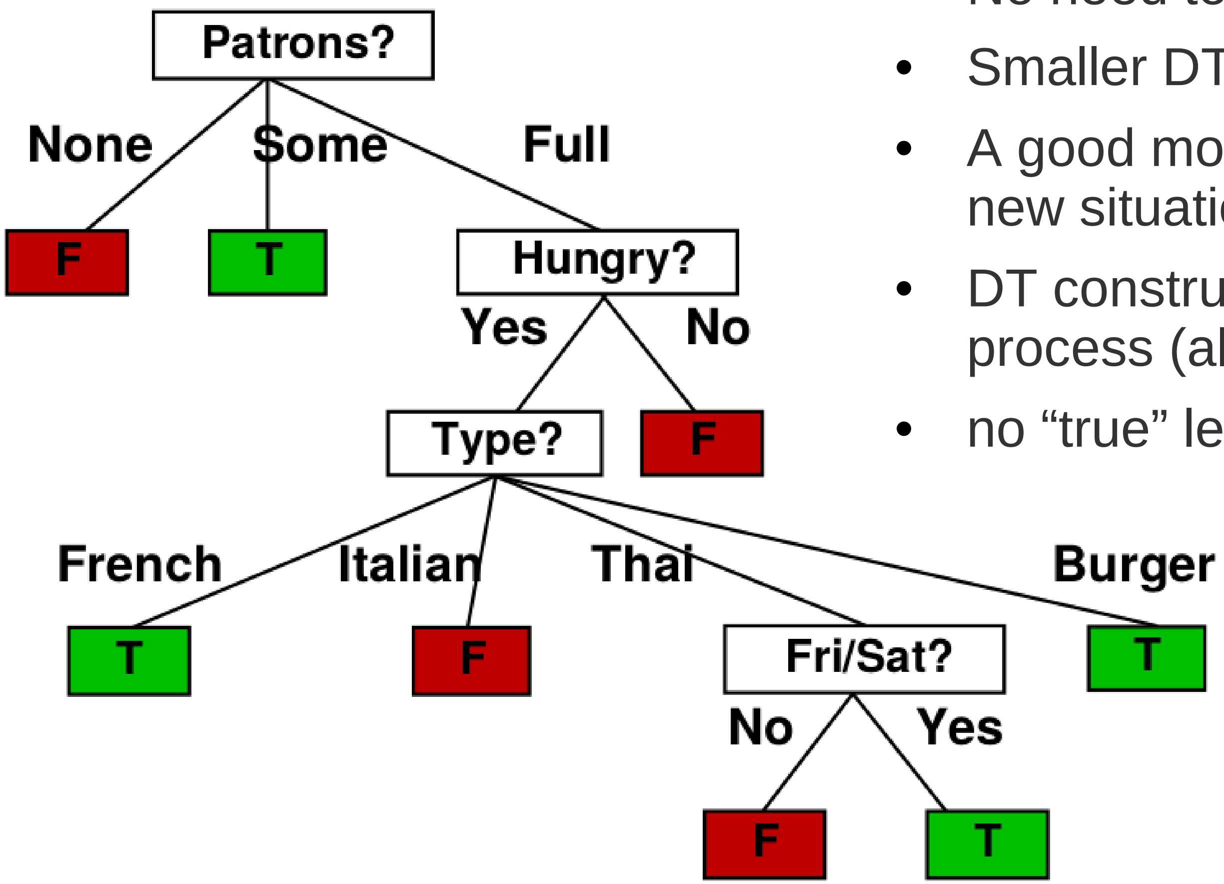
- Suppose we have p positive and n negative examples at the root
- $\Rightarrow B(p/(p+n))$ bits needed to classify a new example

  e.g., for 12 restaurant examples, $p = n = 6$, so we need 1 bit
- Information gain from attribute A = the reduction of entropy (B) about correct classification:

  $$\mathrm{IG}(A) = B(p/(p+n)) - \mathrm{Remainder}(A)$$

  e.g. IG(*Patrons*) ≈ 0.541 bit; IG(*Type*) = 0 bit
- So observing *Patrons* is more informative, since the entropy is reduced to only 0.459 bit.

Decision tree learned from 12 examples



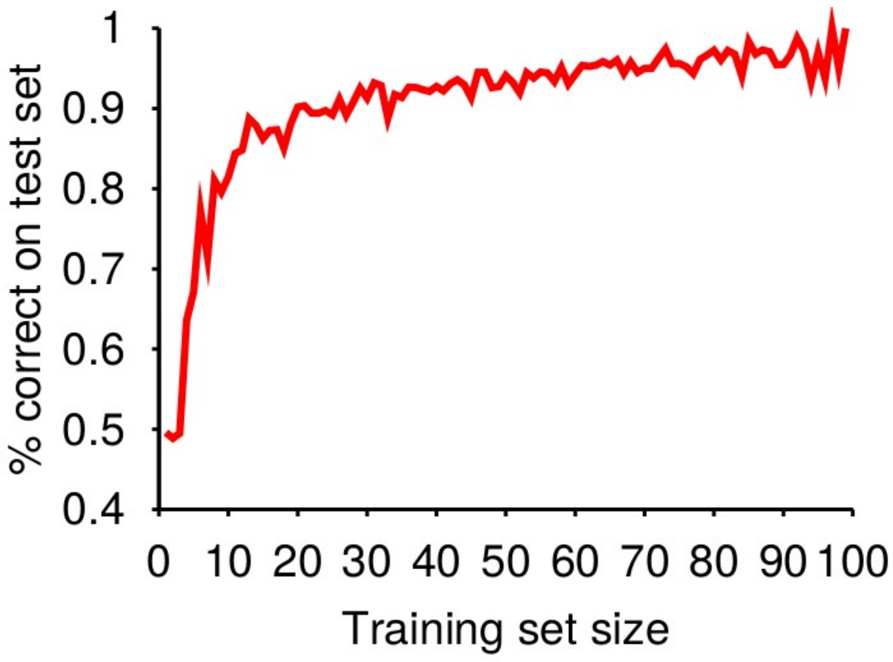
- No need to look at all attributes
- Smaller DT = simpler model
- A good model should generalize to new situations (set of attributes)
- DT construction is a deterministic process (algorithm)
- no “true” learning involved :-(

Substantially simpler than the previous example – a more complex hypothesis isn’t justified by small amount of data.

Performance measurement

- How do we know that *h* ≈ *f* ?
- We would need to test our DT in new situations
- We try *h* on a new test set of examples (with the same distribution over example space as training set)

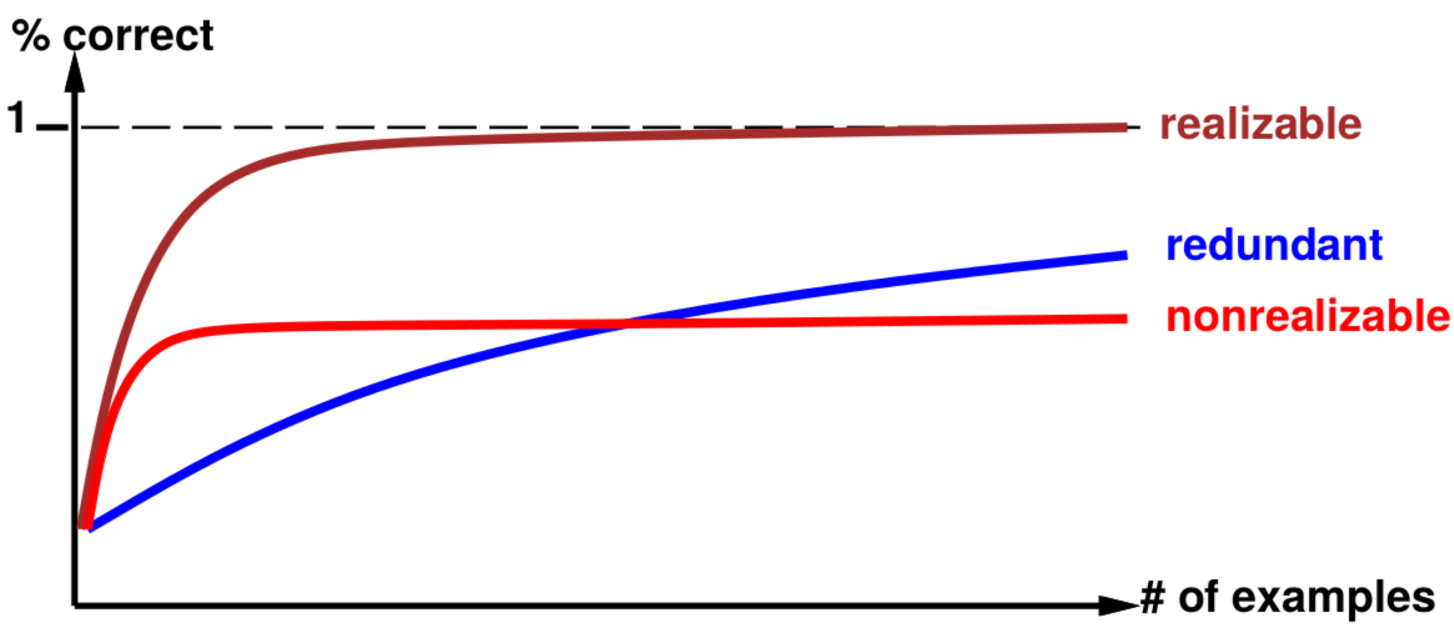
- The more training data we have, the more accurate model we can get.
- The accuracy of the model also depend on its complexity.



Performance measurement (ctd)

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)

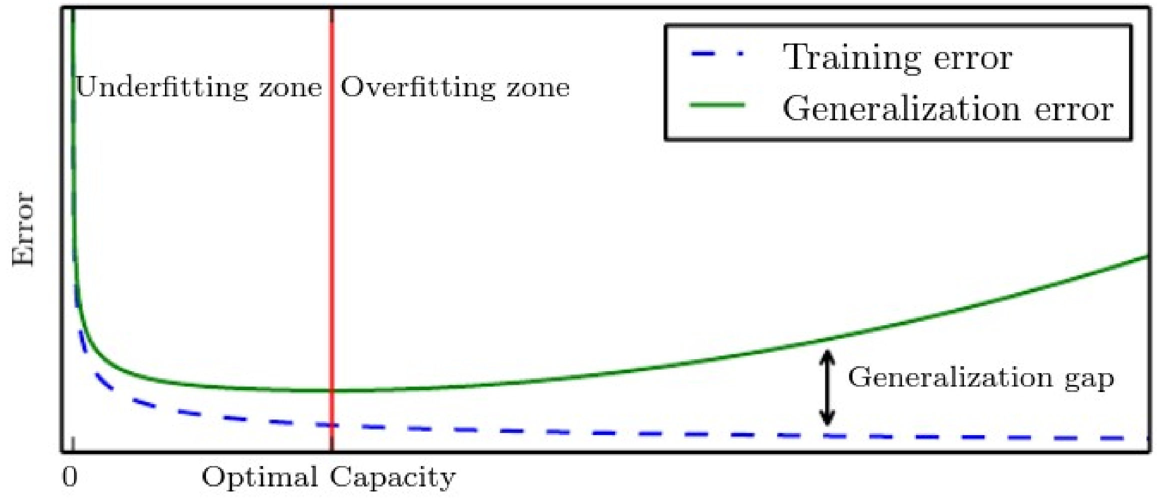


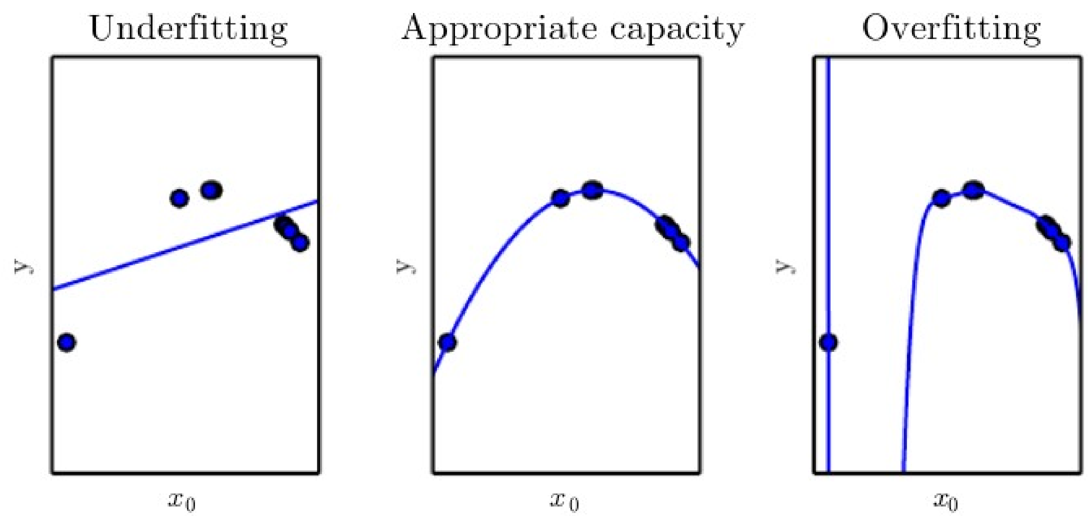
Generalization

Data set:

$$A = A_{estim} \cup A_{val} \cup A_{test}$$

- Validation set is used for model selection.
- Generalization (assessed first on validation set) is important



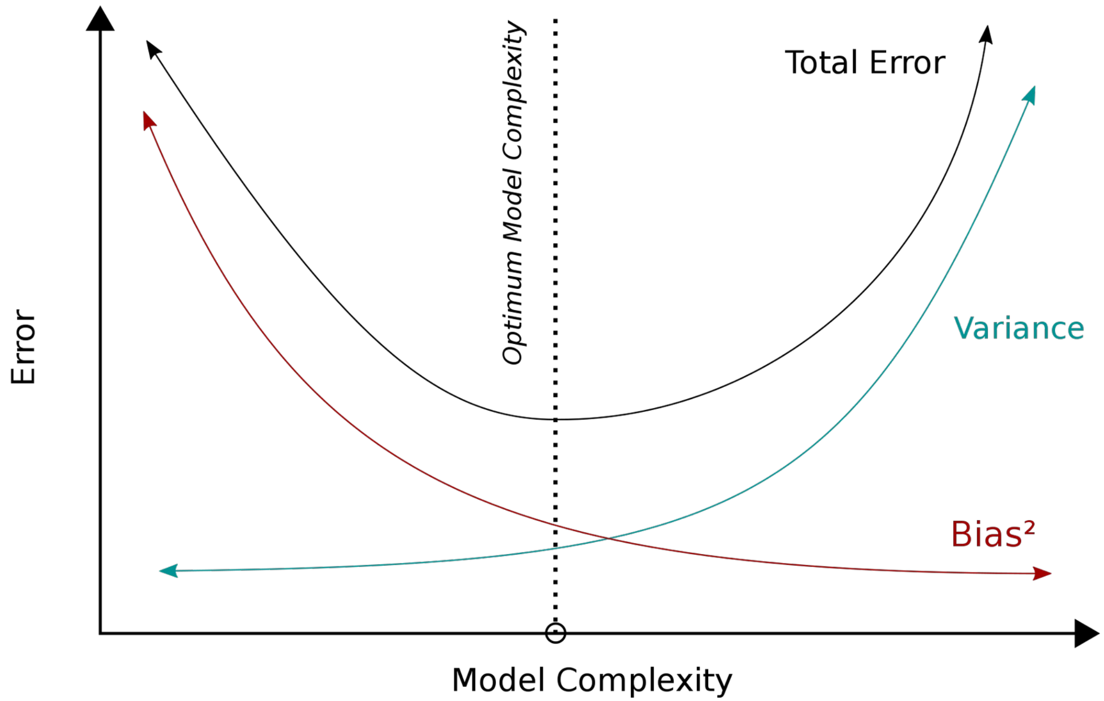


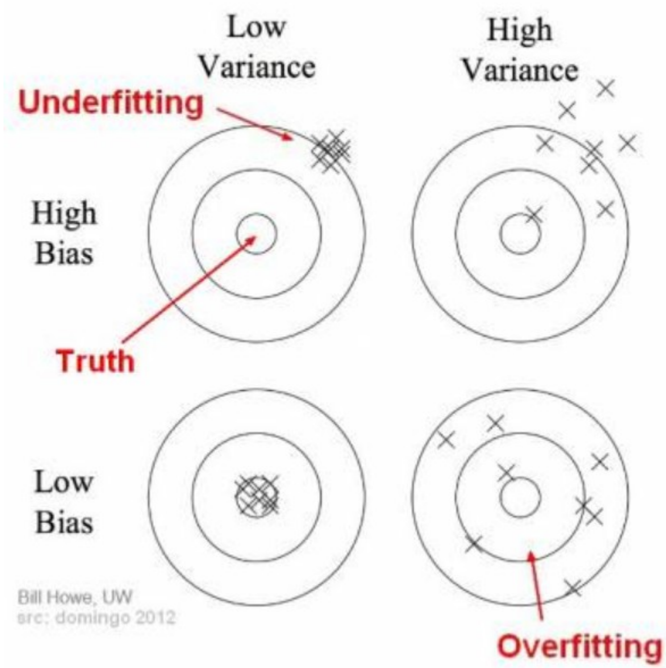
Generalization depends on:

- size of A_{estim} and its representativeness
- architecture of NN
- task complexity

(Goodfellow et al., 2016)

Bias–variance tradeoff





https://en.wikipedia.org/wiki/Bias-variance_tradeoff

$$\mathrm{E}\left[\left(y-\hat{f}(x)\right)^2\right] = \mathrm{Bias}\left[\hat{f}(x)\right]^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2 \qquad y = f(x) + \epsilon$$

$$\mathrm{Bias}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x) - f(x)\right] \qquad \mathrm{Var}\left[\hat{f}(x)\right] = \mathrm{E}[\hat{f}(x)^2] - \mathrm{E}[\hat{f}(x)]^2$$

Summary

- Learning needed for unknown environments
- 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 is based on maximizing information gain.
- Learning performance = prediction accuracy measured on test set
- Good generalization = performance on test set (is crucial) in machine learning.