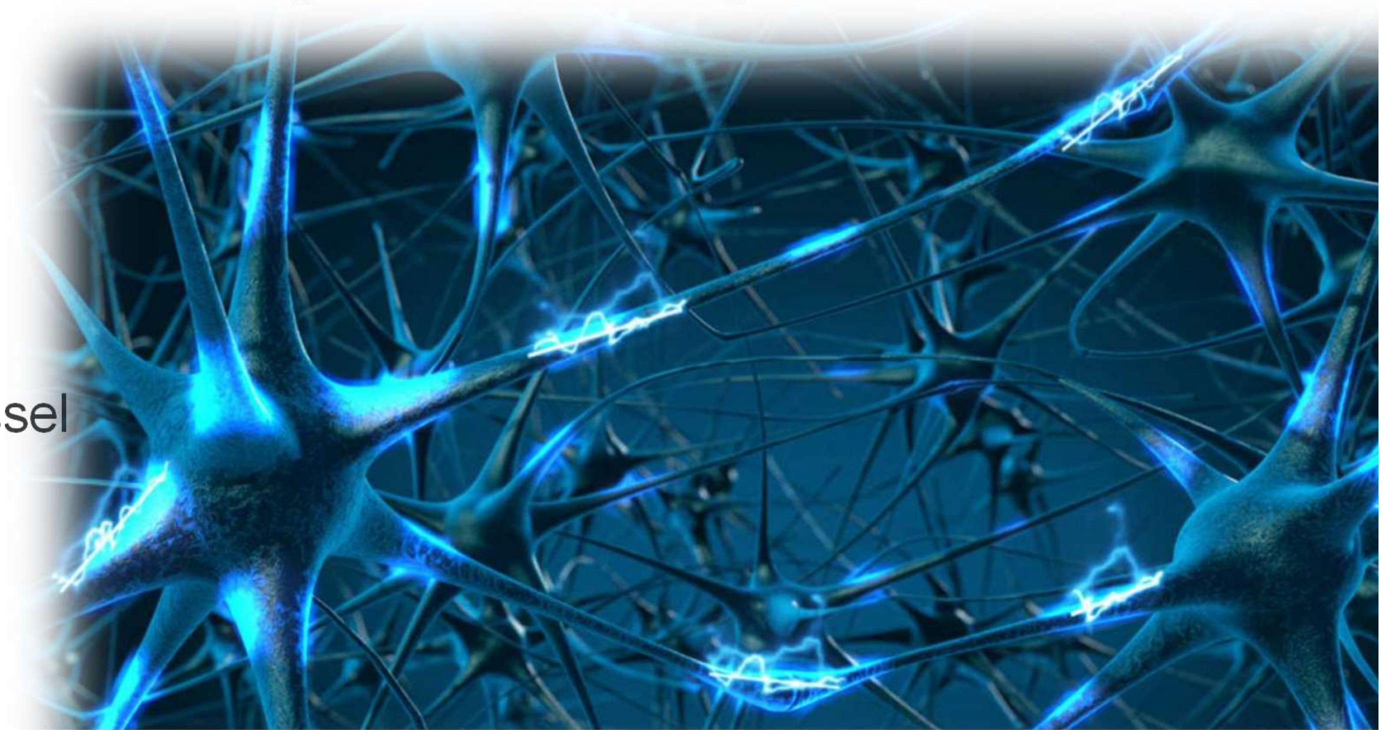
INTELLIGENZA ARTIFICIALE

APPRENDIMENTO AUTOMATICO DA ESEMPI

Corsi di Laurea in Informatica, Ing. Gestionale, Ing. Informatica, Ing. di Internet
(a.a. 2021-2022)

Roberto Basili

(*) dalle *slides* di S. Russel



Overview (AIMA chpt. 18.1-18.4)

- Agents & machine learning
- Learning from examples:
  - Complexity and Expressiveness
  - The definition of model selection
- An example: **Decision Tree learning**
  - Recursive search among Boolean formulas
  - Attribute Selection in DT: Information Gain
- Learning methodology: design, experiment/ evaluation and model selection
  - Cross validation

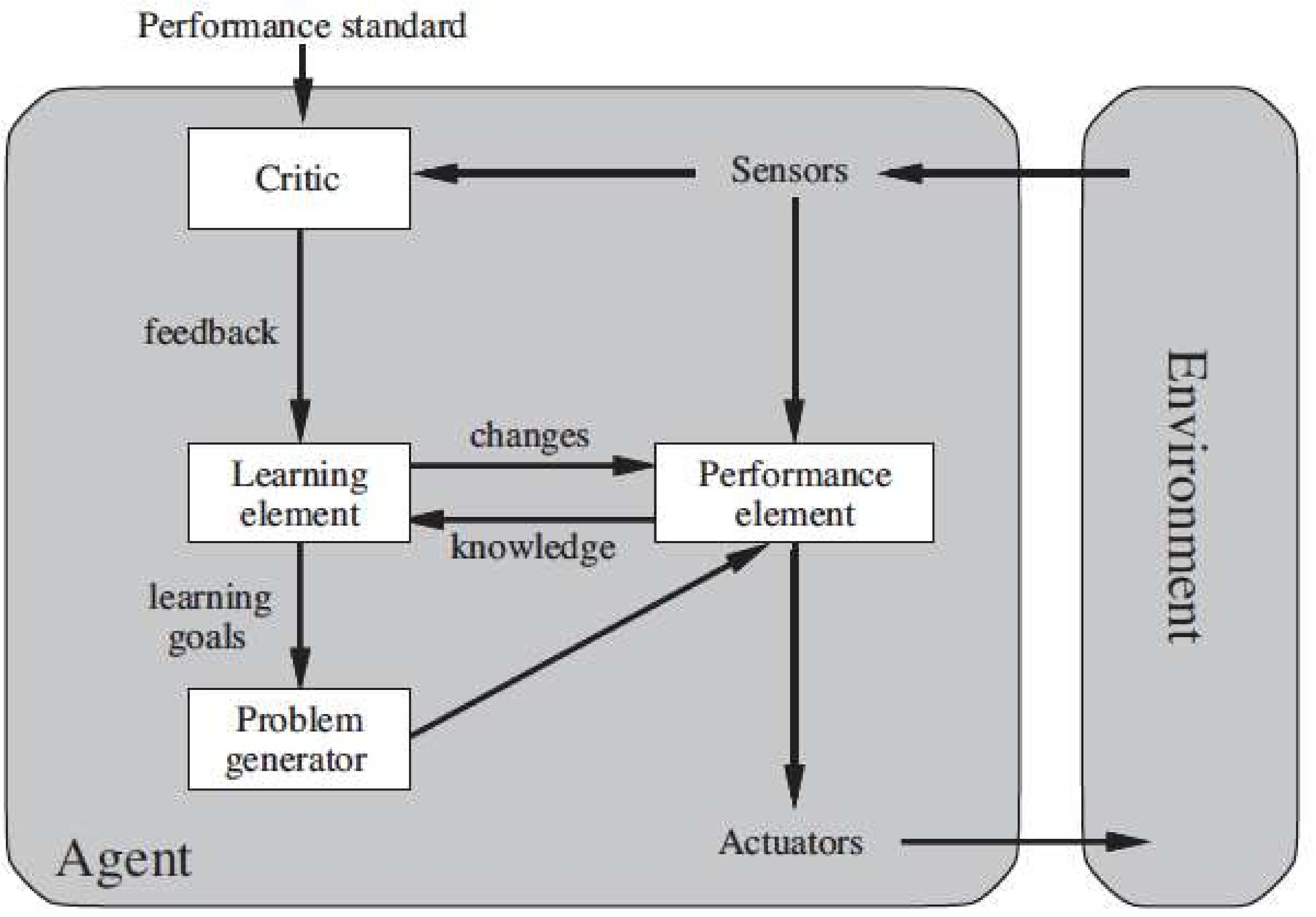
Introduction to machine learning

- Introduction to machine learning
  - When appropriate and when not appropriate
  - Task definition

  - Learning methodology: design, experiment, evaluation
  - Learning issues: representing hypothesis

  - Learning paradigms
    - Supervised learning
    - Unsupervised learning
    - Reinforcement learning

AIMA learning architecture



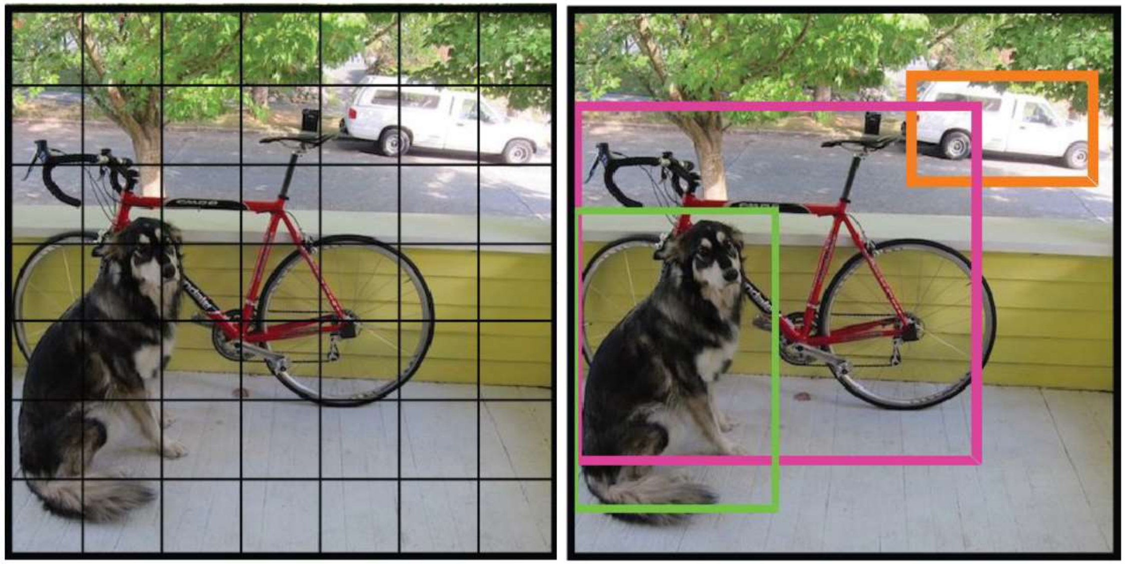
Machine learning: definition

- *A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E* [Mitchell]

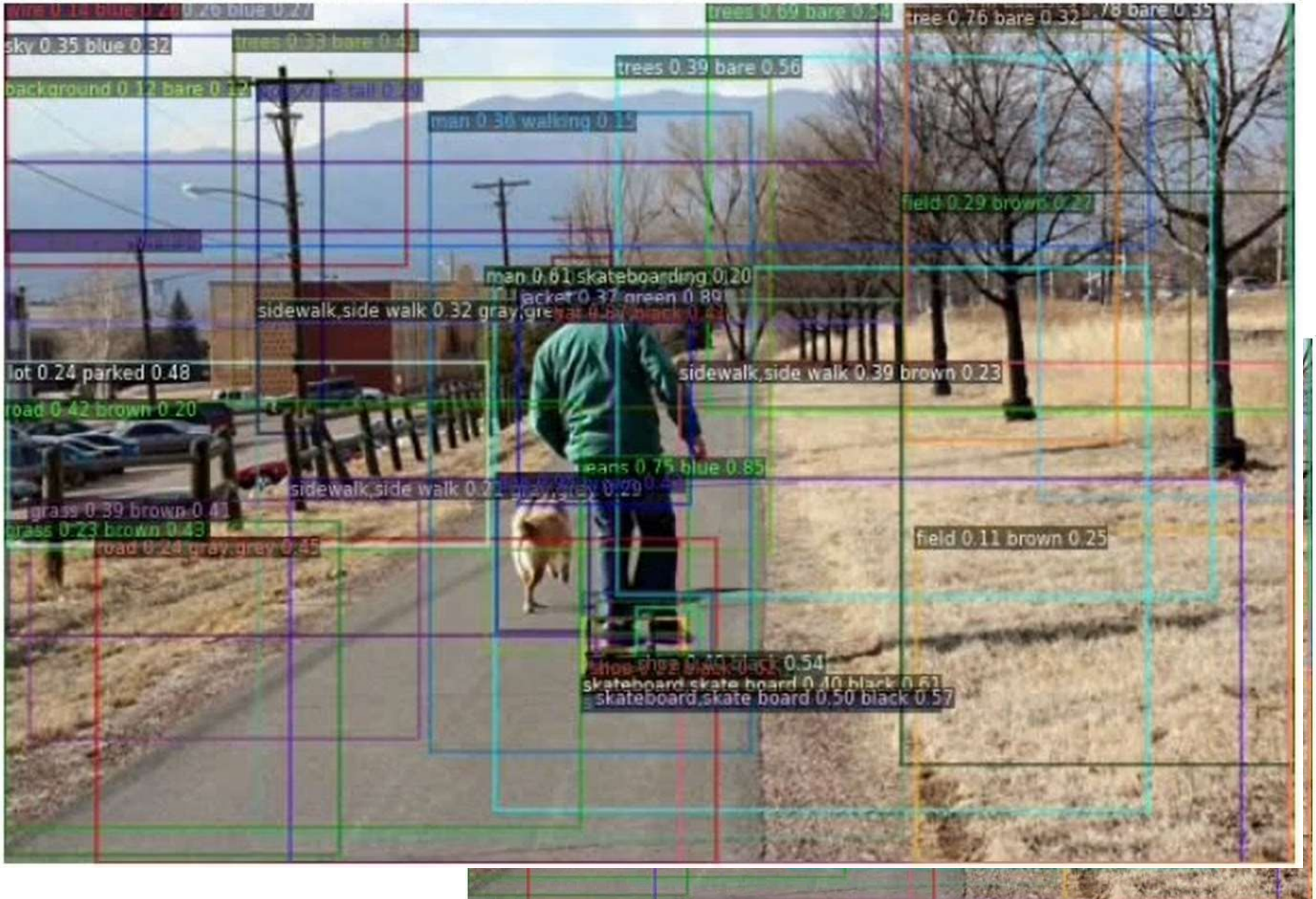
- Problem definition for a learning agent
  - Task T
  - Performance measure P
  - Experience E

Machine learning: definition

- *A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E* [Mitchell]
- Examples of learning agents (1):
  - Task1 T1: image classification, Performance P1: rate of recognized objects in images, Experience E1: annotated images



Machine learning: definition



document-classifier-3.webp

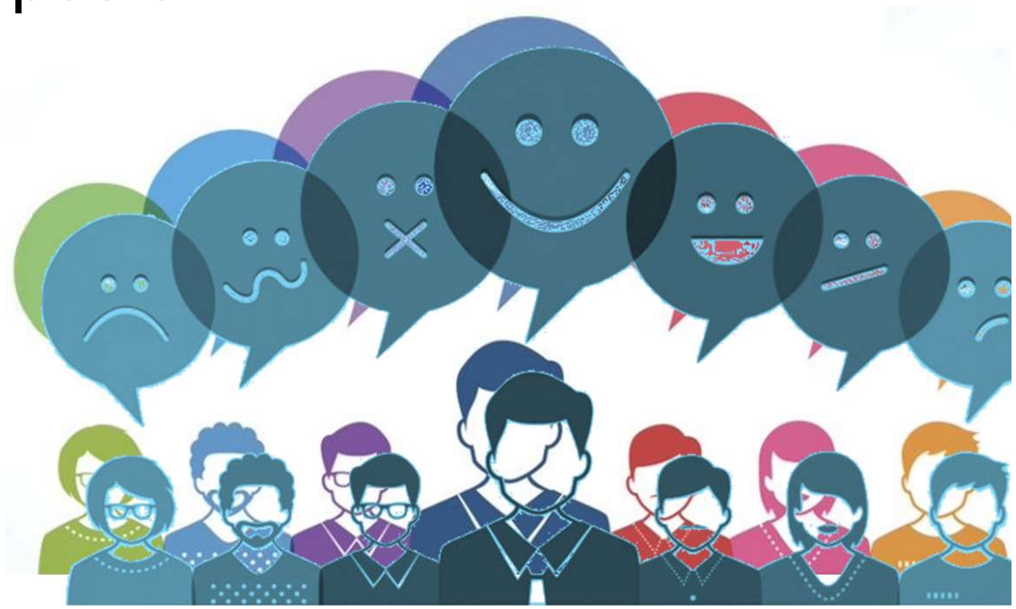
Machine learning: definition

- *A computer program is said to* ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E* [Mitchell]
- Examples of learning agents (2):
  - Task2 T2: news classification,
    Performance P1: rate of correctly classified news items,
    Experience E1: categorized news

document-classifier-3.webp

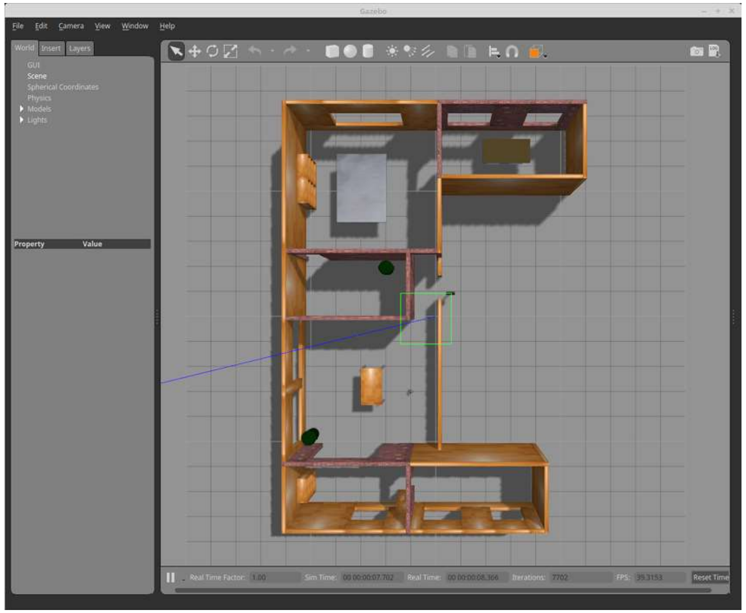
Machine learning: definition

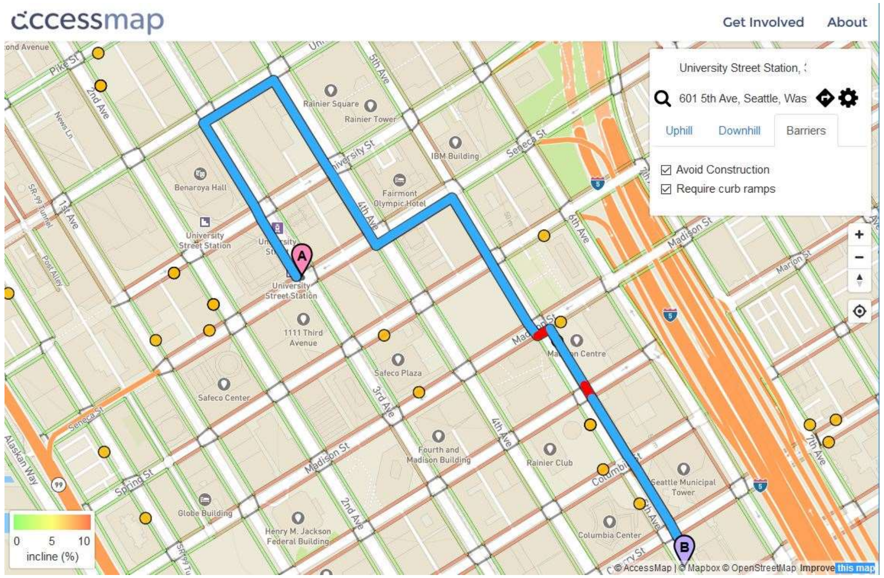
- *A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E* [Mitchell]
- Examples of learning agents (3):
  - Task2 T2: (social) sentiment analysis,
  - Performance P1: %recognized posts in sentiment classes
  - Experience E1: categorized posts



Machine learning: definition

- *A computer program is said to* ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E* [Mitchell]
- Examples of learning agents (3):
  - Task2 T2: route finding,
  - Performance P1: time to target
  - Experience E1: perfect routes and/or examples of precomputed heuristics at branching steps





Designing a learning system

1. Choosing the training experience
   - Examples of best moves, games outcome …
2. Choosing the target function
   - board-move, board-value, …
3. Choosing a representation for the target function
   - linear function with weights (hypothesis space)
4. Choosing a learning algorithm for approximating the target function
   - A method for parameter estimation

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
  such that $h \approx f$
  given a training set of examples

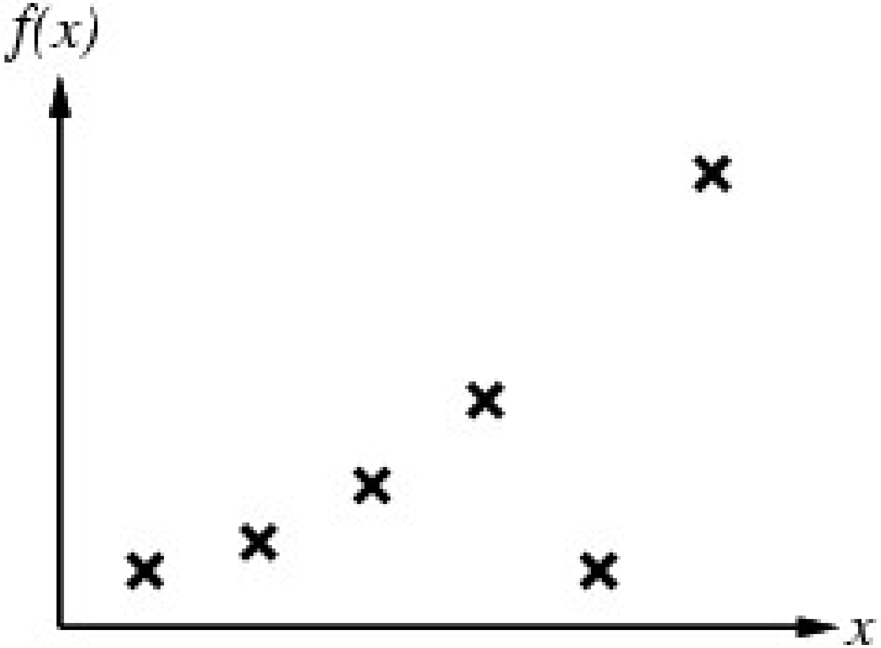
  (This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes examples are given)

Inductive learning method

- Construct/adjust *h* to agree with *f* on training set

(*h* is consistent if it agrees with *f* on all examples)

e.g., curve fitting:

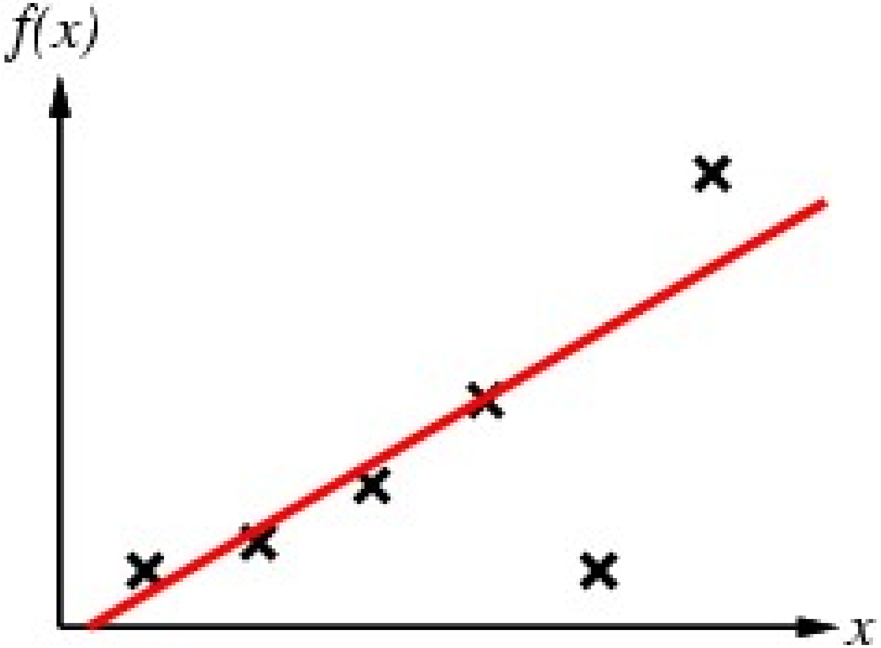


Inductive learning method

- 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:

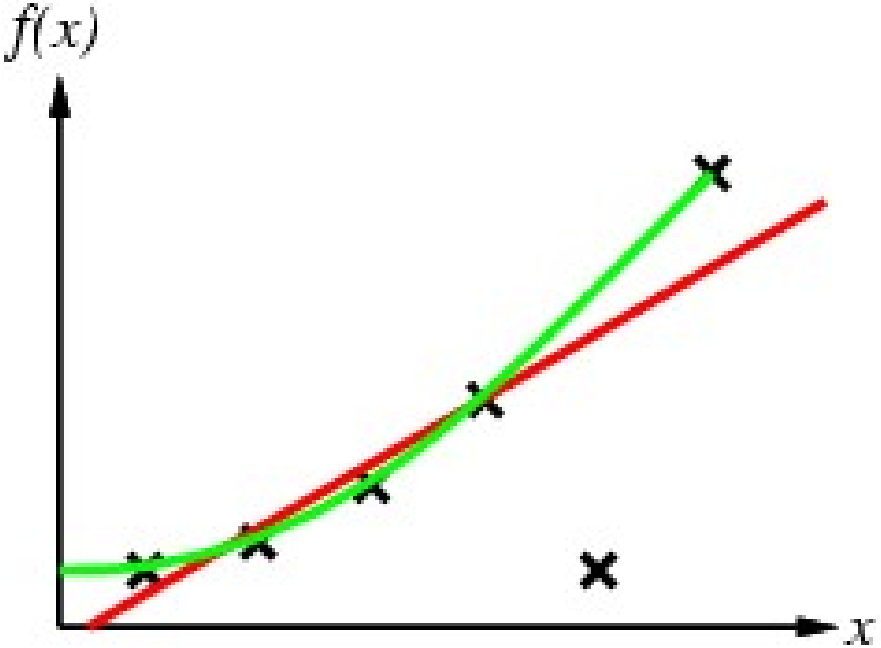


Inductive learning method

- 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:

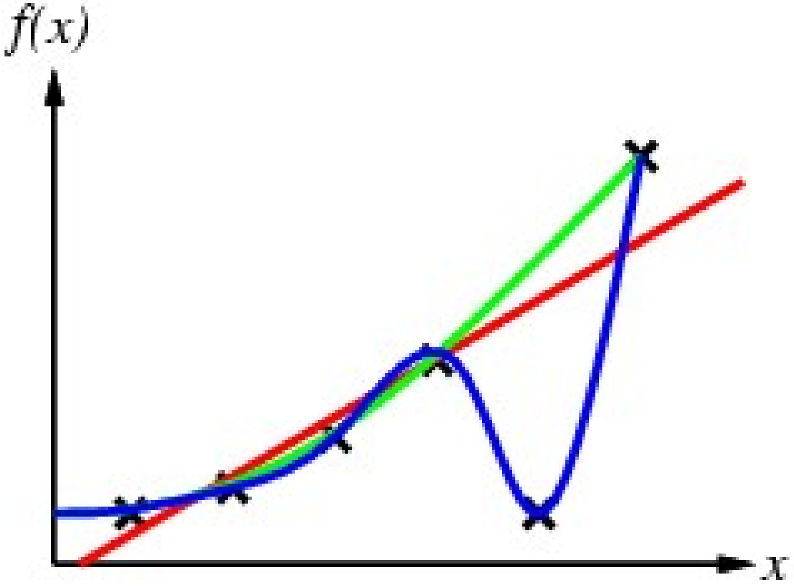


Inductive learning method

- 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:

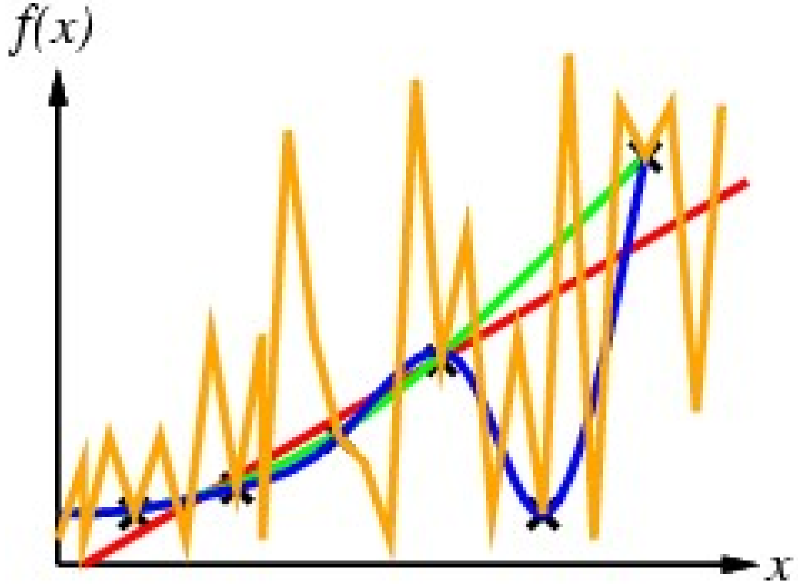


Inductive learning method

- Construct/adjust *h* to agree with *f* on training set

(*h* is consistent if it agrees with *f* on all examples)

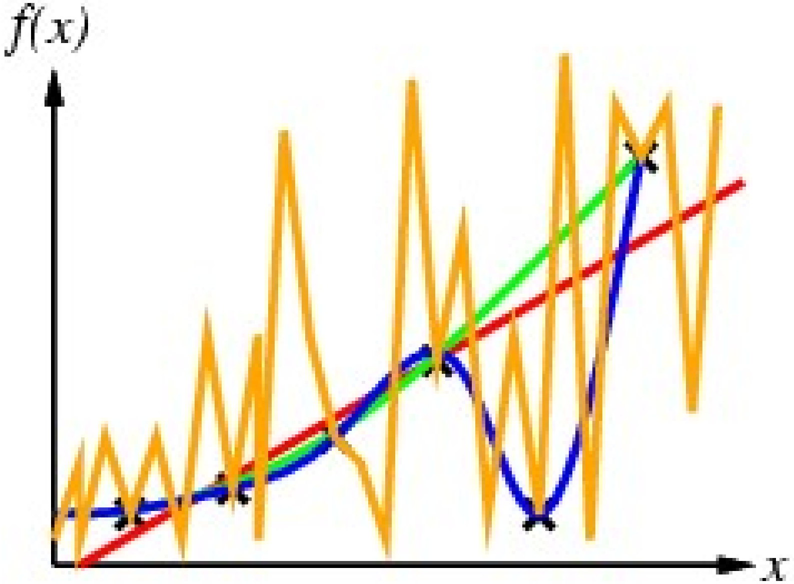
e.g., curve fitting:

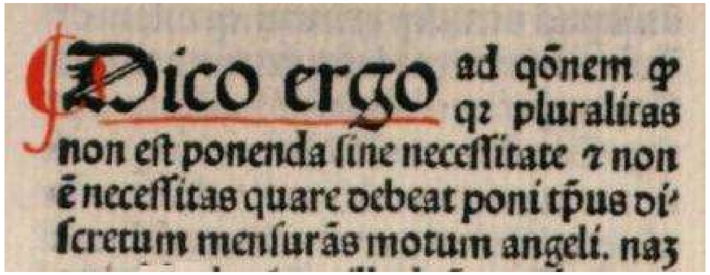


Inductive learning method

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

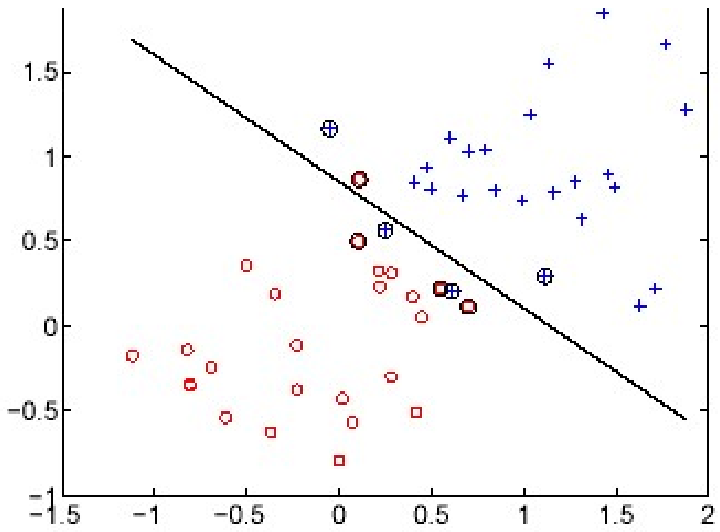
E.g., curve fitting:



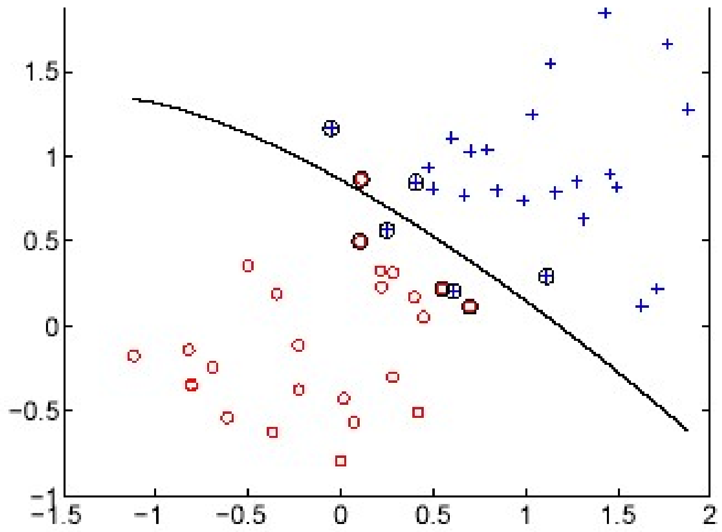

Dico ergo ad qõnem ꝙ qꝛ pluralitas non est ponenda sine necessitate ⁊ non ē necessitas quare debeat poni tp̄us di- scretum mensurãs motum angeli. naꝫ

novacula Occami

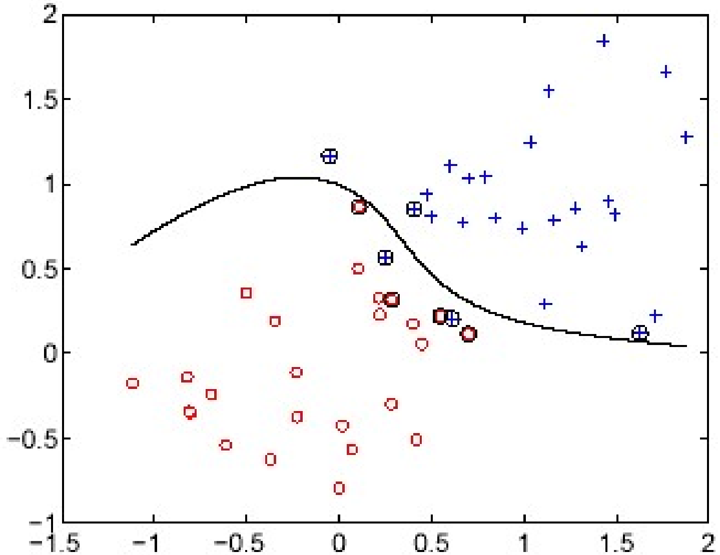
Ockham's razor:
prefer the simplest hypothesis consistent with data



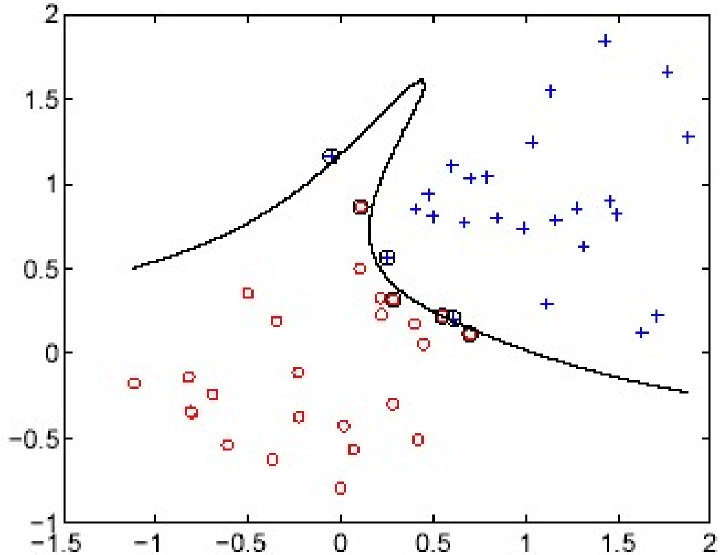
linear



2^{nd} order polynomial

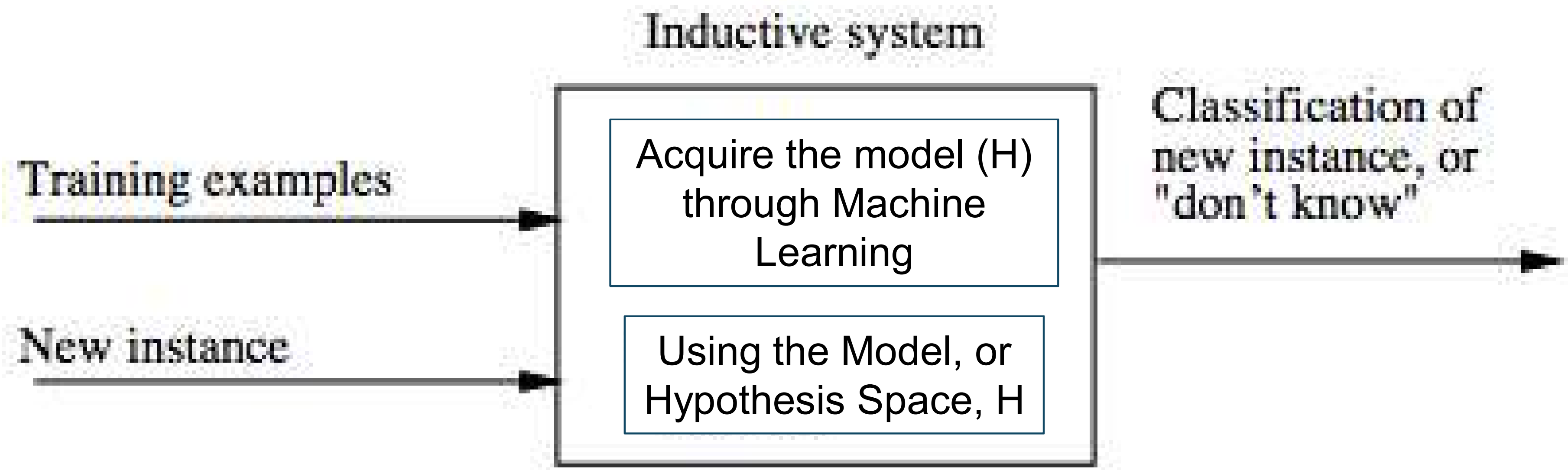


4^{th} order polynomial

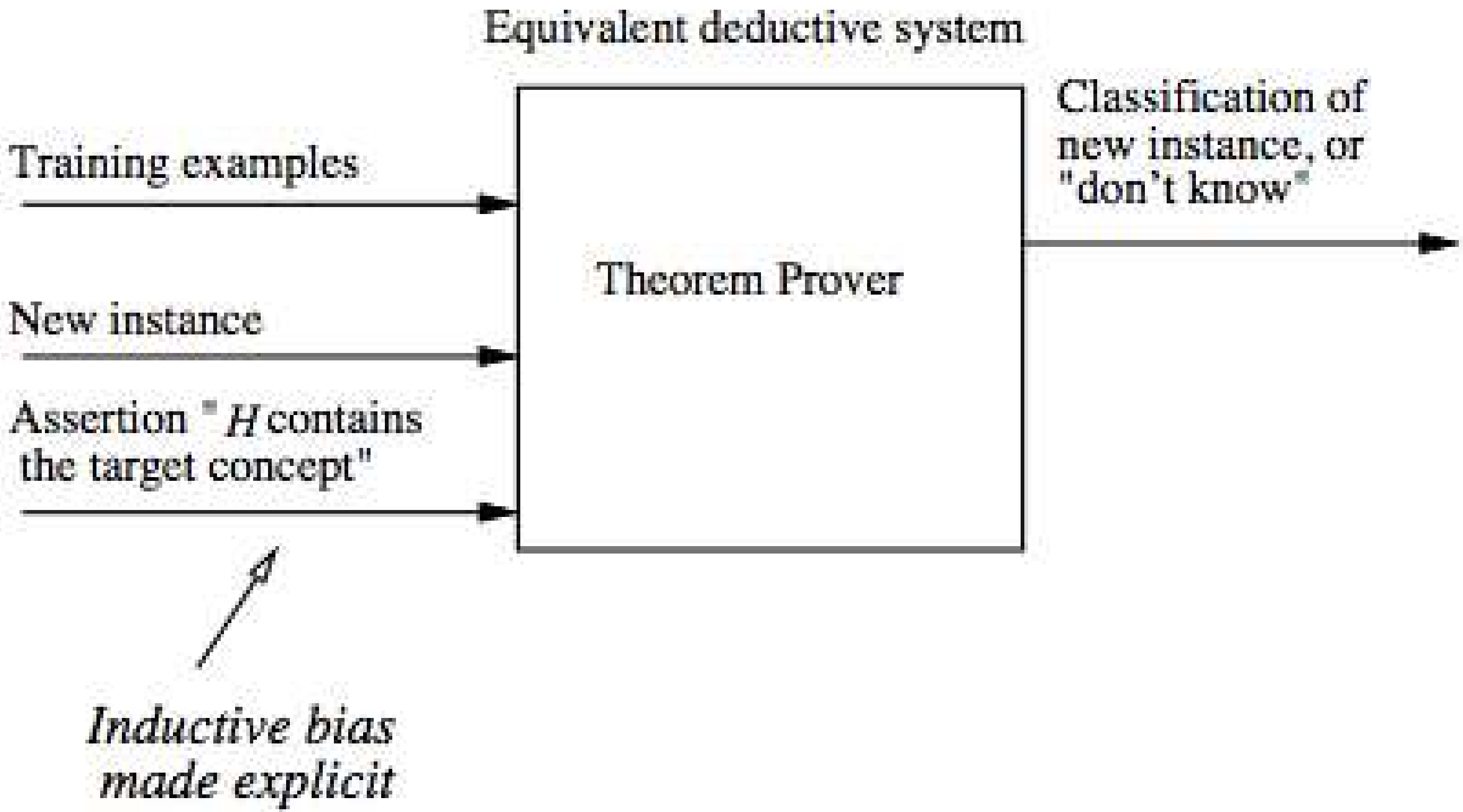


8^{th} order polynomial

Inductive system



Equivalent deductive system



Learning decision trees

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

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

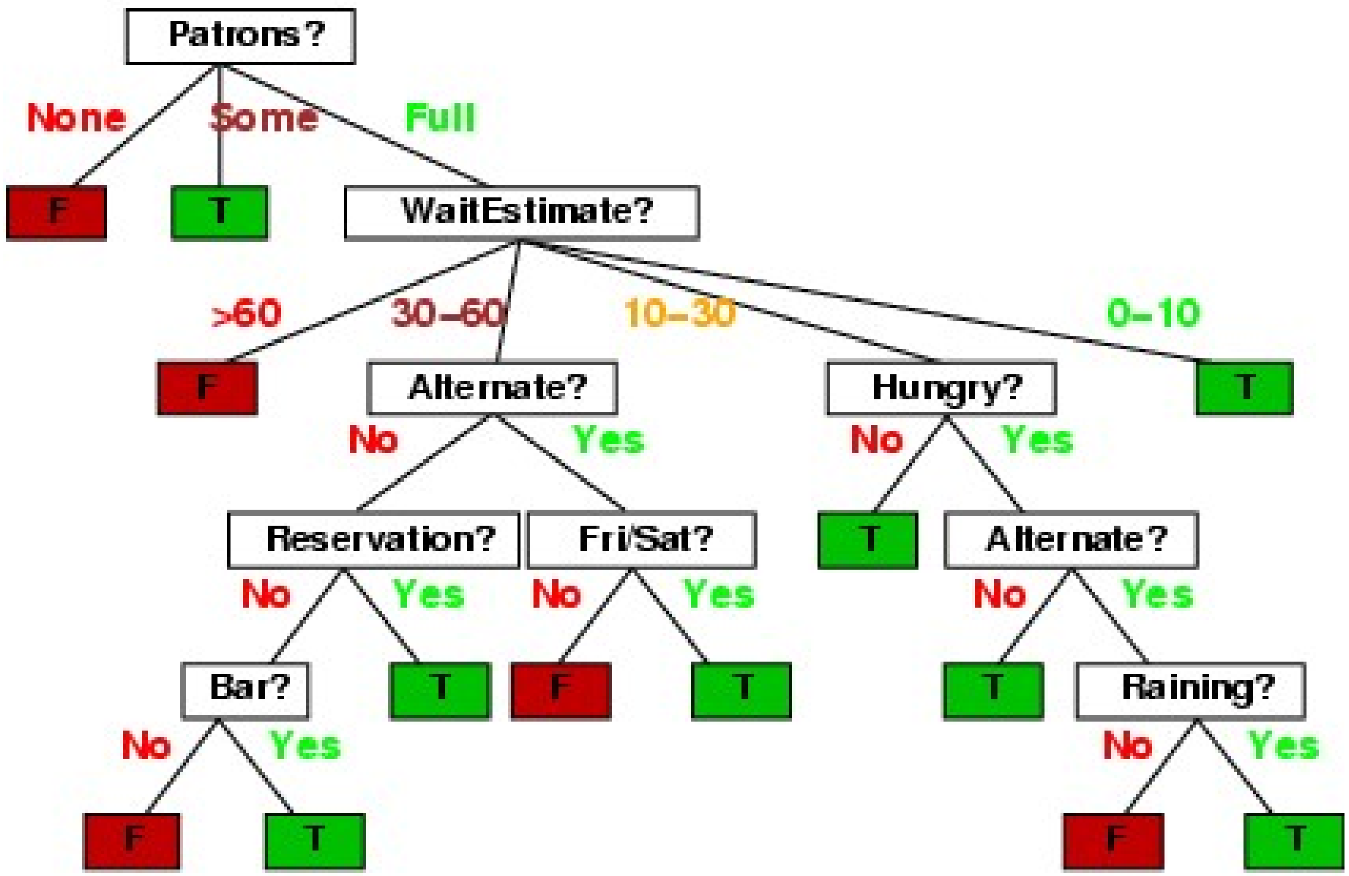
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

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

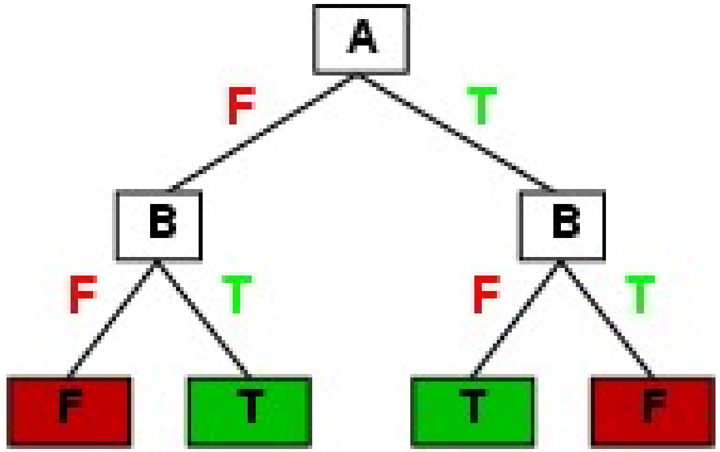
- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:



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 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?

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

- 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 with training set

    $\Rightarrow$ may get worse predictions

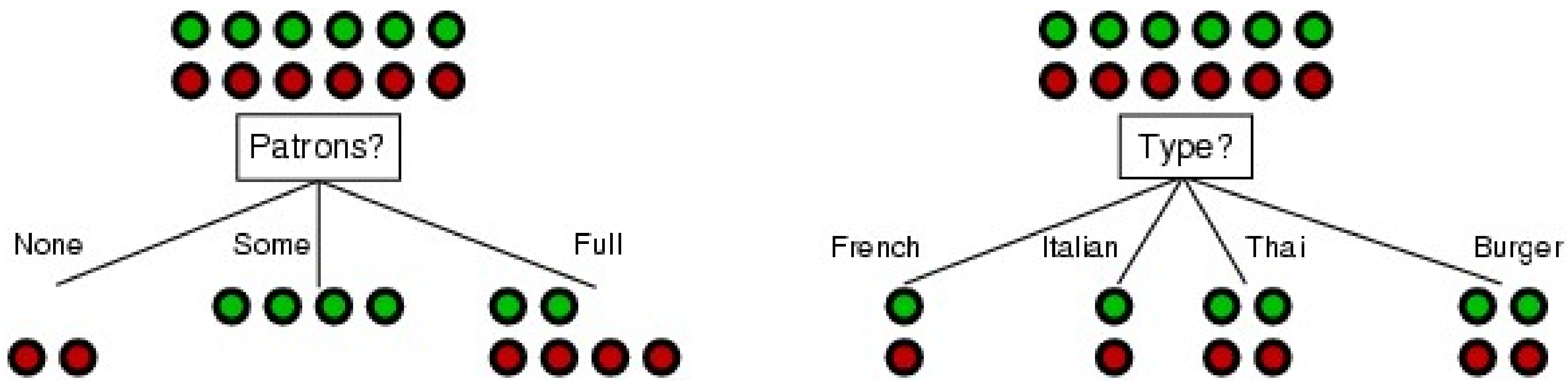
Decision tree learning

- 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*

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):

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

- 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

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

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

(also called entropy of the prior)

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

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

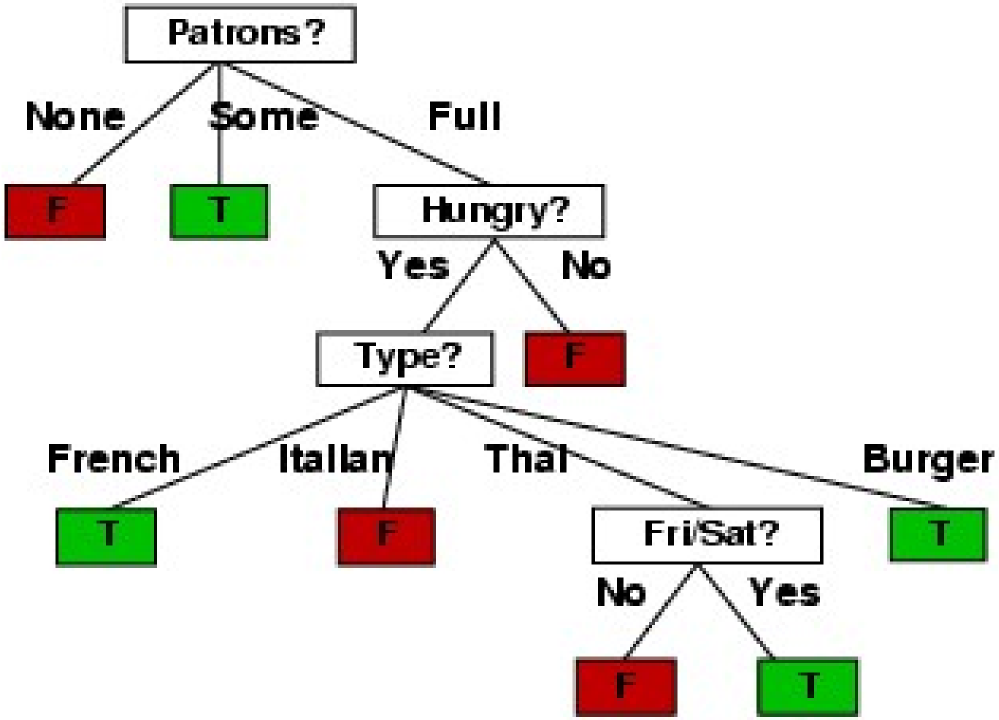
$$\Sigma_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$$

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

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

Example contd.

- Decision tree learned from the 12 examples:



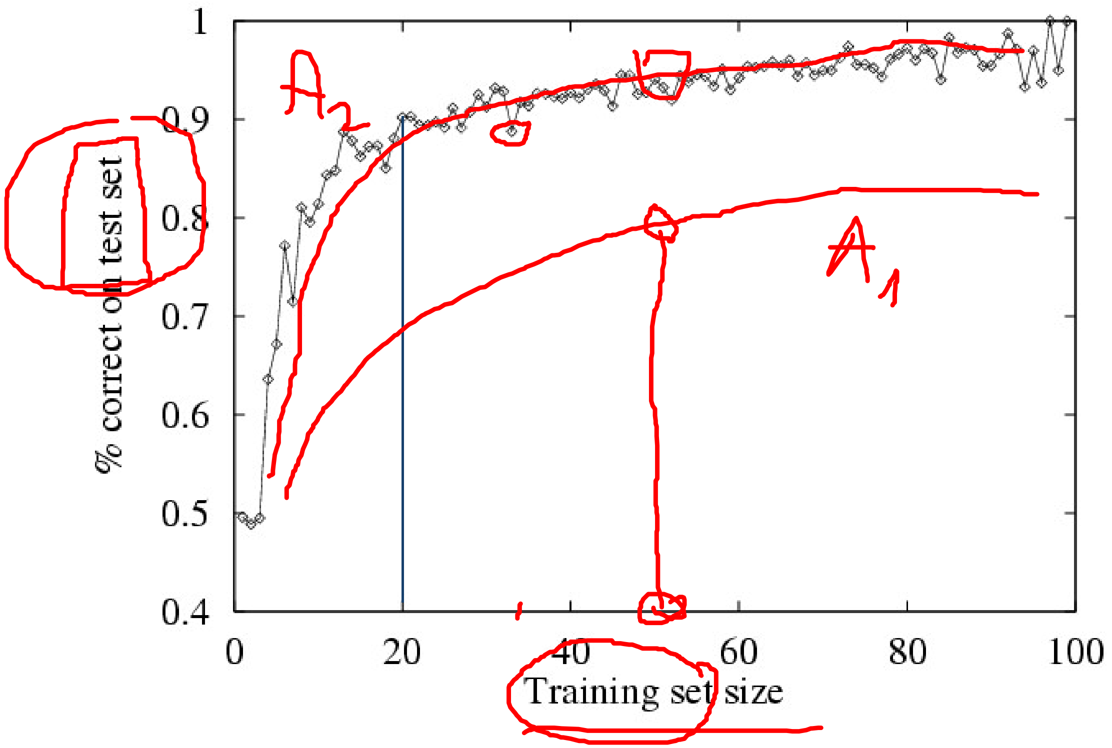
- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data

Performance measurement

- How do we know that we can stop learning, i.e. $h \approx f$?
  1. Use theorems on *h/f* (computational/statistical learning theory)
  2. Empricially, we try *h* on a new test set of examples

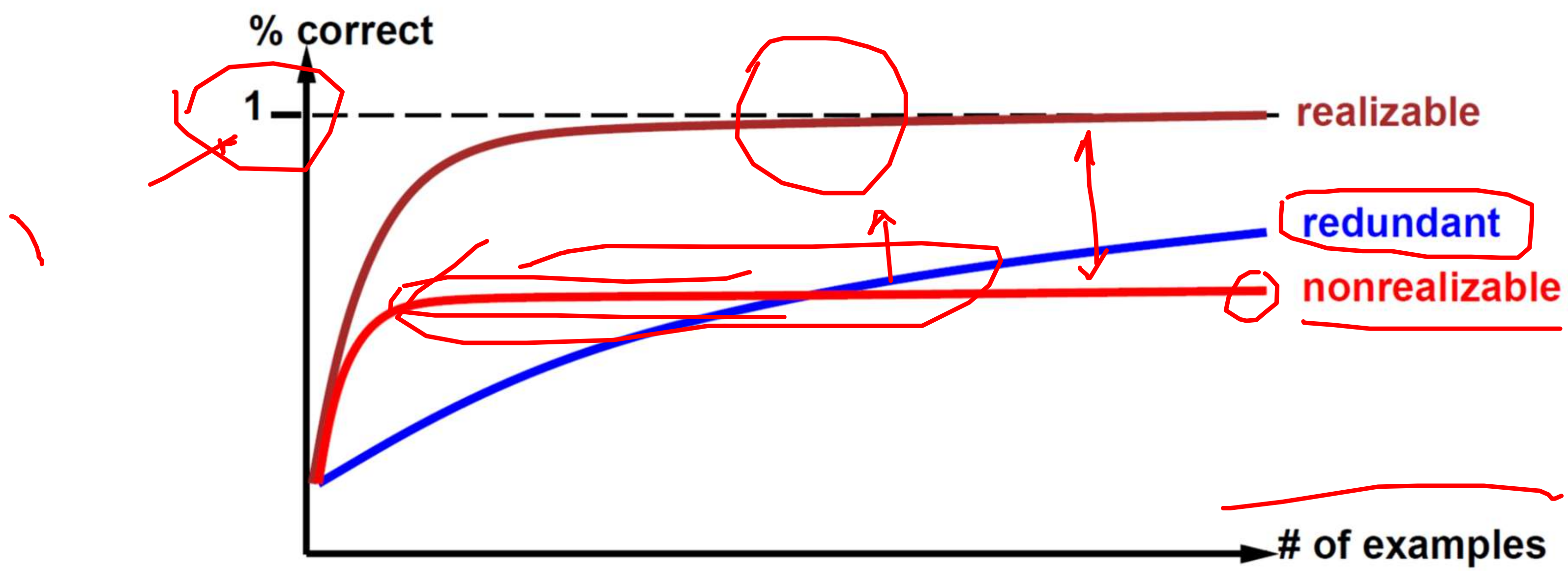
     (use same distribution over example space as training set)

Learning curve = % correct decisions on the test set as a function of the training set size



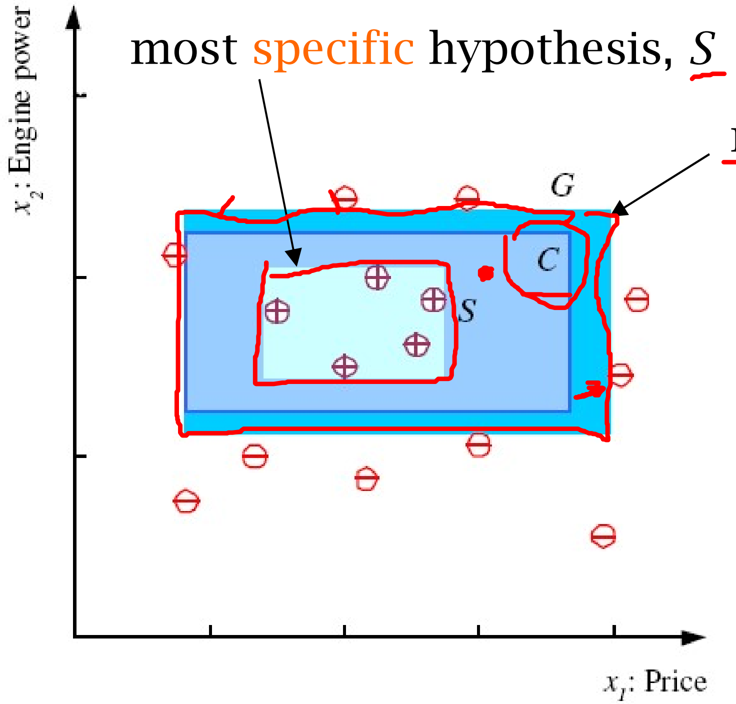
Performance measurements (2)

- **Learnability** depends on
  - **realizable** kind of performances vs.
  - … **non-realizable** ones
  - **Non-realizability** depends on
    - Missing attributes
    - Limitation on the hypothesis space (e.g. non expressive functions)
  - **Redundant expressiveness** is related to cases where a a large number of irrelevant attributes are used



… short look at *model selection*

Spazi e Modelli nel processo di *Learning*



The model $h \in \mathcal{H}$ floats between S and G to be consistent

The corresponding family makes up the version space

(Mitchell, 1997)

Model selection

- We try to find the model with the best balance of complexity and the fit to the training data
- Ideally, we would select a model from a nested sequence of models of increasing complexity (VC-dimension)

  Model 1 d_1
  Model 2 d_2
  Model 3 d_3

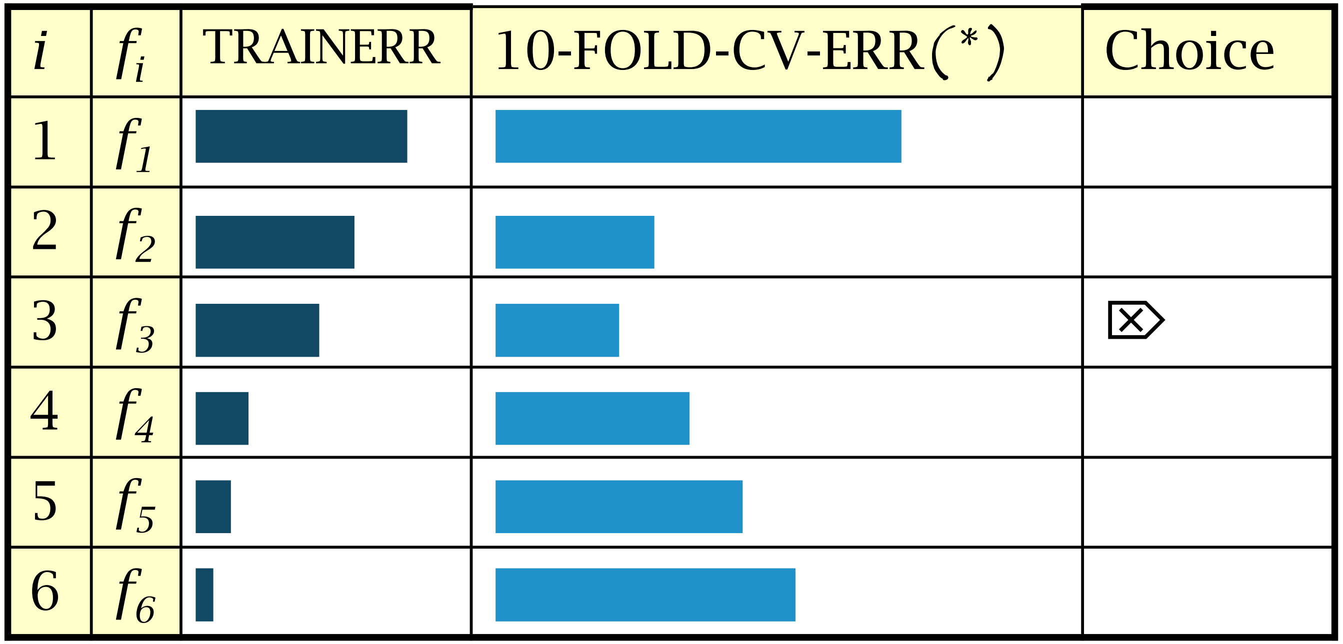
  where $d_1 \leq d_2 \leq d_3 \leq \ldots$

- The model selection criterion is: find the model class that achieves the lowest upper *bound* on the expected loss

$$\text{Expected error} \leq \text{Training error} + \text{Complexity penalty}$$

Alternatives to theory-driven model selection

- **Cross-validation** *(*)*, repeat training vs. many testing data set (e.g. through sampling)



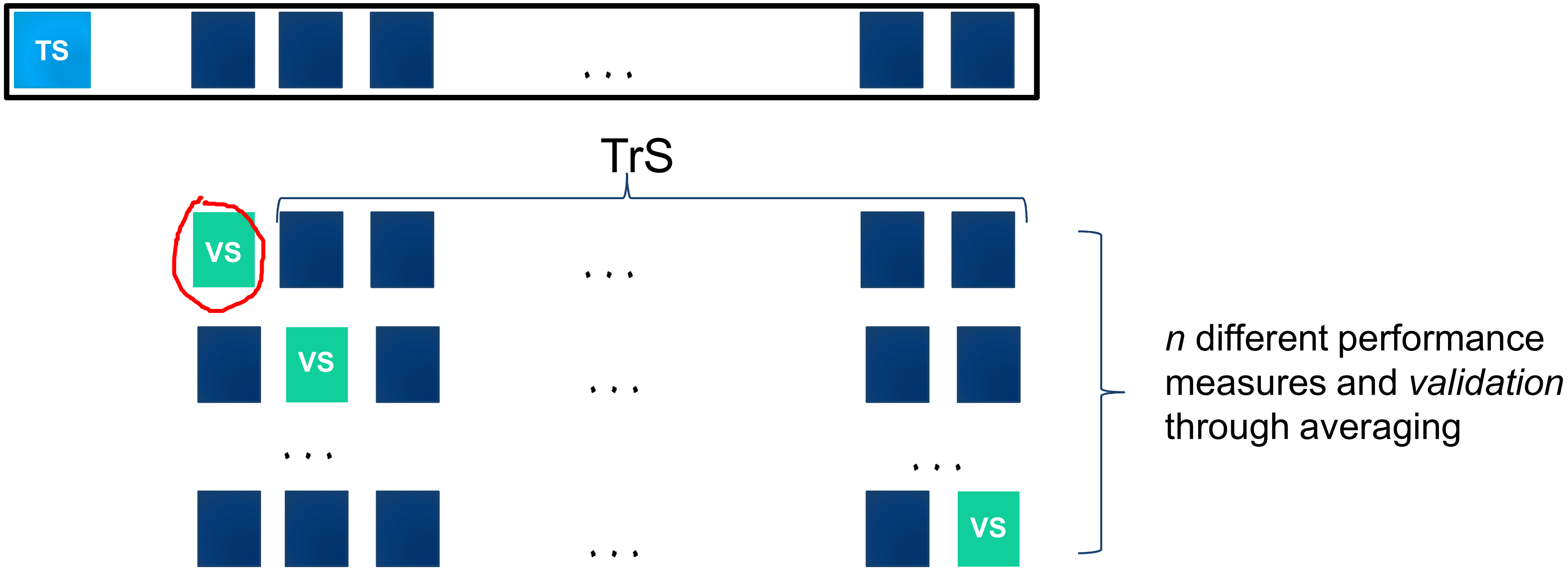
Best trade-off

(*) Different test set TS (as folds) are used for validation, see next slide

n-fold Cross validation

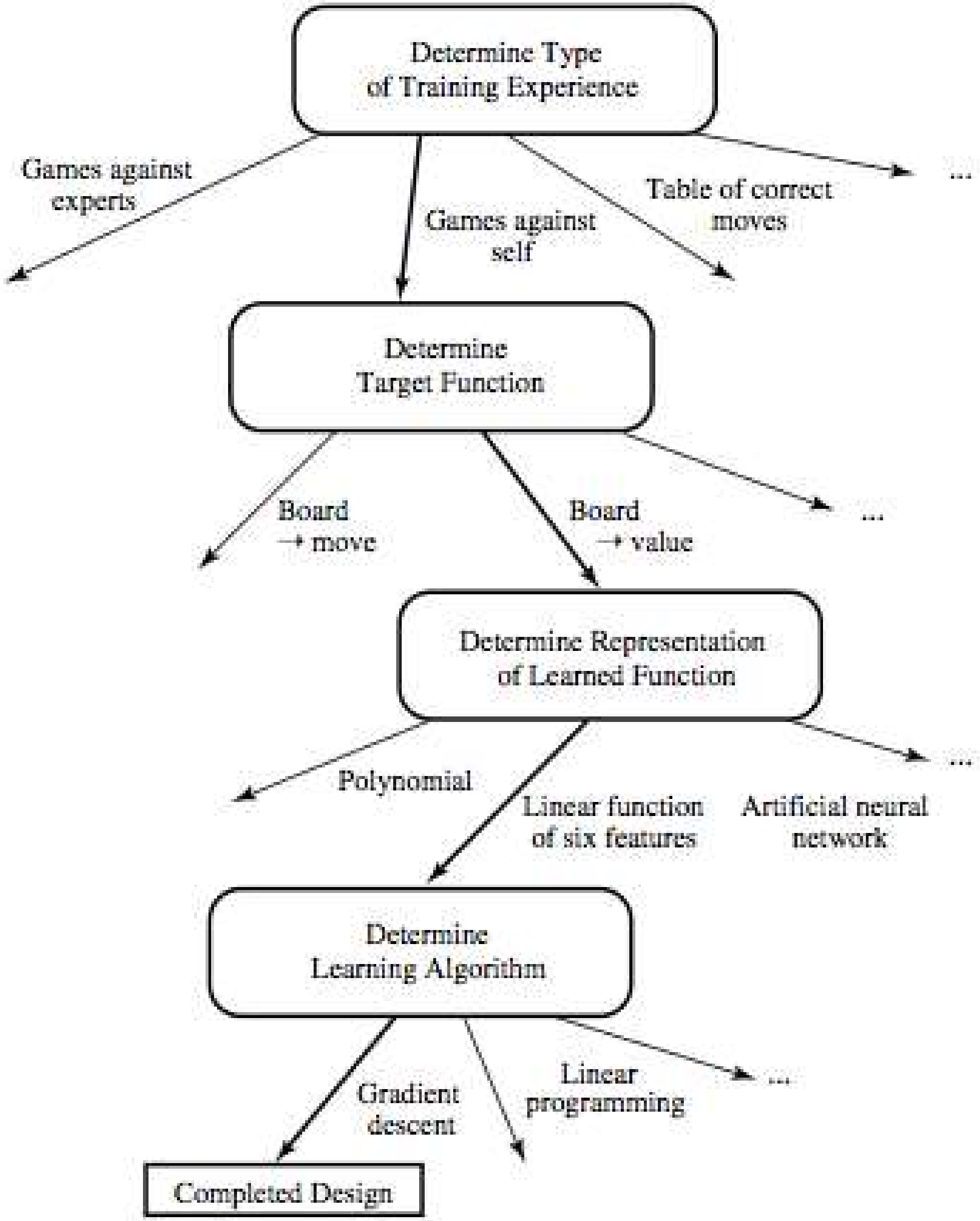
Annotated Data form a collection of already categorized examples. It can be split into:

- **Test Set** (**TS**): a usually fixed portion of examples randomly selected from the annotated ones
- **Training Data**: the portion of annotated data, partitioned into *n* sets of the same size, called *folds*
  - **Validation set** (**VS**): 1 fold that can be randomly picked up to *n* times
  - **Training Set** (**TrS**): all the remaining *n-1* folds

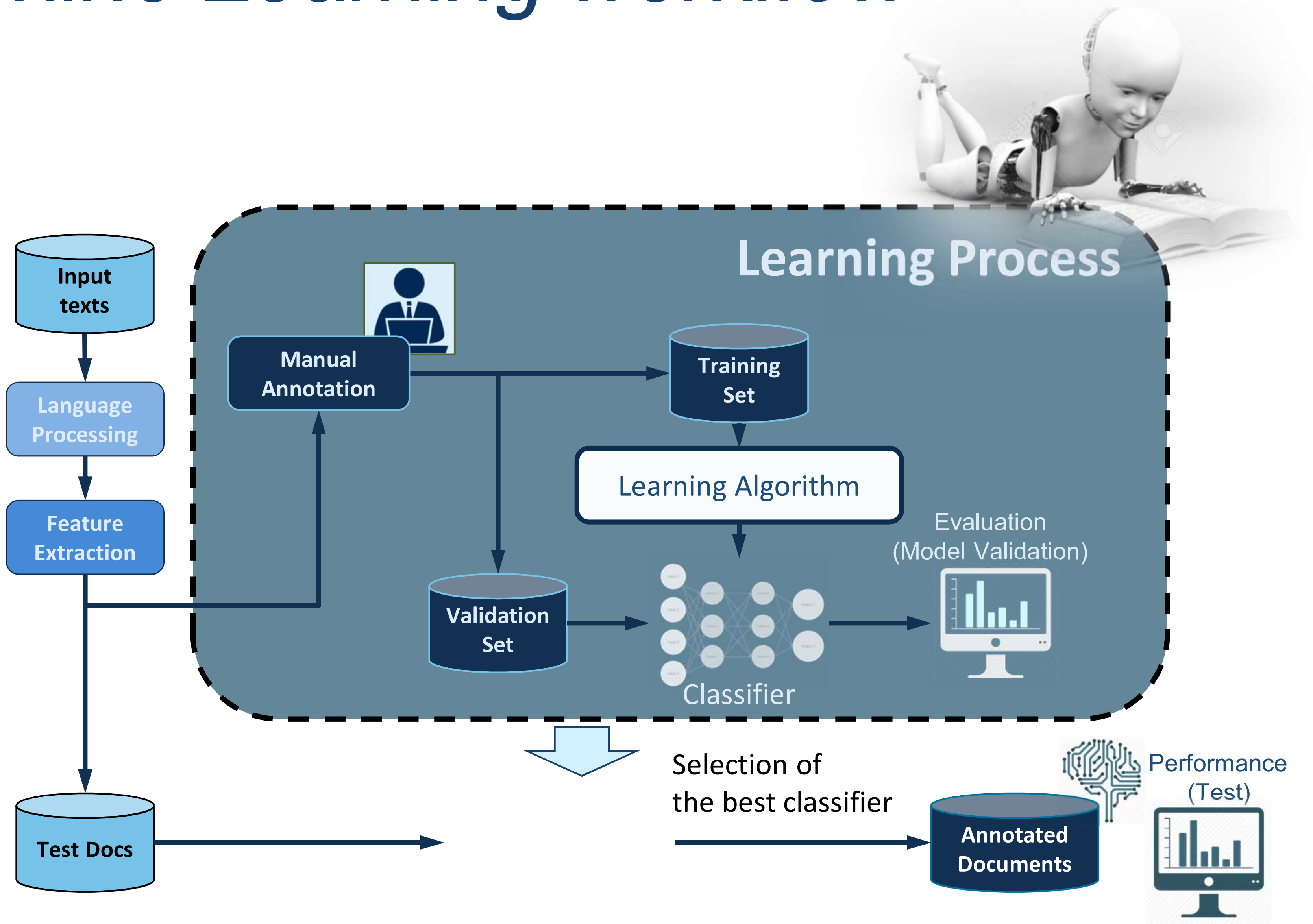


Design of a learning system

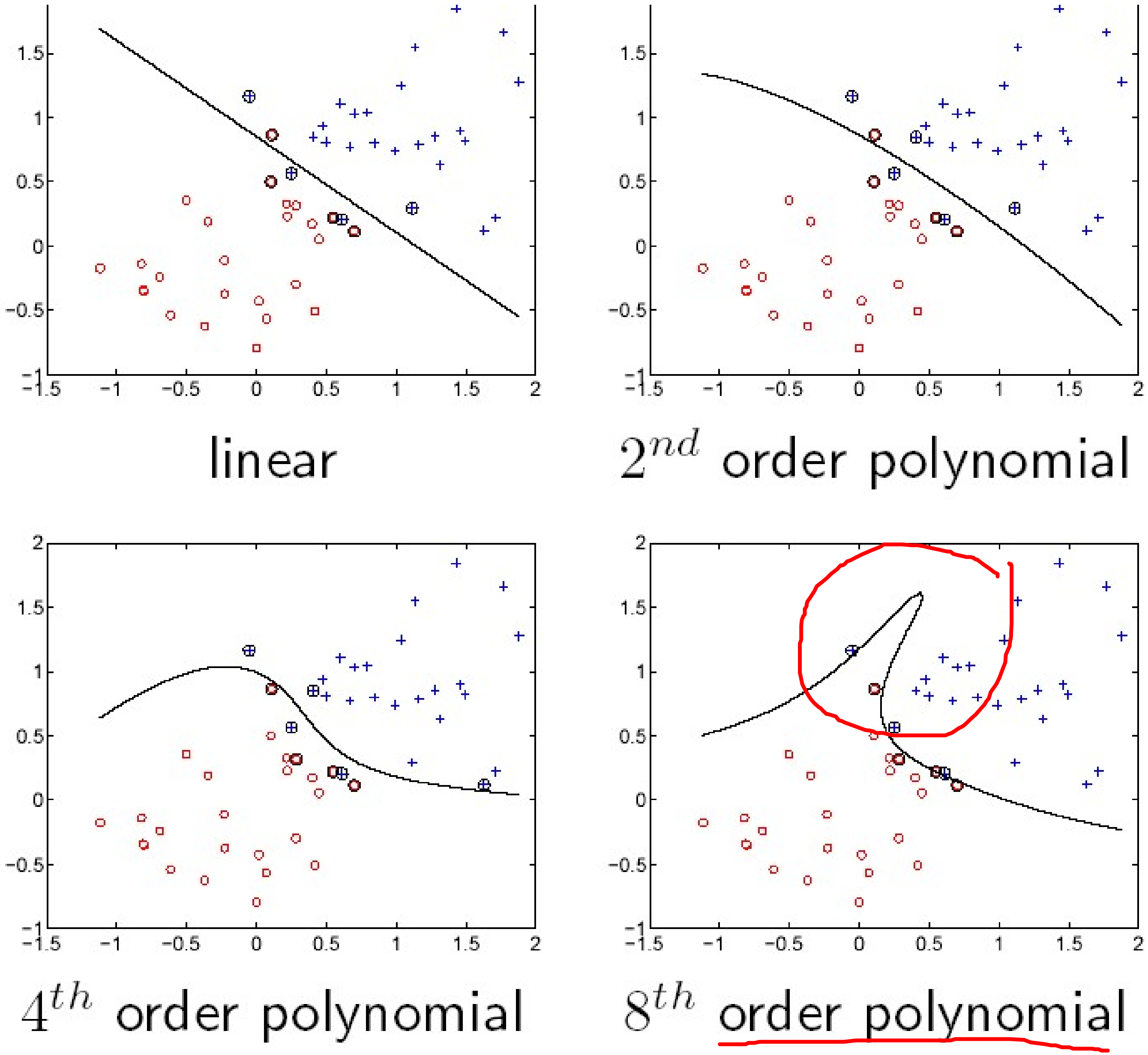
Mitchell, 1997



Machine Learning workflow



The risk of overfitting the data



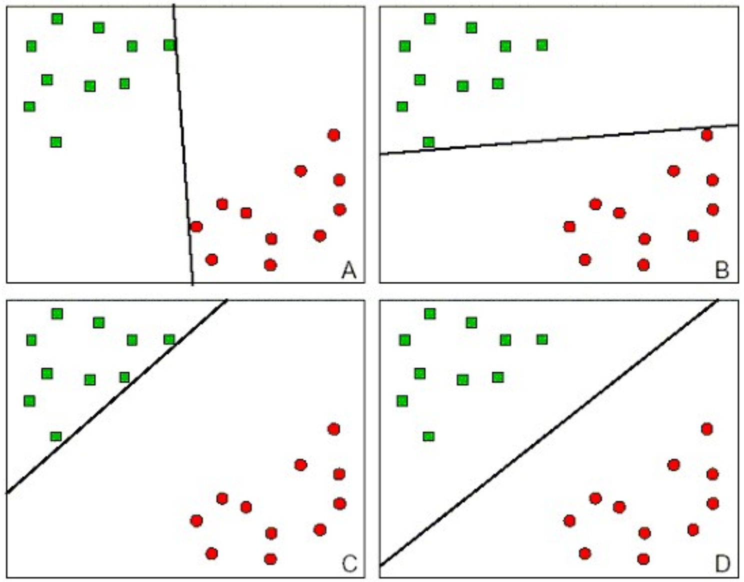
Walter Hartwell White, *aka* Heisenberg

Other approaches to model selection

- **Discriminative approaches**
  - Linear functions (e.g. SVM)

    $h(x) = sign(\, W \cdot x + b)$

  - Challenges:
    - How to estimate the best linear model (i.e. an hyperplane)?
    - How to combine the results of different binary decisions?

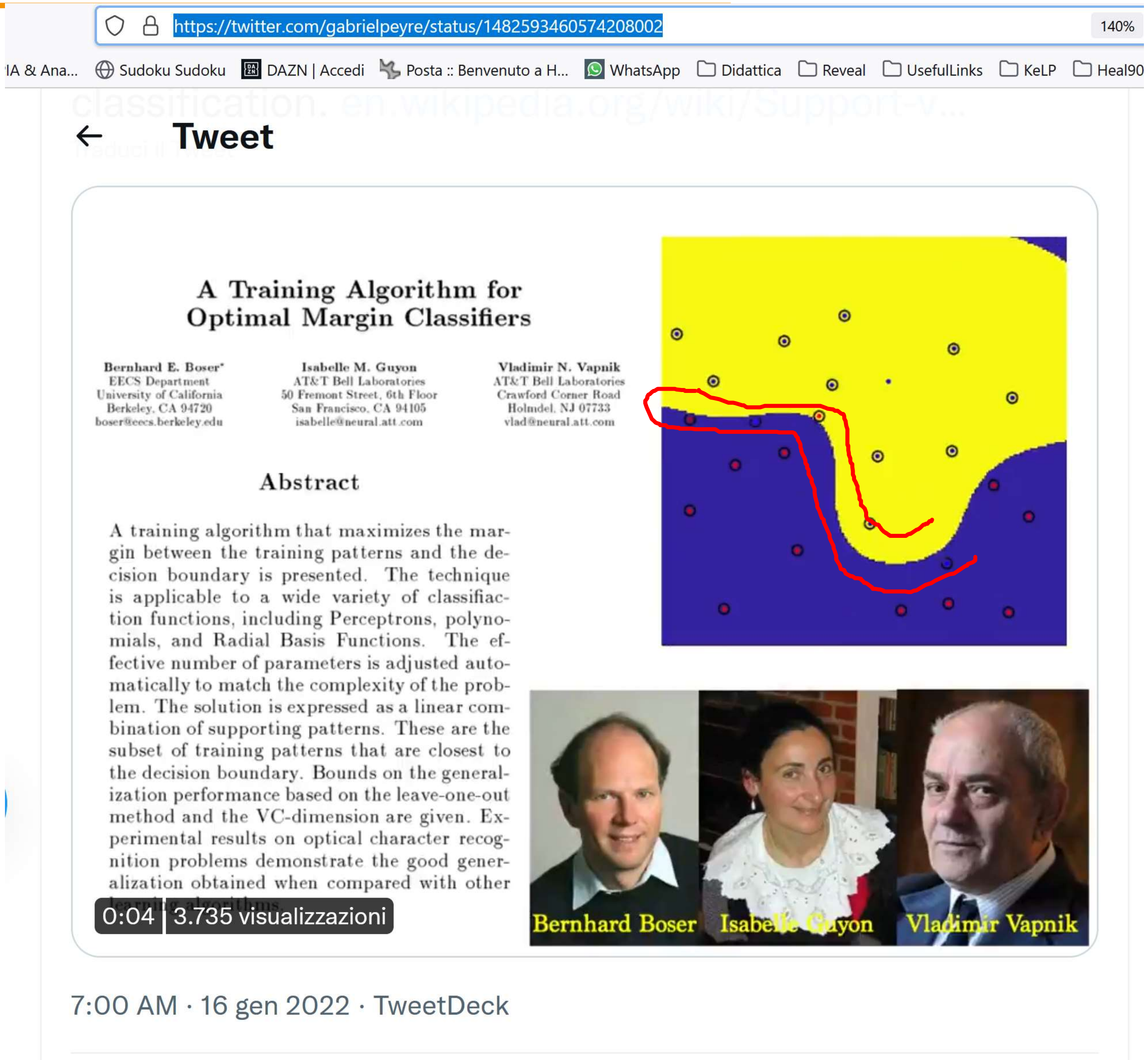


- **Probabilistic Approaches**
  - Probabilty estimates of $p(\mathbf{x}|\mathcal{C}_k)$ through the training set
  - Application of a generative model through the Bayes inversion

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}.$$

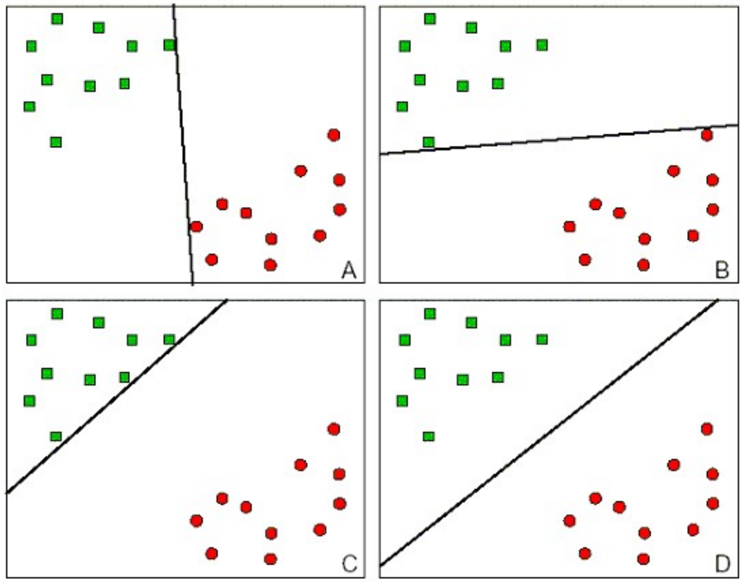
Linear classifiers and kernels

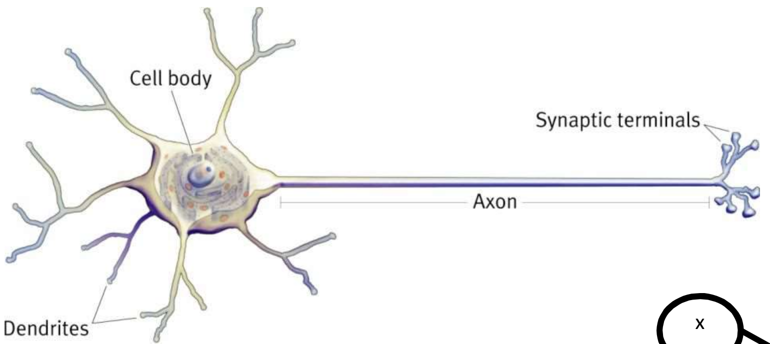
- [Support Vector Machine and Kernels]



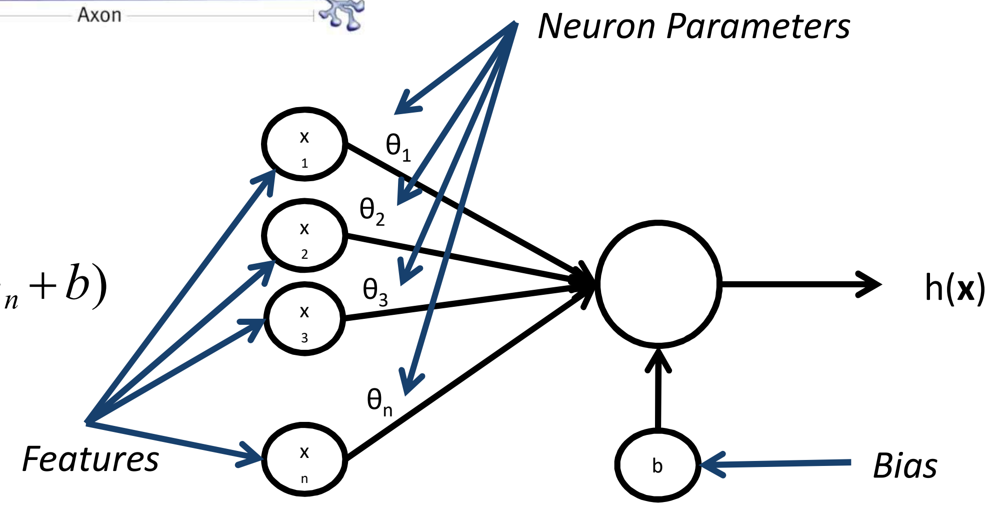
Perceptron (Rosenblatt, 1958)

- Linear Classifier mimicking a neuron





$$h(\vec{x}) = g(\sum_n \theta_n x_n + b)$$



SummarAIzing (1)

- Machine learning from examples is concerned with the ability to induce a decision function out from data that examplify the decision onto a small set (i.e. a sample) of data.
- Learning here means
  - Describe the problem through a set of features that characterize individual instances
  - Define a class of functions (hypothesis) working in the feature space, the target decision function should belong to and
  - Find the best parameters for selecting the best function among different hypothesis: this will be called the model i.e. the function able to decide about the problem in an accurate way
  - The *machine learning workflow* is an iterative incremental cycle of model optimization based on standard example creation practices, data sampling (cross-validation) and performance measurements (accuracy but also precision, recall)

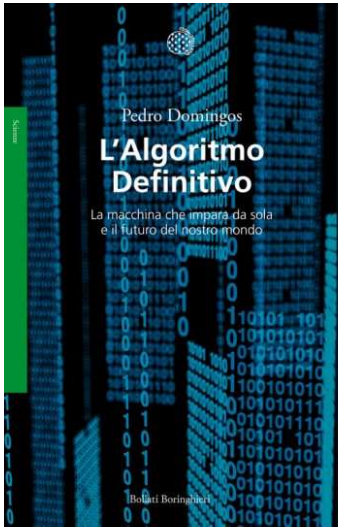
SummarAIzing (2)

- Decision trees are learning algorithms that combine
  - information theory criteria to search within the model space (i.e. as an heuristic function that support fast and optimized search for the *best* model)
  - Recursive algorithmics to develop a data structure (i.e. a tree) as a computationally attractive decision function (logarithmic in the number of tests required to decide)
    - Decision trees are isomorphic to *boolean formulas*
  - *Information gain* is very effective in keeping the size of the DT, i.e. the cost of the search, minimum under probabilistic assumptions about future instances

Riferimenti Bibliografici

- *AIMA,* Chapter 18
- READING. *Machine Learning*, Tom Mitchell, Mc Graw-Hill International Editions, 1997 (Cap 3).

- L'Algoritmo Definitivo, Pedro Domingos, Bollati Boringhieri, 2016



A different view: probabilistic approaches

- The text classification case