INTELLIGENZA ARTIFICIALE

APPRENDIMENTO AUTOMATICO DA ESEMPI

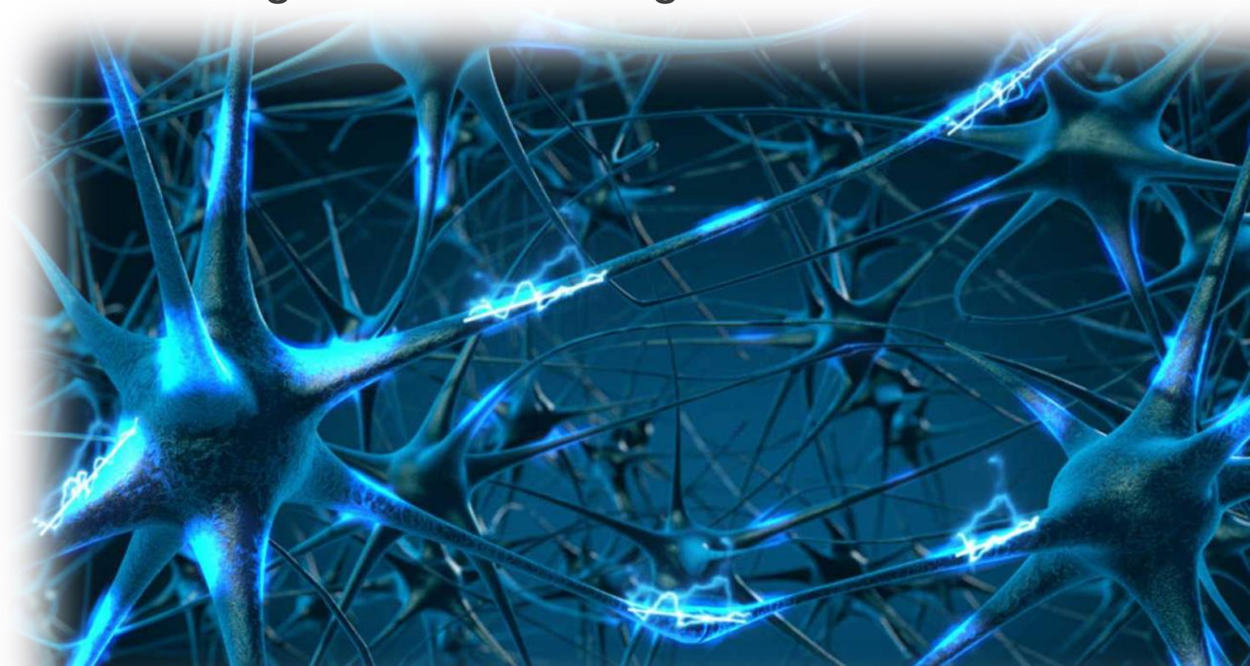
Corsi di Laurea in Informatica, Ing. Gestionale, Ing. Informatica, Ing. di Internet

(a.a. 2023-2024)

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
- Performance Evaluation
- Learning methodology: design, experiment/ evaluation and model selection
  - Cross validation
- An example: Decision Tree learning
  - Recursive search among Boolean formulas
  - Attribute Selection in DT: Information Gain

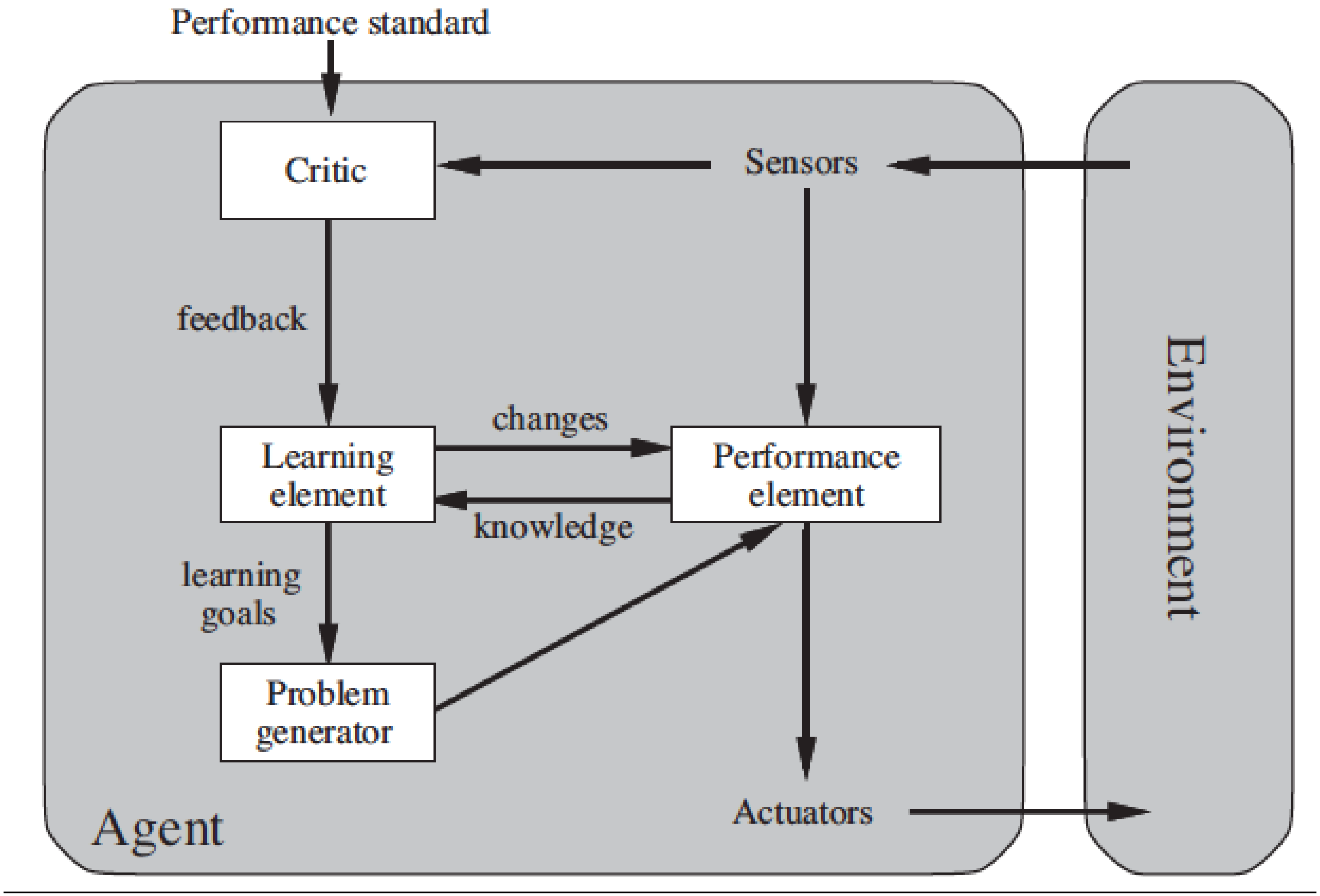
Introduction to machine learning

- 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

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

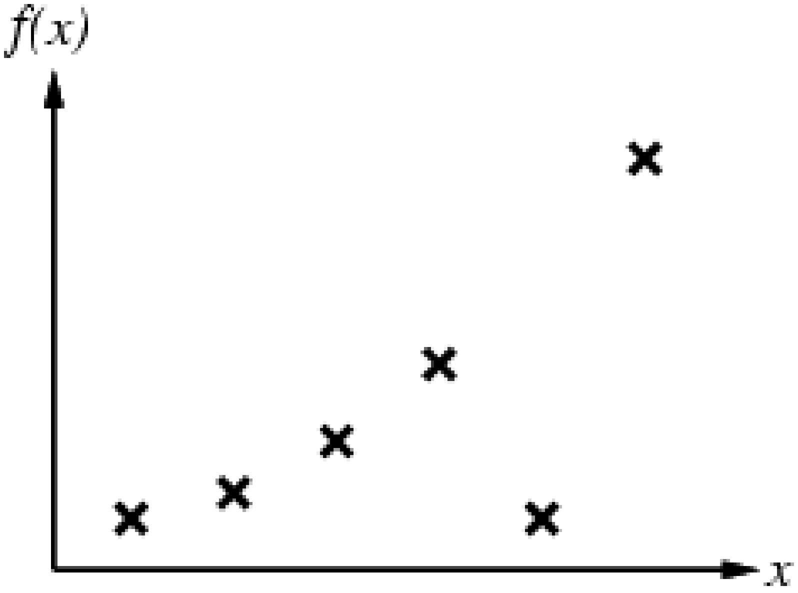
(This is a highly simplified model of real learning:
- Ignores prior knowledge
- Assumes examples are given)

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:

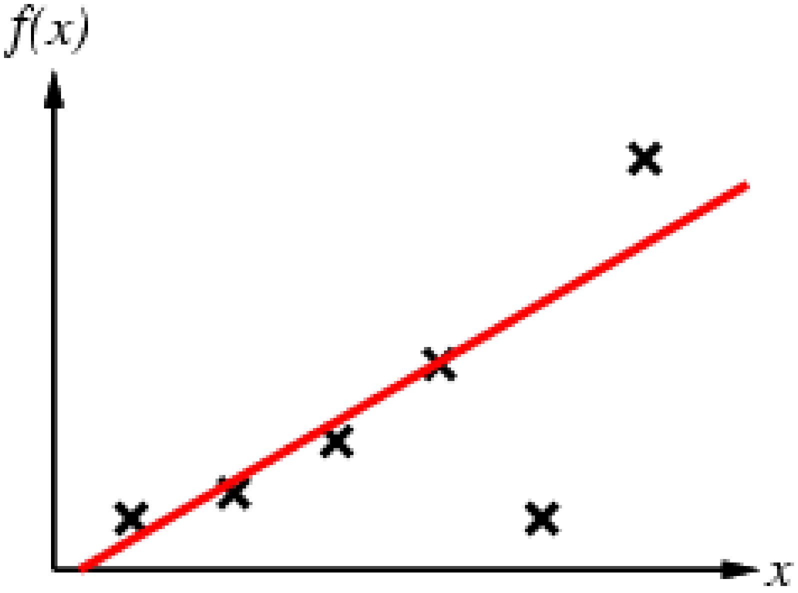


Inductive learning method

- 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:

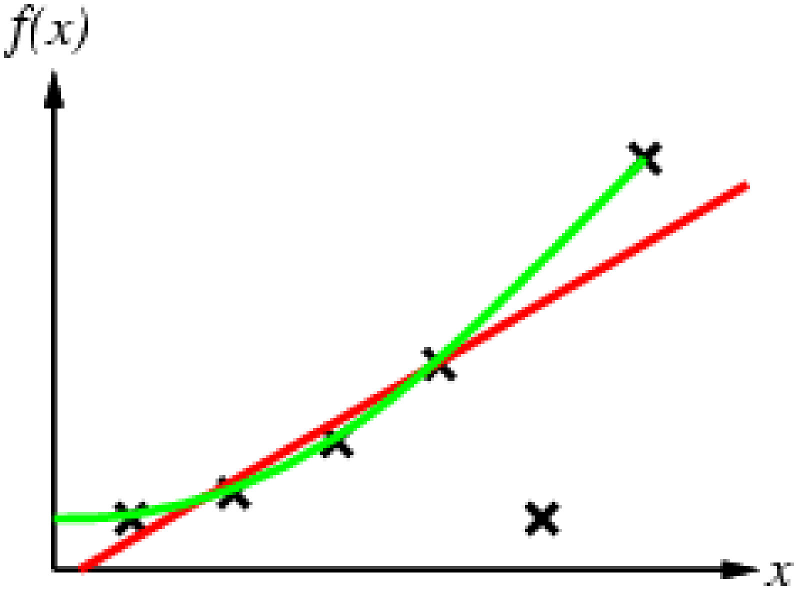


Inductive learning method

- 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:

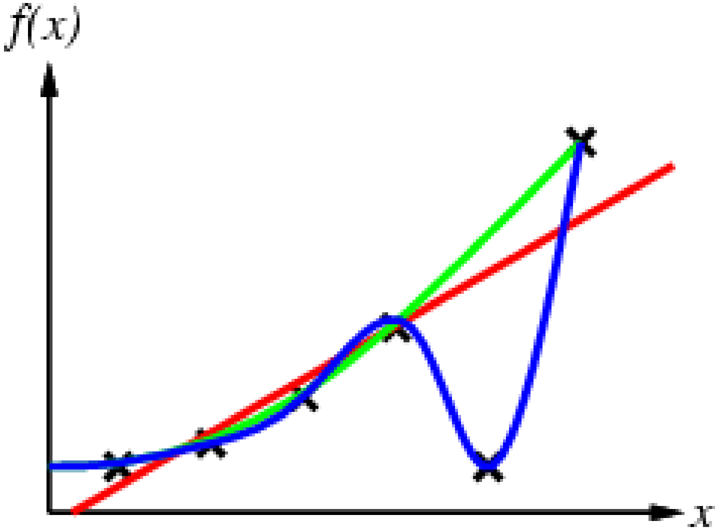


Inductive learning method

- 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:

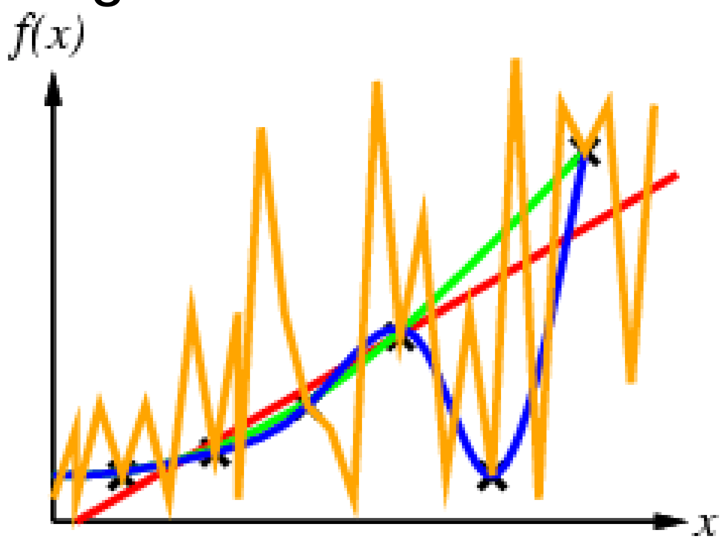


Inductive learning method

- 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:

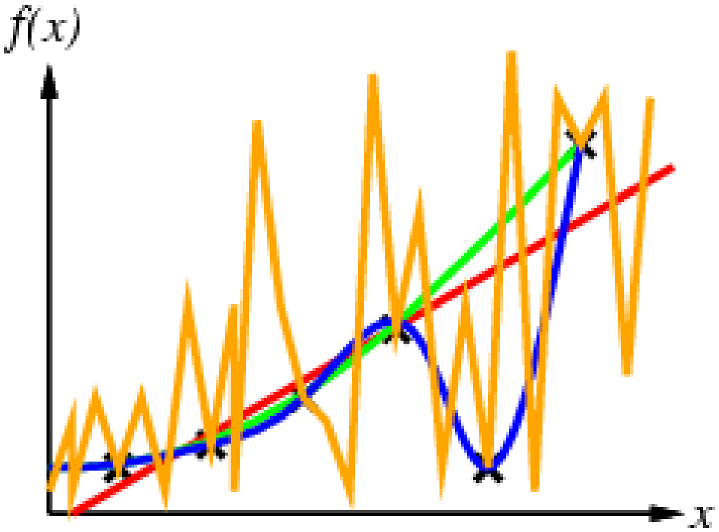


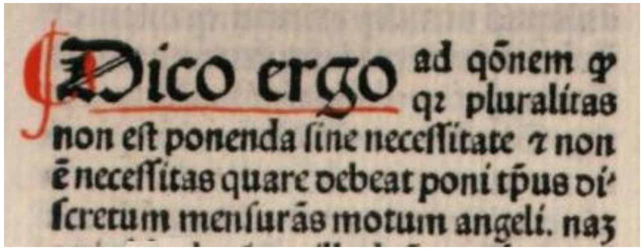
Inductive learning method

- 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:

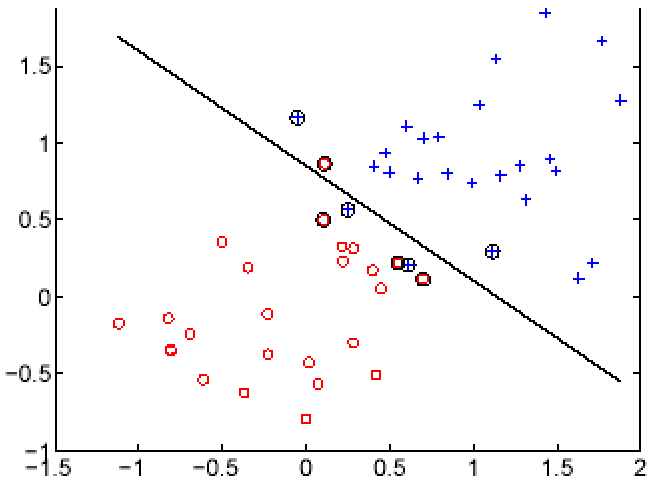




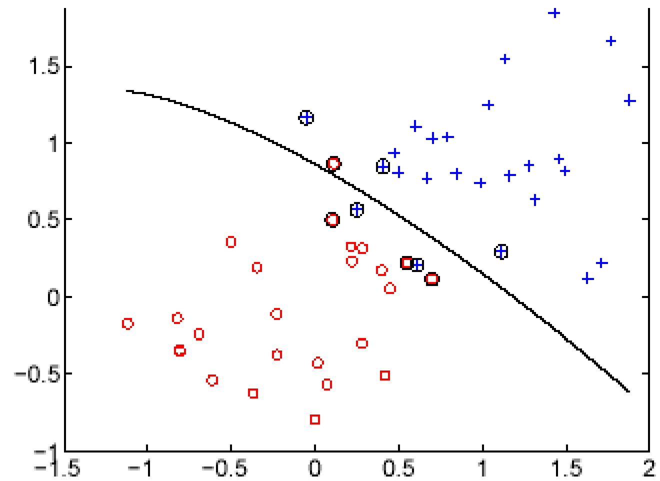
novacula Occami

Ockham's razor:

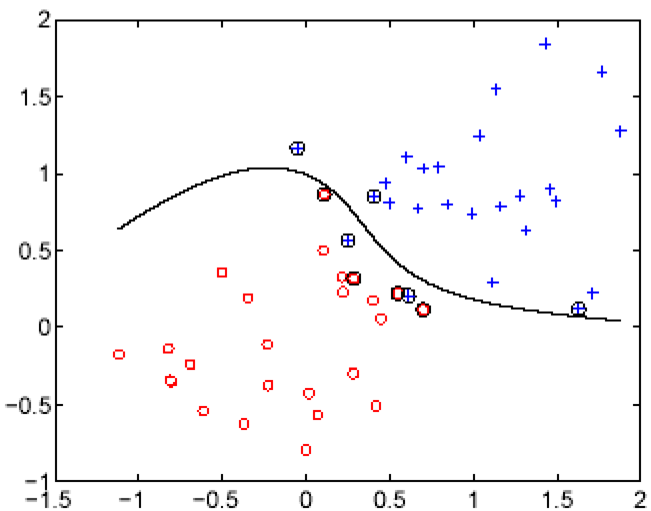
prefer the simplest hypothesis consistent with data



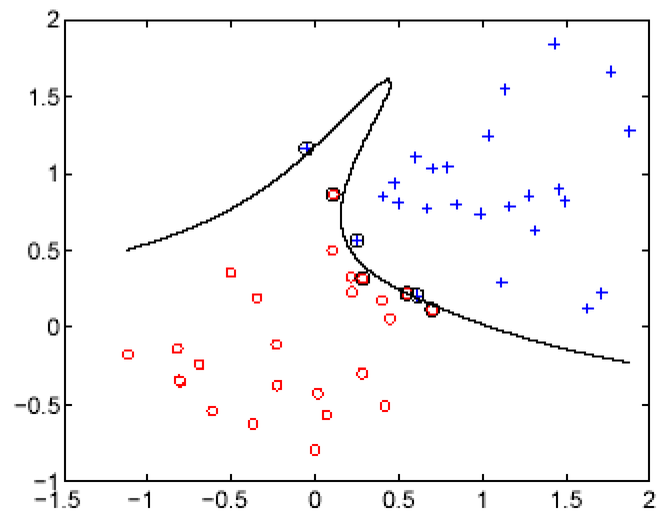
linear



2^{nd} order polynomial

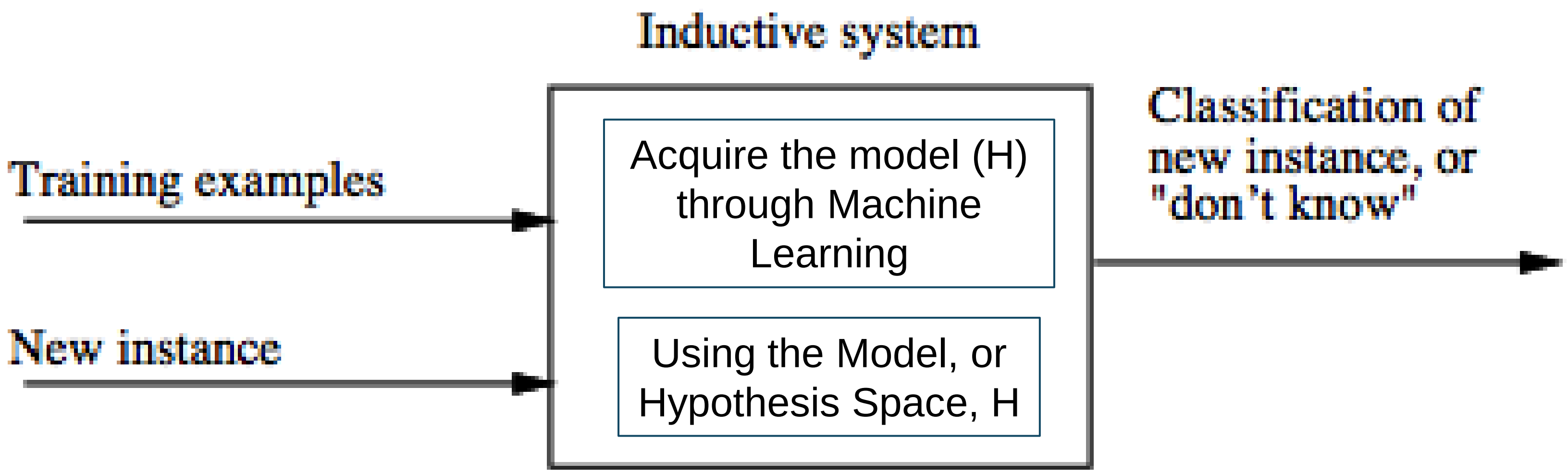


4^{th} order polynomial

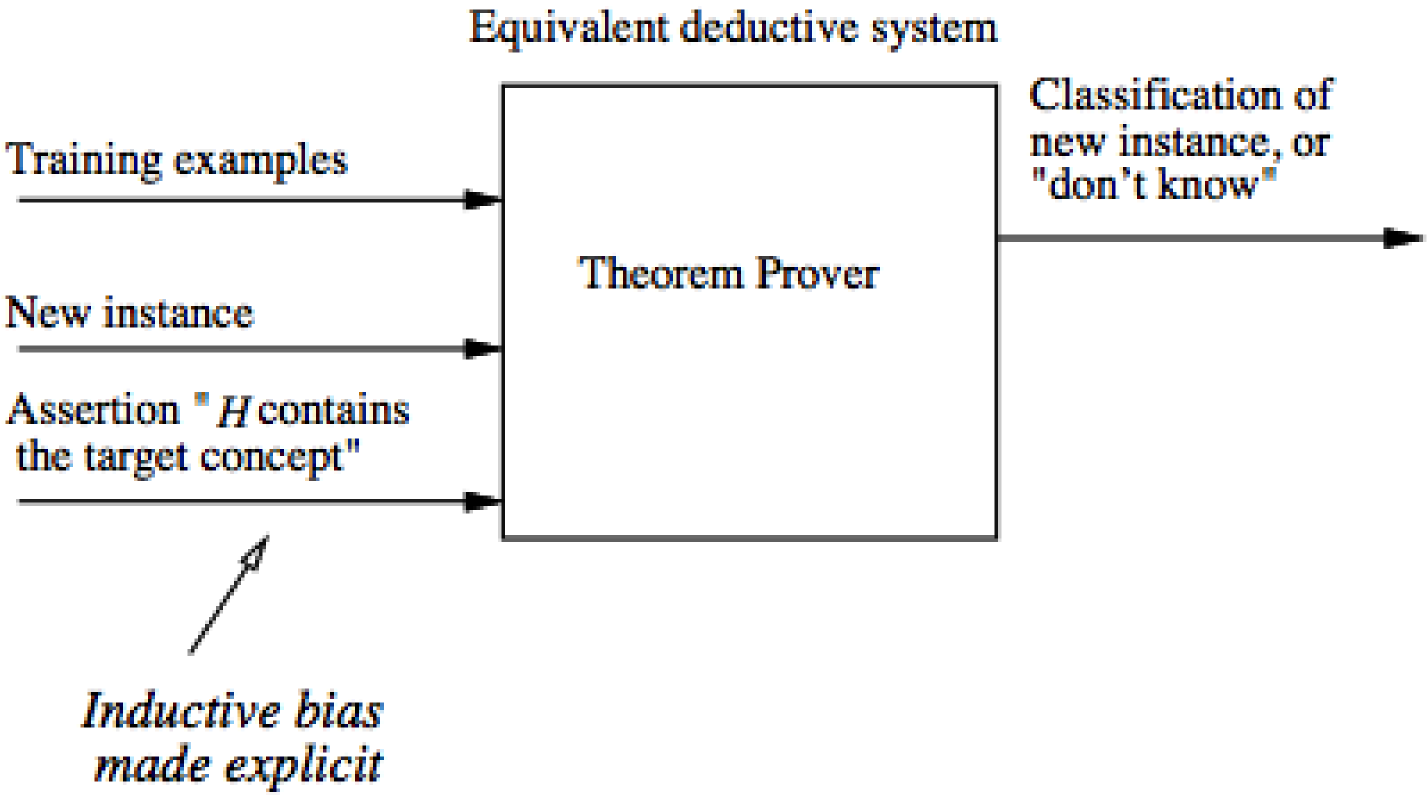


8^{th} order polynomial

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

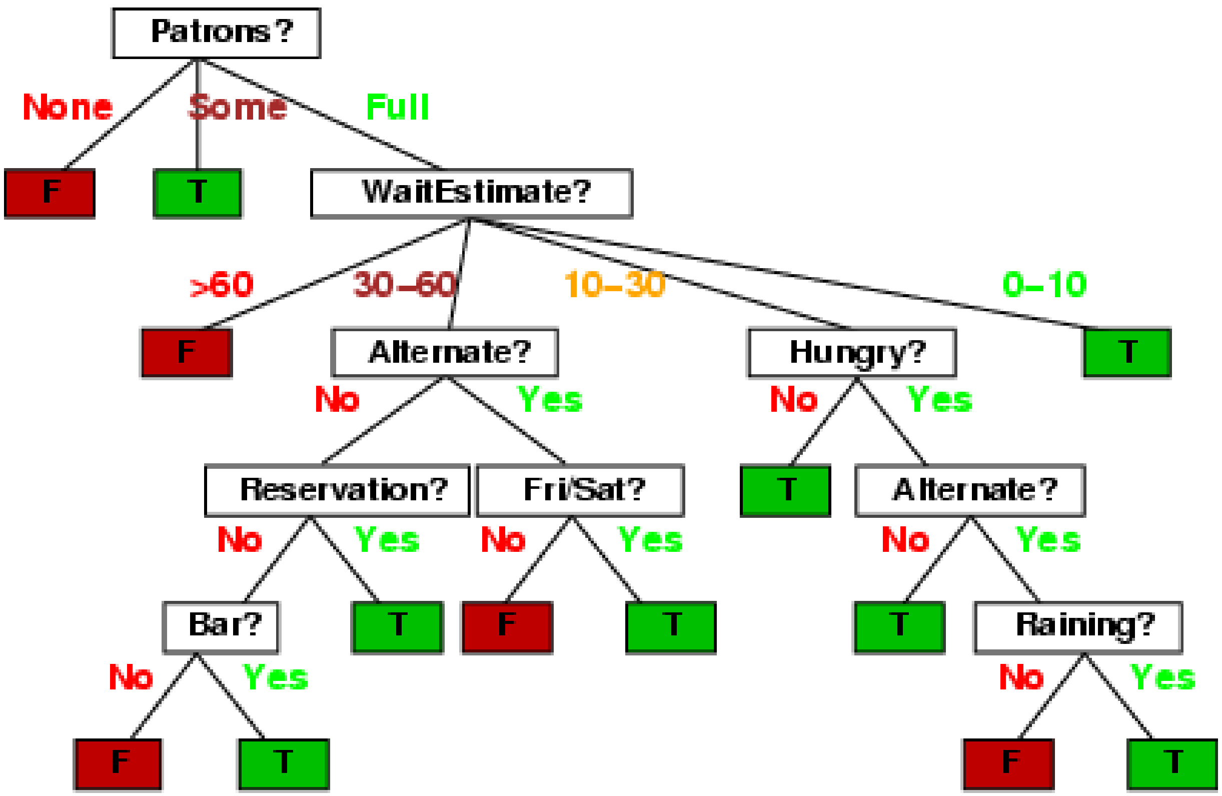
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

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

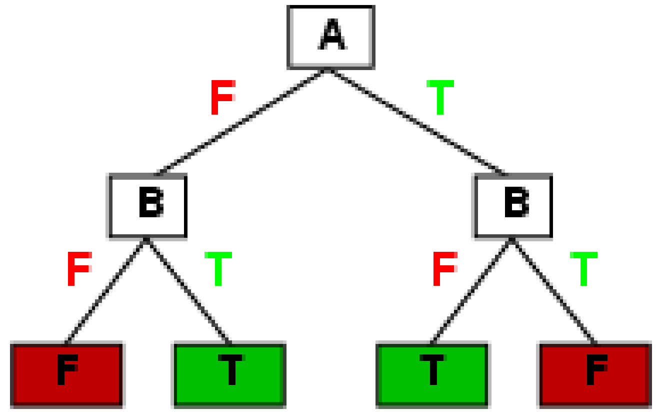
- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:



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

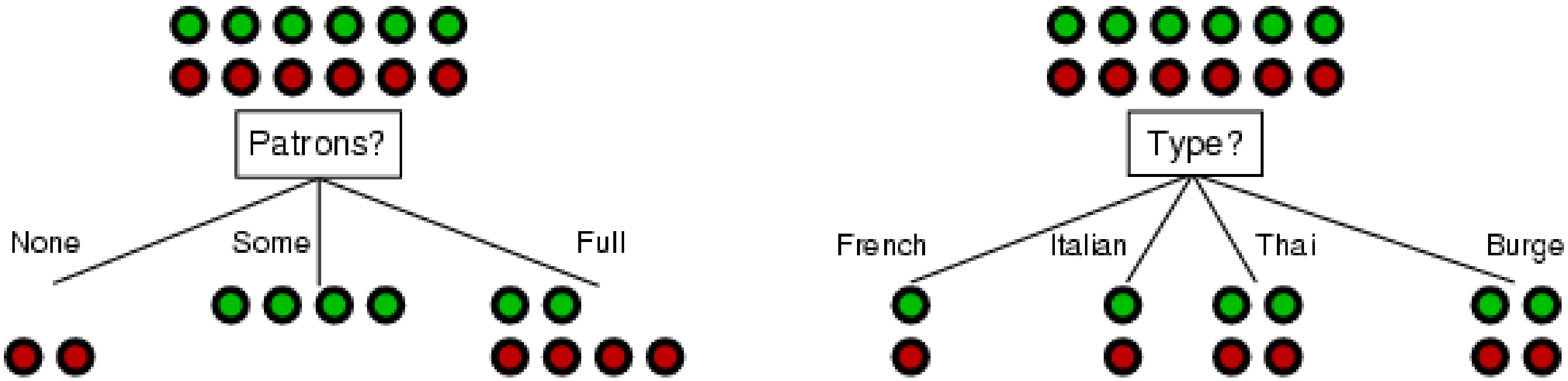
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
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 gain

- A chosen attribute *A* divides the training set *E* into subsets E_1, … , 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):

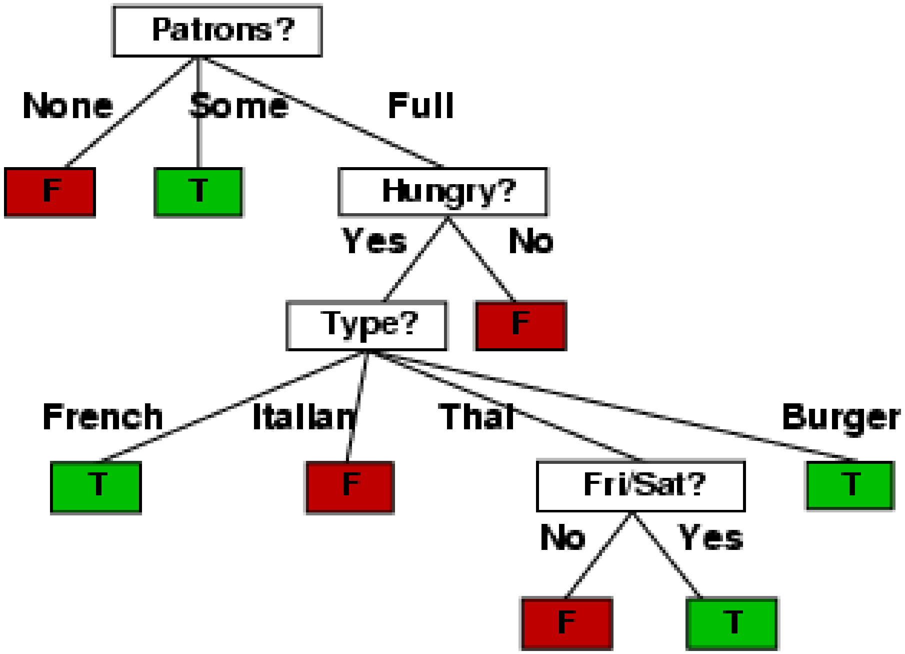
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

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

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

Example contd.

- Decision tree learned from the 12 examples:



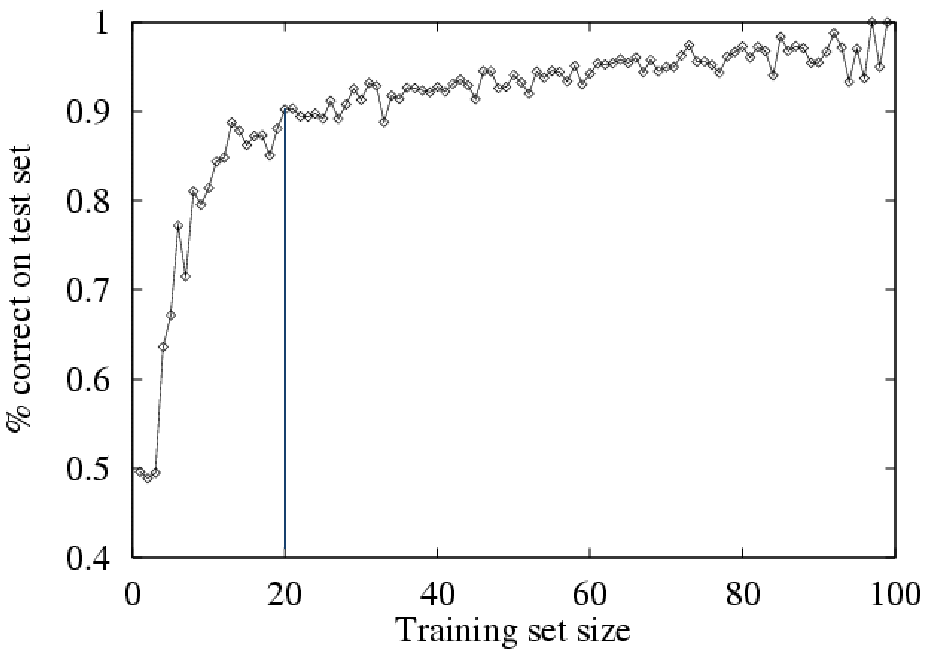
- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data

Performance measurement

- How do we know that $h \approx f$?
  1. Use theorems of computational/statistical learning theory
  2. Try h on a new test set of examples

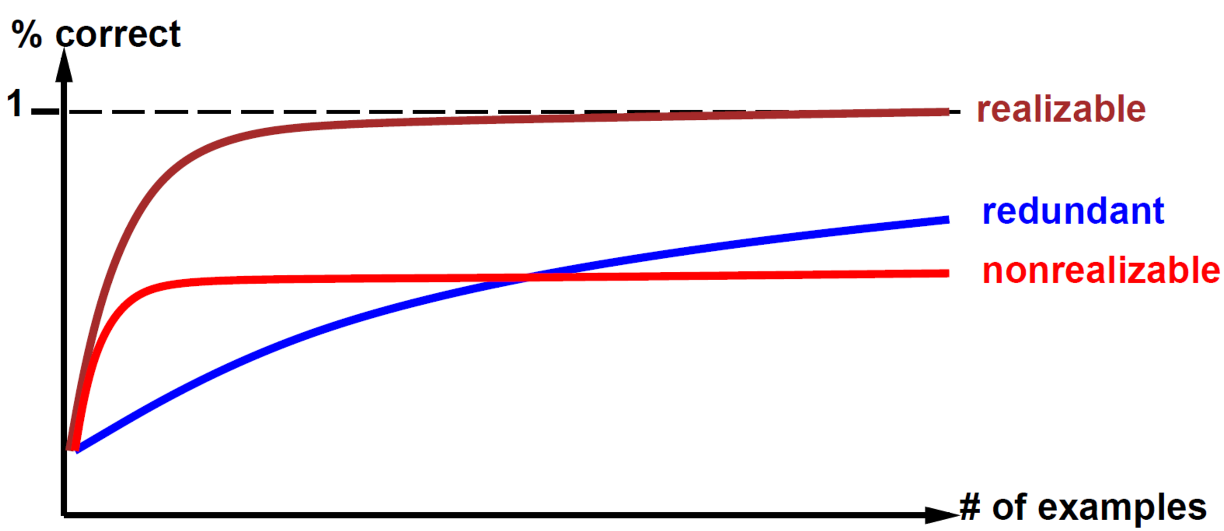
     (use same distribution over example space as training set)

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

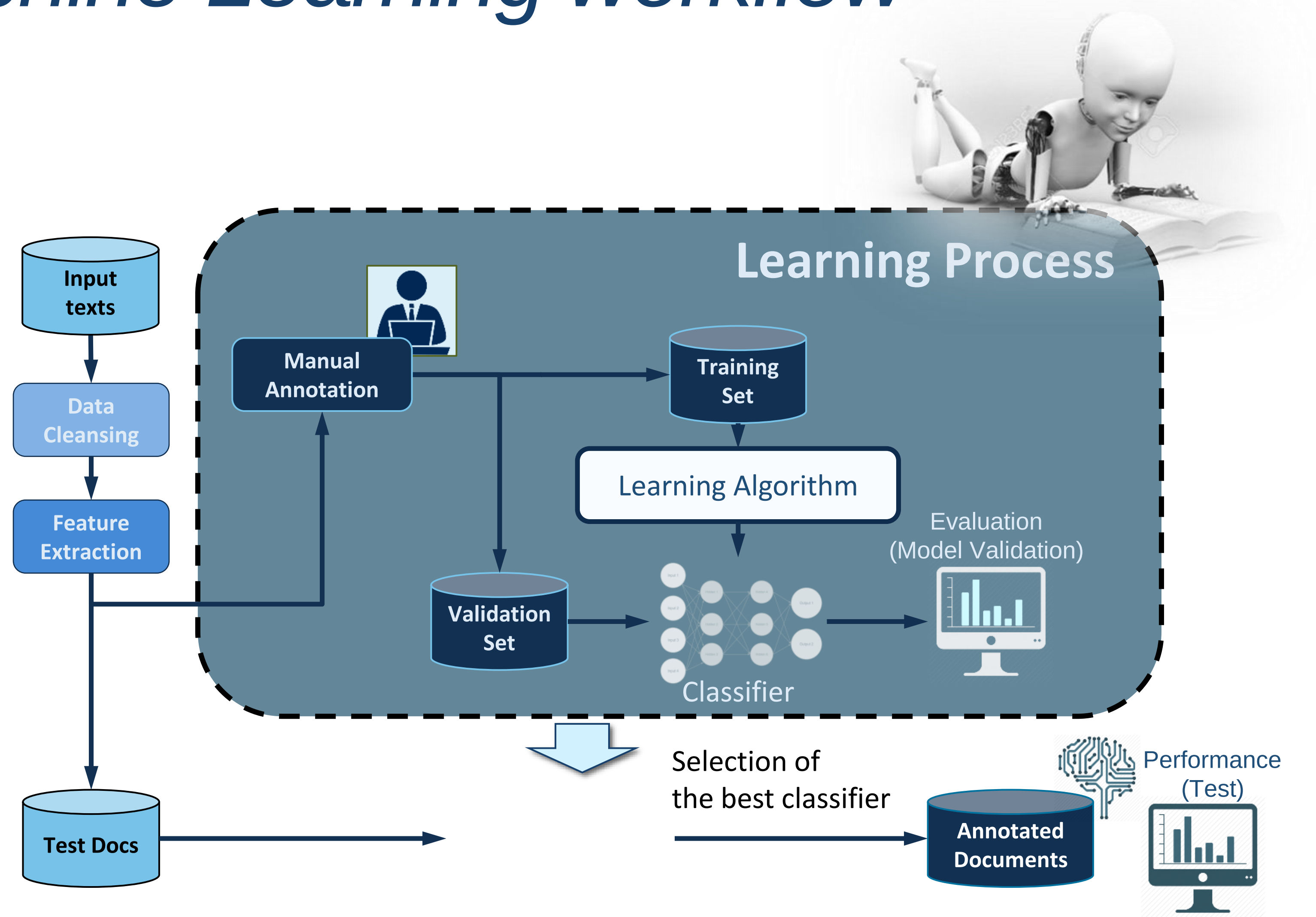


Performance measurements (2)

- 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 largenumber of irrelevant attributes are used



Machine Learning workflow



Evaluation of a ML system

- Performance Evaluation Metrics
  - Classifier Evaluation Metrics

- Tuning and Evaluation Methods

Classifier Evaluation: Confusion Matrix

		PREDICTED VALUE		
		Class A	Class B	Class C
ACTUAL VALUE	Class A	38	12	0
	Class B	5	43	2
	Class C	6	0	44

$$accuracy = \frac{\#correct\ classifications}{\#classifications} = \frac{38+43+44}{150} = 83.33\%$$

$$error\ rate = \frac{\#incorrect\ classifications}{\#classifications} = \frac{12+5+2+6}{150} = 16.67\%$$

Evaluation with skewed data

- Accuracy is not a suitable metric for task with imbalanced classes (for instance a spam detector)

		PREDICTED VALUE	
		Spam	Non-Spam
ACTUAL VALUE	Spam	0	10
	Non-Spam	0	9990

Very bad performance on the Spam class, that is the target of the classifier!! … nonetheless …

$$accuracy = \frac{\#correct\ classifications}{\#classifications} = \frac{9990}{10000} = 99.9\%$$

Single Class Metrics

		PREDICTED VALUE	
		Class C	Not Class C
ACTUAL VALUE	Class C	TP True Positive	FN False Negative
	Not Class C	FP False Positive	TN True Negative

$$precision = \frac{TP}{TP + FP}$$

what percentage of instances the classifier labeled as positive are actually positive?

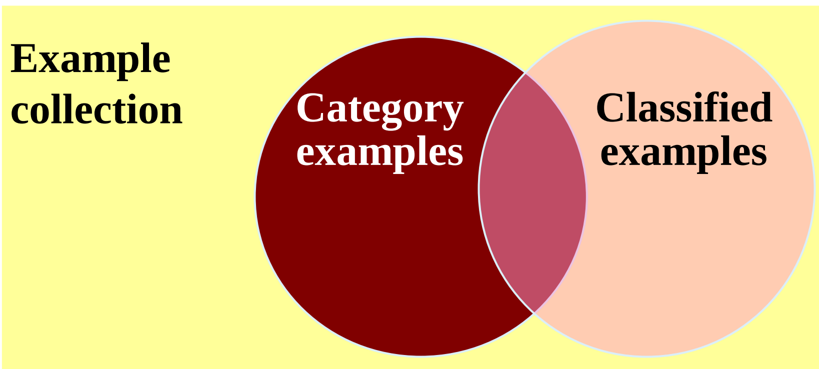
$$recall = \frac{TP}{TP + FN}$$

what percentage of positive instances did the classifier label as positive?

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

F-measure is the harmonic mean of precision and recall

Class-based evaluation



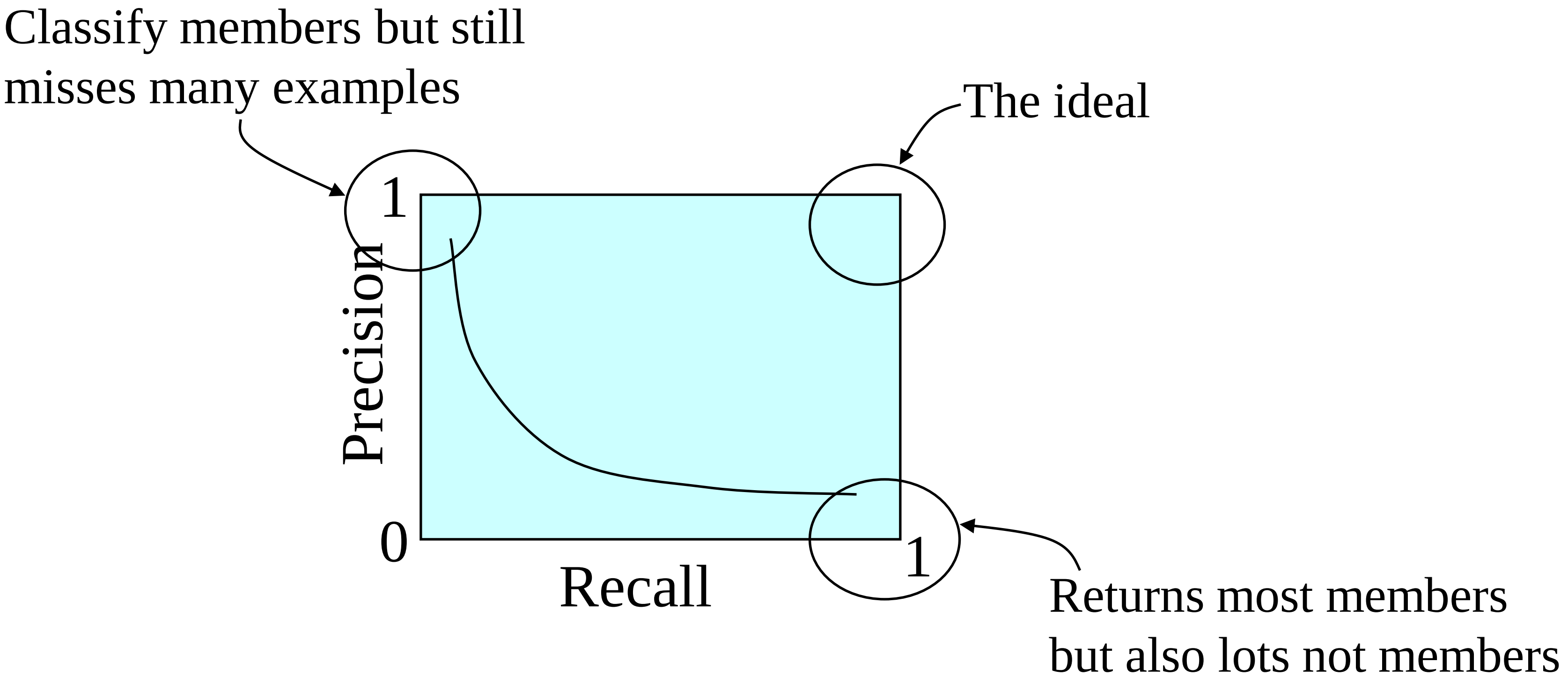
	Classified	Rejected
Not Members	Classified but not Members	Rejected & not Members
Members	Classified & Members	Rejected but Members

$$precision = \frac{\#\ of\ Members\ Classified}{\#\ of\ Members\ Classified\ + \#\ of\ Classified\ not\ Members}$$

$$recall = \frac{\#\ of\ Members\ Classified}{\#\ of\ Members\ Classified\ + \#\ of\ Rejected\ Members}$$

What about accuracy???

Trade-off between Precision and Recall



Other class based measures

Precision and Recall of C_i

- a, corrects (TP_i)
- b, mistakes (FP_i)
- c, instances of a $Class_i$ that are not actually retrieved, (FN_i)

The *Precision* and *Recall* are defined by the above counts:

$$Precision_i = \frac{a_i}{a_i + b_i}$$

$$Recall_i = \frac{a_i}{a_i + c_i}$$

		PREDICTED VALUE		
		Class A	Class B	Class C
ACTUAL VALUE	Class A	38	12	0
	Class B	5	43	2
	Class C	6	0	44

- $\text{Precision}_A = 38/(38+5+6) = 38/49$
- $\text{Recall}_A = 38/(38+12) = 38/50$

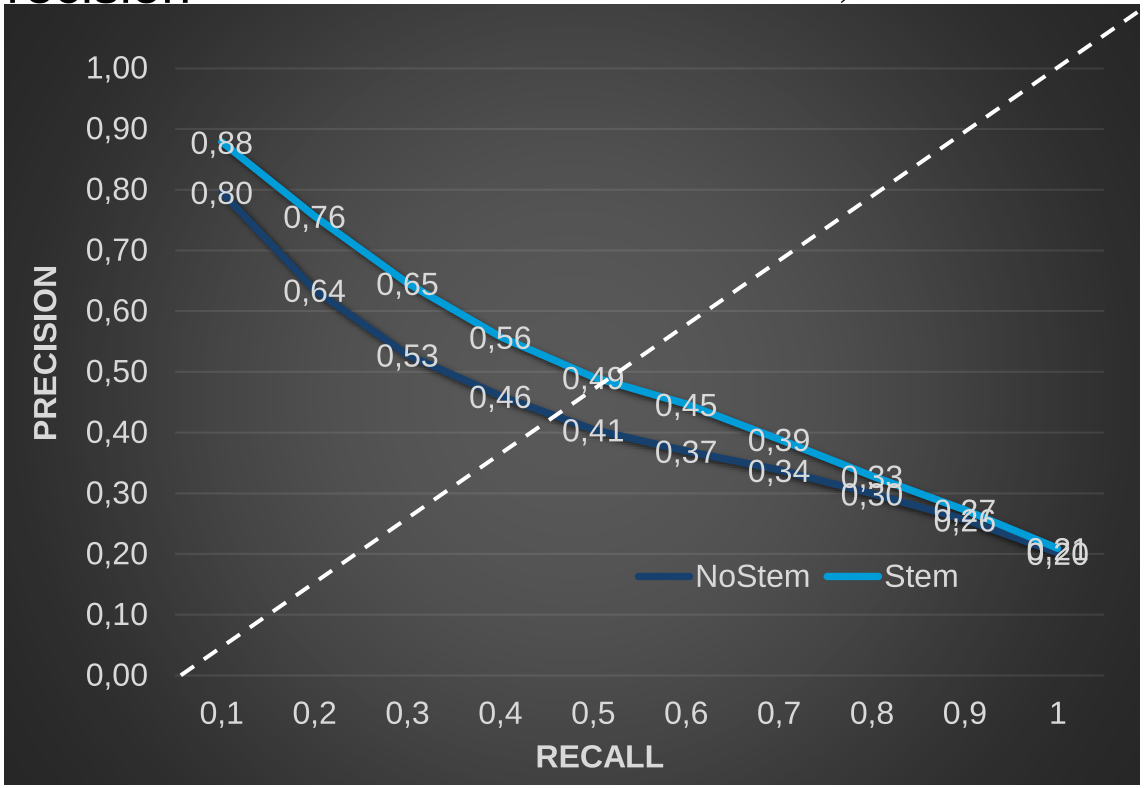
- $\text{Precision}_B = 43/(43+12) = 43/55$
- $\text{Recall}_C = 44/(44+6) = 44/50$

Performance Measurements (cont'd)

- Breakeven Point
  - Find thresholds for which
    Recall = Precision
  - Interpolation
- F-measure $$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$
  - Harmonic mean between precision and recall
- Global performance on more than two categories
  - Micro-average
    - The counts refer to classifiers
  - Macro-average (average measures over all categories)

Break-even Point

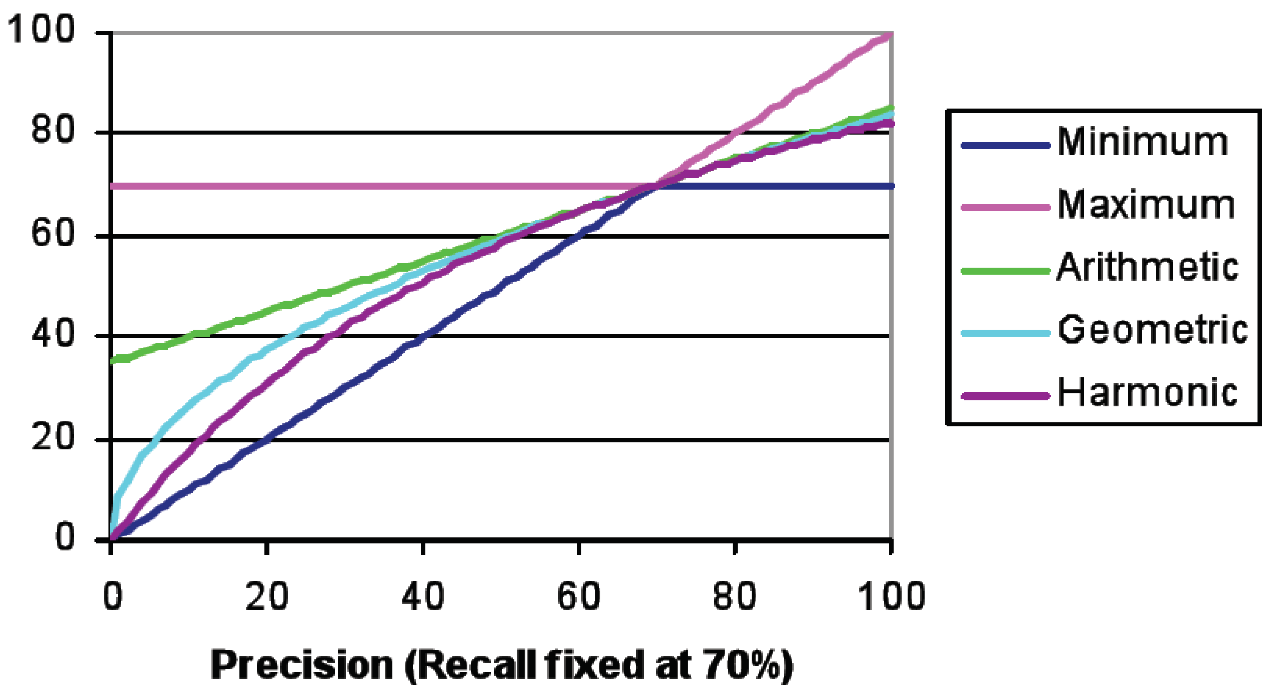
- The BEP is the interpolated estimate of the value for which Recall=Precision



- It shows the superiority of methods whose behavior is closer to the (1,1) ideal performance

Averaging Precision & Recall: A comparison

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$



$$\min(p, r)$$

$$\max(p, r)$$

$$arithM(p, r) = \frac{p + r}{2}$$

$$geomM(p, r) = \sqrt{p \cdot r}$$

$$harmM(p, r) = \frac{2}{p^{-1} +}$$

Averaging Precision & Recall: cross-categorical analysis

- Individual scores characterize the performance about each specific class
- Simple macro averaging can be applied to have

$$MPrecision = \sum_{i=1}^{n} Precision_i$$

$$MRecall \quad = \sum_{i=1}^{n} Recall_i$$

$$MF_1 = \frac{2 \cdot MPrecision \cdot MRecall}{MPrecision + MRecall}$$

F-measure e MicroAverages

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$\mu Precision = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} a_i + b_i}$$

$$\mu Recall = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} a_i + c_i}$$

$$\mu BEP = \frac{\mu Precision + \mu Recall}{2}$$

$$\mu f_1 = \frac{2 \times \mu Precision \times \mu Recall}{\mu Precision + \mu Recall}$$

		PREDICTED VALUE		
		Class A	Class B	Class C
ACTUAL VALUE	Class A	38	12	0
	Class B	5	43	2
	Class C	6	0	44

- $Precision_A = 38/(38+5+6) = 38/49$
- $Precision_B = 43/(43+12) = 43/55$
- Segue che:

$$Mprecision = 1/3(38/49 + 43/55 + \ldots)$$

		PREDICTED VALUE		
ACTUAL VALUE		Class A	Class B	Class C
	Class A	38	12	0
	Class B	5	43	2
	Class C	6	0	44

- $Precision_A = 38/(38+5+6)=38/49$
- $Precision_B = 43/(43+12)=43/55$
- Segue che:

  $\mu Precision=(38+43+44)/(38+43+44+11+12+2)$

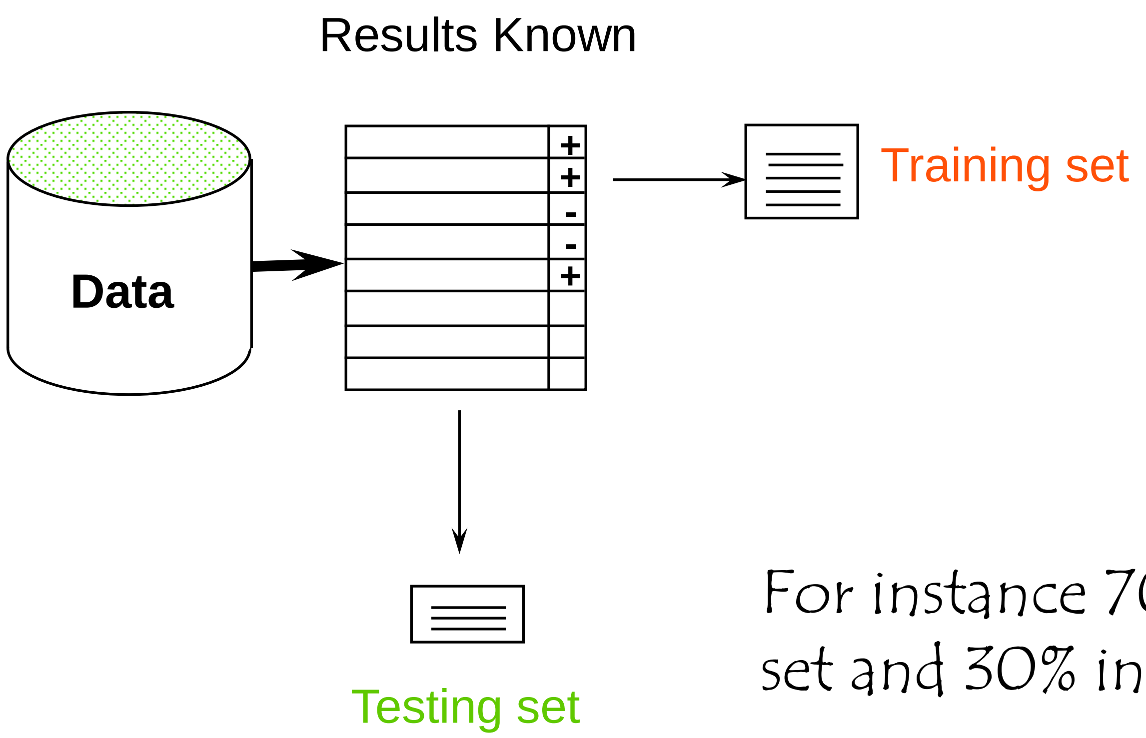
Overview

- Performance Evaluation Metrics
  - Classifier Evaluation Metrics
  - Information Retrieval Systems Evaluation Metrics
- **Tuning and Evaluation Methods**
- Error Diagnostics

Testing Data

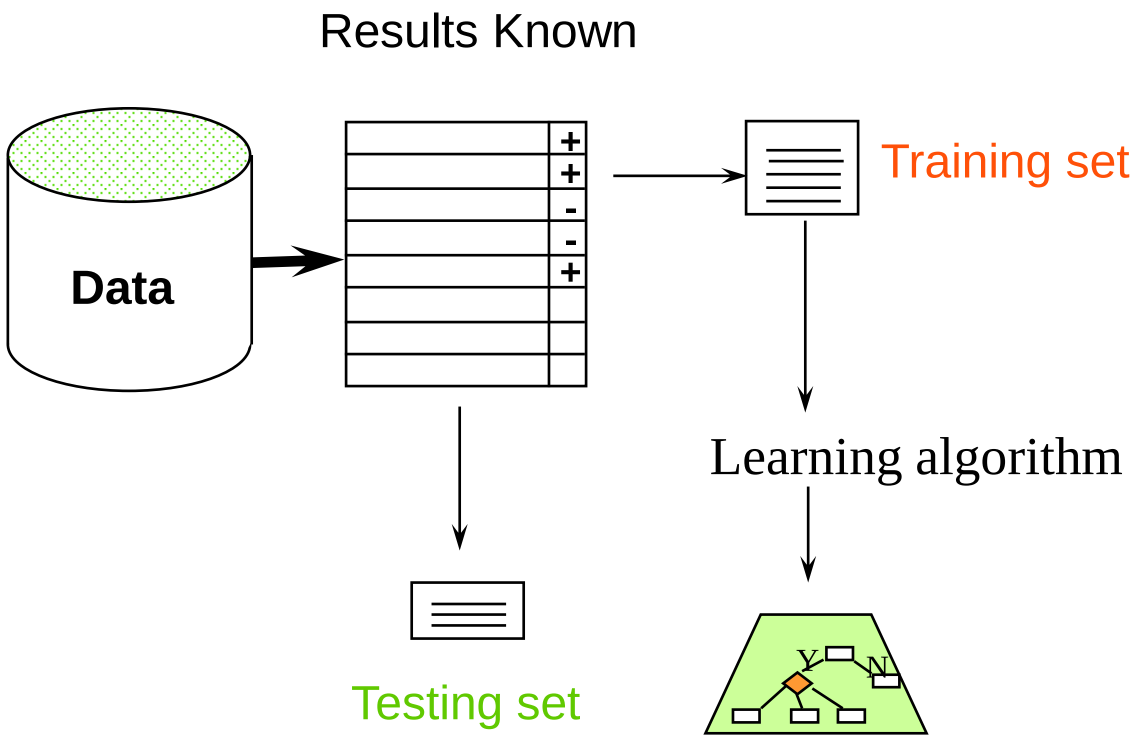
- To obtain a reliable estimation, test data **must be** instances **not** employed for the training step:
  - Error on the training data is *not* a good indicator of performance on future data, because new data will probably not be **exactly** the same as the training data!
  - *Overfitting* – fitting the training data too precisely - usually leads to poor results on new data
  - We want to evaluate how much accurate predictions of the model we learned are, and not other computational aspects (e.g. its memorization capability)

Step 1: dataset splitting

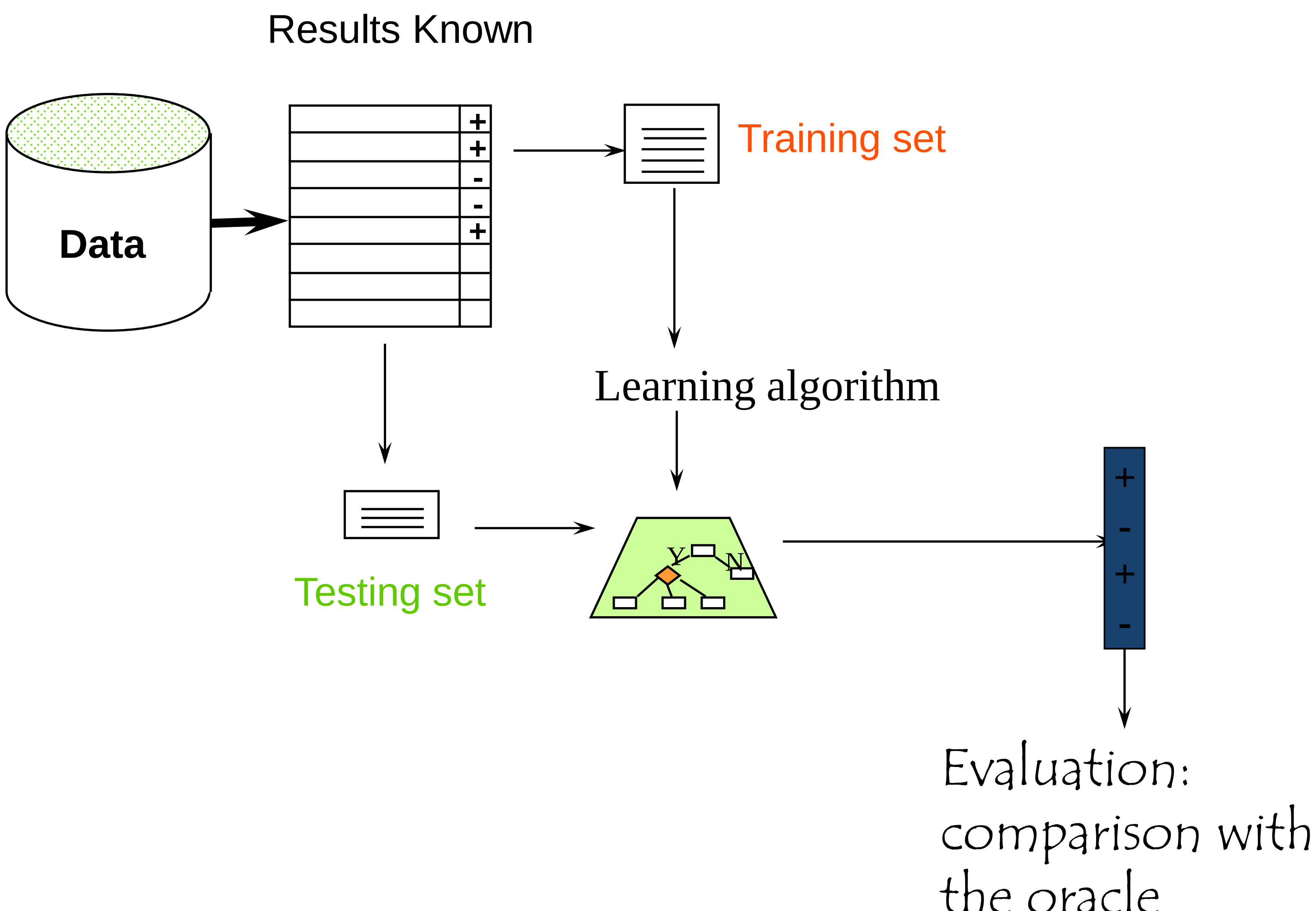


For instance 70% in the training set and 30% in the test set

Step 2: learning phase



Step 3: testing the model



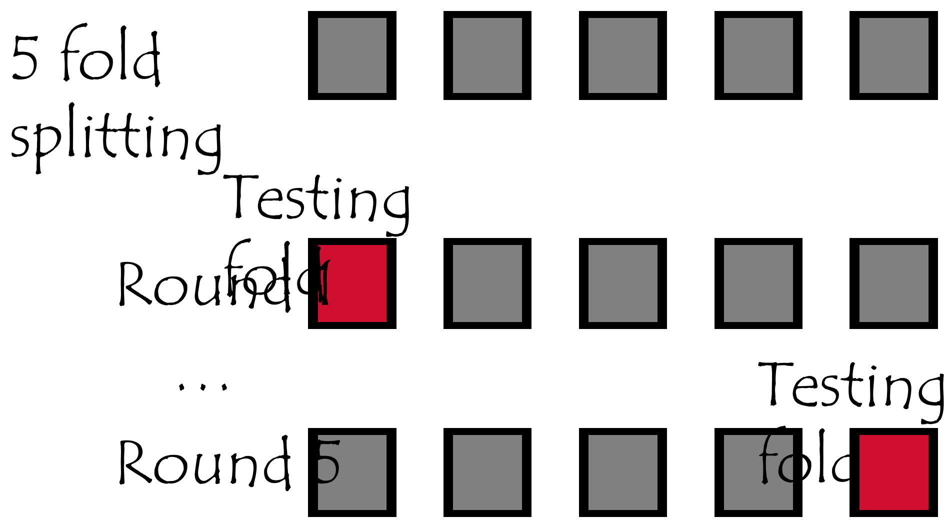
Evaluation on Few Data

- When data is scarce (totally or for a single class), a single evaluation process could not be enough representative
  - The testing set could contain too few instances to produce a reliable result

- SAMPLING: The evaluation process must be repeated with different splitting

N-Fold Cross Validation

- Data is split into *n* subsets of equal size
- Each subset in turn is used for testing and the remainders *n-1* for training
- The metrics estimated in each round are averaged

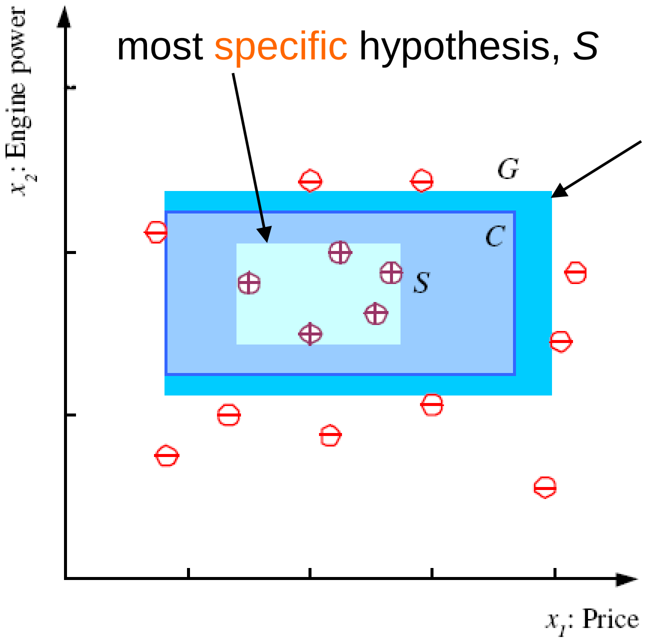


Tuning a Classifier

- Most of ML algorithms depends on some parameters
  - Examples: k in KNN, w_i in Rocchio, $p(w_i \mid c_j)$ for NB
- The best configuration must be choosen after a proper tuning stage:
  - A set of configurations must be established (for instance, $k=1,2,5,10,\ldots,50$)
  - Each configuration must be evaluated on a validation (or tuning) set

… short look at *model selection*

(Vector) Spaces, Functions and Learning



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

It makes up the version space

(Mitchell, 1997)

$$h \in \mathcal{H}$$

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 $\quad d_1$
  Model 2 $\quad d_2$
  Model 3 $\quad d_3$

  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

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

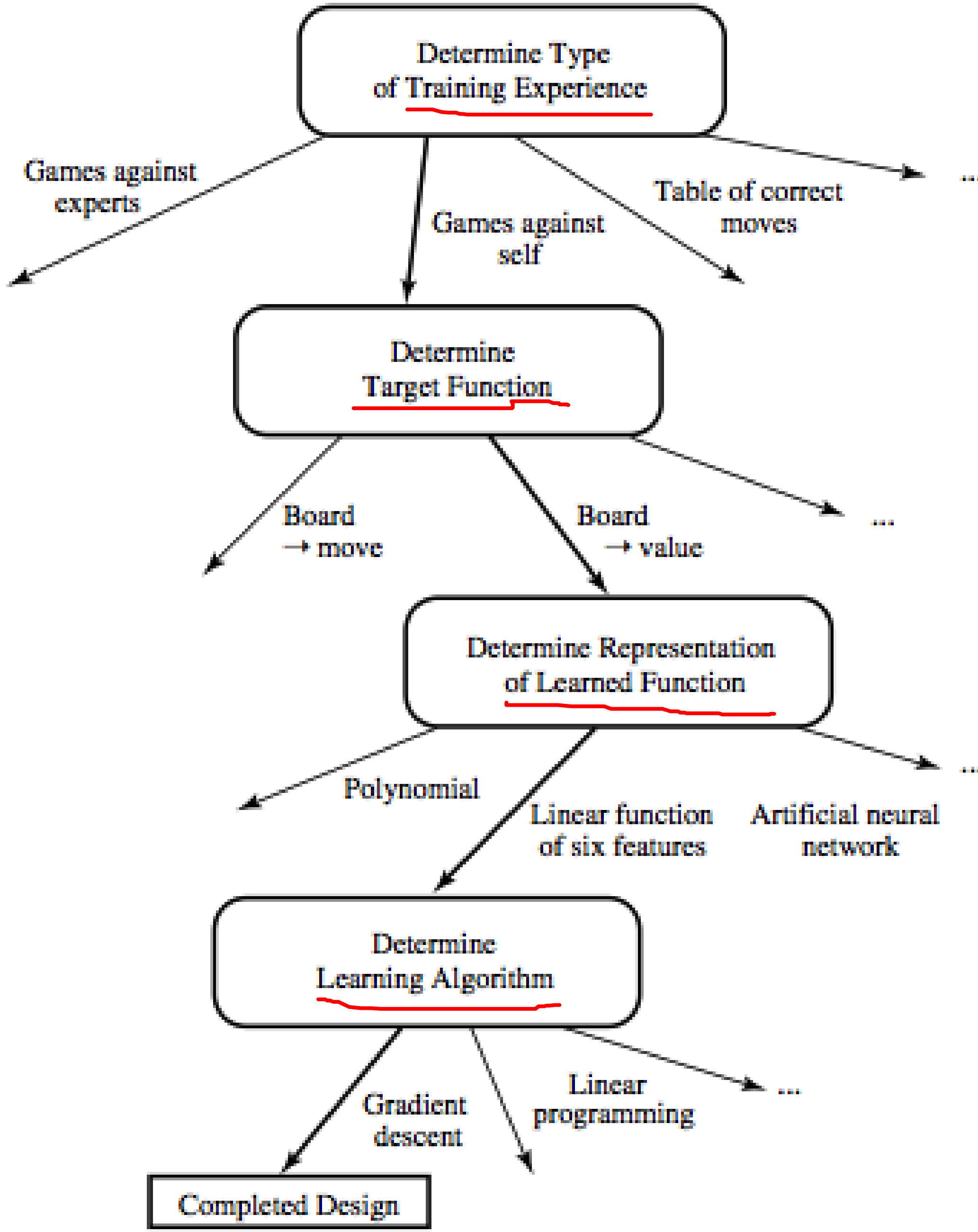
Alternatives to VC-dim-based model selection

- What could we do instead of the scheme below?
  - Cross-validation

i	f_i	TRAINERR	10-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			⍰
4	f_4			
5	f_5			
6	f_6			

Design of a learning system

Mitchell, 1997

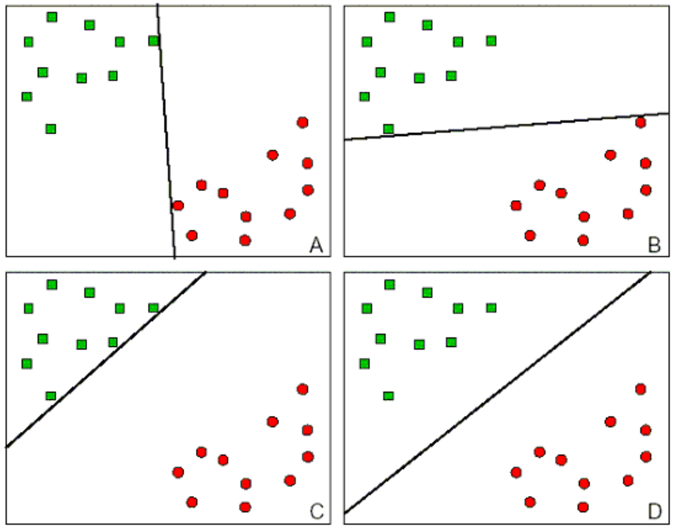


Machine Learning Tasks

- Supervised learning da esempi
  - Classification
    - Approcci dicriminativi
    - Approcci generative
    - Outlier and deviation detection
  - Regression
  - Dependency modeling
    - Discovery di Associazioni/Relazioni, Sommari, Inferenza/Causalità
  - Sequence Classification
    - Temporal learning
    - Trend analysis and change/anomaly detection
- Unsupervised learning
  - Clustering
  - Embedding ottimo: Enconding/Decoding
    - Representation Learning for Images
    - PreTraining as optimal encoding

Metodi di ML: selezione dei modelli

- Approcci discriminativi
  - Lineari



  - $h(x) = sign(\, \mathbf{W} \cdot \mathbf{x} + b)$

- Approcci probabilistici
  - Stima delle probabilità $p(\mathcal{C}_k|\mathbf{x})$ attraverso un training set
  - Modello generativo ed uso della inversione Bayesiana

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

Riferimenti Bibliografici

- *AIMA,* Chapter 18, 18.1-18.4, 18.6-18.7, 18.11
- READING. *Machine Learning*, Tom Mitchell, Mc Graw-Hill International Editions, 1997 (Cap 3).
- L'Algoritmo Definitivo, Pedro Domingos, Bollati Boringhieri, 2016