Sapienza
Università di Roma

AI AND MACHINE LEARNING
A QUICK INTRODUCTION …

Methods in experimental particle physics
Roma 13.6.2019
S. Giagu

REFERENCES AND FURTHER READING ...

- Machine Learning and Deep Learning:
  - Pattern Classification: R.O. Duda, P.R. Hart, D.G. Stork, (2nd ed.) J.Wiley&Sons
  - Stat. Pattern Recognition: A. Webb, (3rd ed.), J.Wiley&Sons
  - Decision Forests for Computer Visions and Medical Image Analysis: A.Criminisi, J.Shotton, Springer
  - Deep Learning: I.Goodfellow, Y.Bengio, A.Courville, The MIT Press

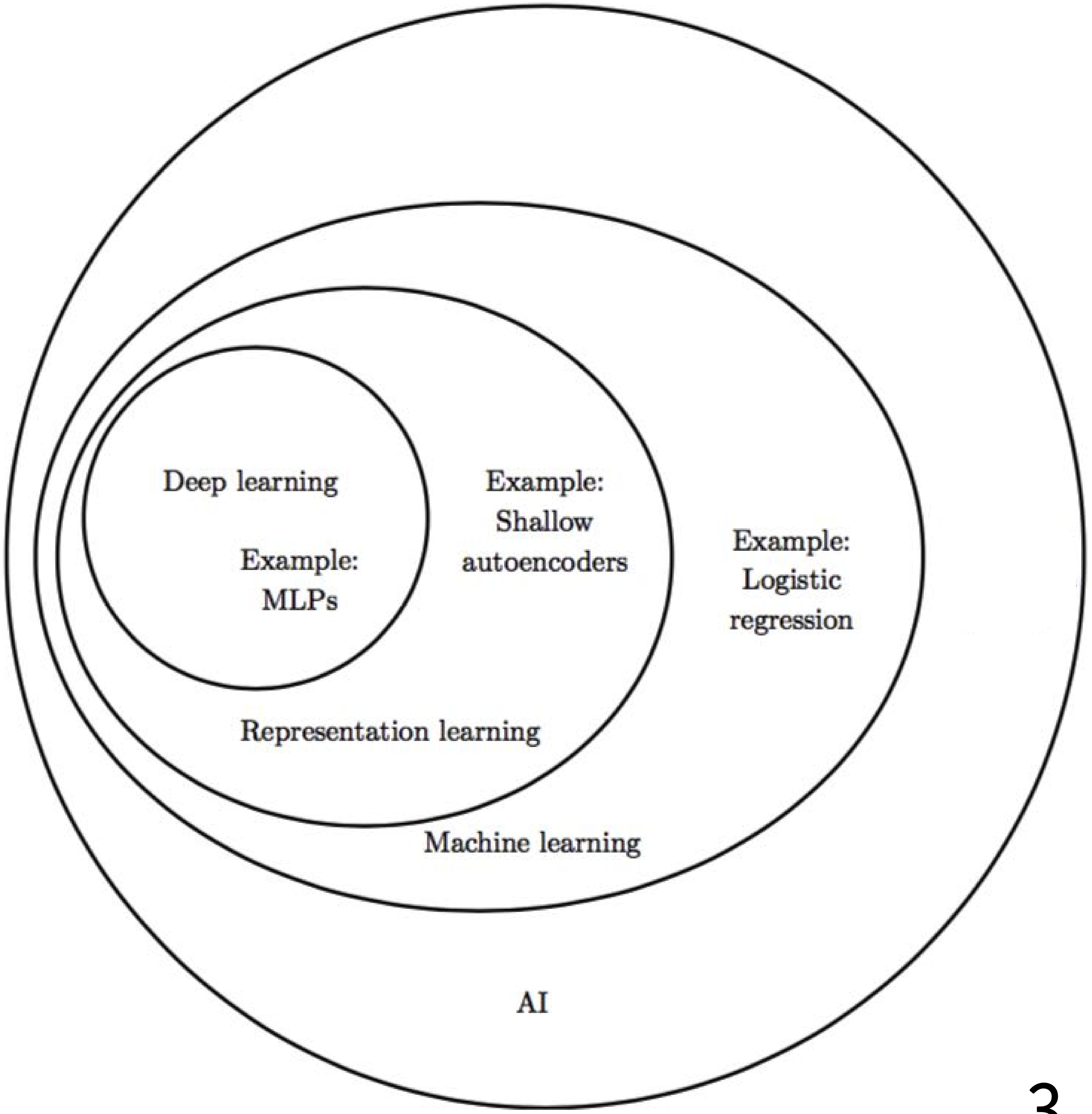
- Artificial Intelligence (introductive):
  - Artificial Intelligence: A Modern Approach: P.Norvig. (free on web)
  - Life 3.0 – Being Human in the Age of Artificial Intelligence: M. Tegmark
  - Fundamental Algorithms: 1 (Artificial Intelligence for Humans): J.Heaton (more advanced)

- Tools/frameworks:
  - Keras & TensorFlow: https://keras.io, https://www.tensorflow.org
  - XGBoost: https://xgboost.ai
  - Scikit-learn: https://scikit-learn.org/stable/
  - PyTorch: https://pytorch.org

INTRODUCTION

- What Machine Learning means?

  - ML is part of a larger research filed called Artificial Intelligence (AI) focused in the attempt to automatize intellectual tasks that are generally performed by humans



AI

- the AI concept and the study and development of ML algorithms used in AI systems started in the early 50', but it is only in the last ~10 years that AI applications are spreading exponentially in the society outside the basic and accademico research field

- This acceleration motivated by three parallel developments:
    - better algorithms (Machine & Deep Learning)
    - higher computing power (GPUs/TPUs/HPCs)
    - ability of the technological and industrial sectors to record and make accessible huge amounts of data/information (grid, clouds)

MACHINE LEARNING

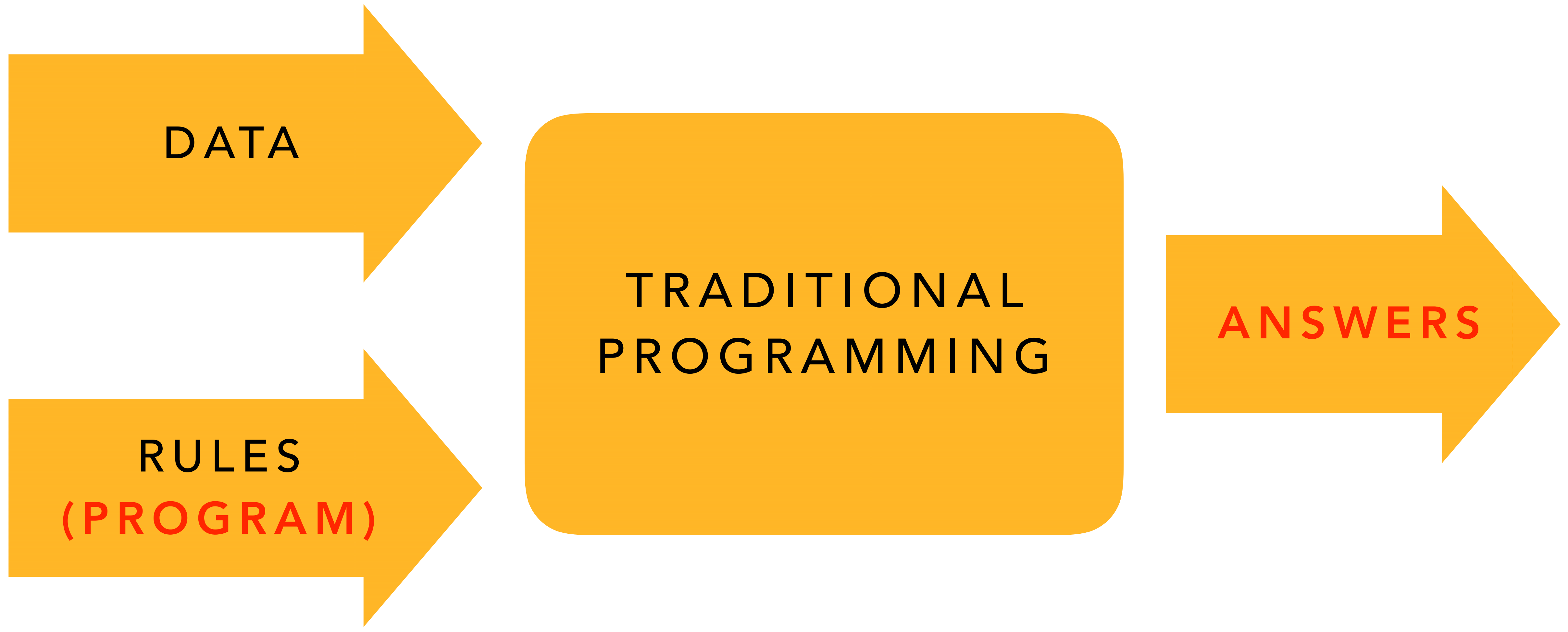
- Original definition (Arthur Samuel, 1959):

  Computational methods (algorithms) able to emulate the typical human, or animal, behaviour of learning based on the experience (i.e. learning from examples), w/o being explicitly programmed

  ML algorithms are meant to solve that class of problems (like image or language recognition) that cannot be simply described with a set of formal mathematical rules (equations) and so too complex to be resolved by a traditional computational algorithm

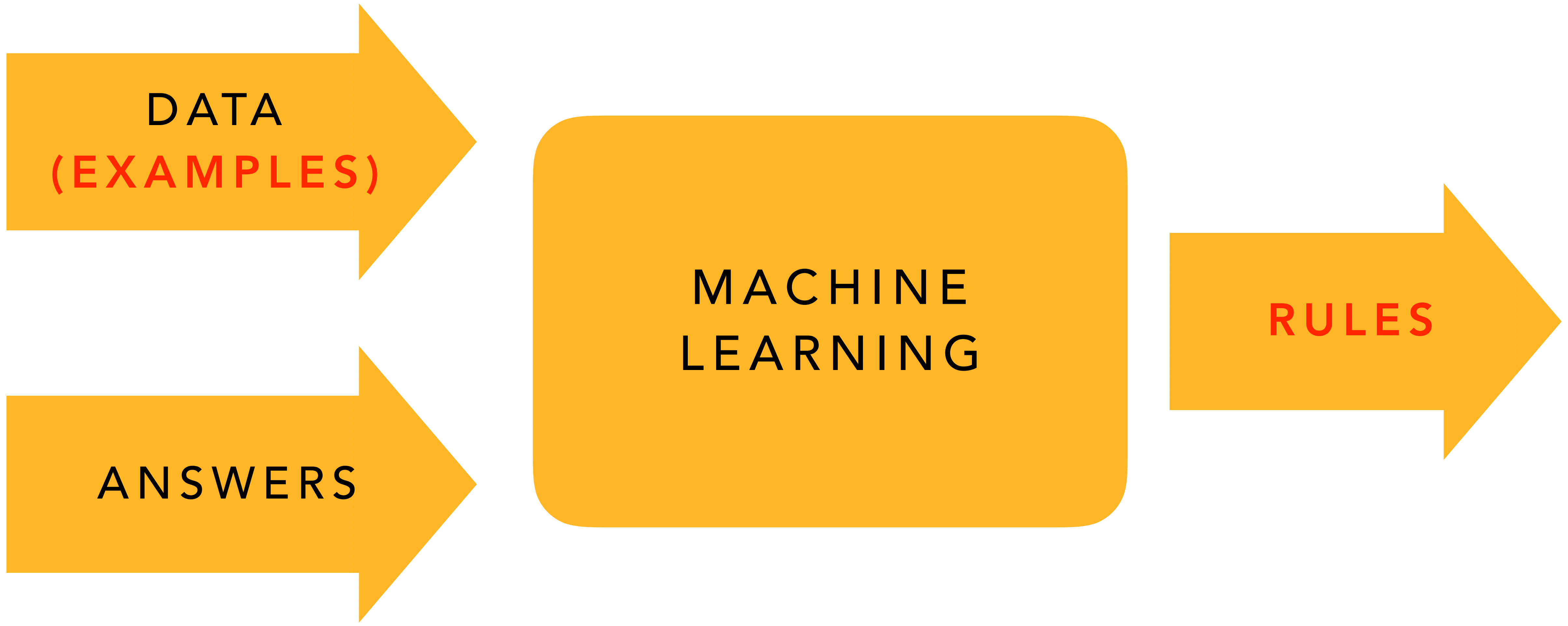
MACHINE LEARNING VS TRADITIONAL COMPUTATION

- Traditional computation (symbolic AI): the programmer (human) design and load a set of rules (program) in the processor together a set of data that are analysed accordingly the set of rules to output an answer to the problem we want to solve



MACHINE LEARNING VS TRADITIONAL COMPUTATION

- <u>ML</u>: the programmer present to the processor both the data set and the set of answers expected for that data set. The algorithm output a set of rules that can then applied to indipendenti datasets to get the original answers



a ML system is “trained” not programmed …

- is feed with a set of relevant examples gli vengono presentati un certo numero di esempi significativi
- try to find statistical structures in these examples (we assume these structures exist), that eventually will allow the algorithm to learn the rules needed to learn to perform a certain task

A MODERN DEFINITION (MITCHELL, 1998)

- an algorithm is said to learn from experience (E) with respect to some class of tasks (T) and a performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E

- Task T: are described in terms of how the ML algorithm should process the example E
  - typical ML tasks:
    - classification ($f:R^n \rightarrow \{1,...,k\}$), regression ($f:R^n \rightarrow R^m$), images segmentation, transcription (ex. OCR), conversion of sequences of symbols (automatic translation), anomaly detection, synthesis/sampling (es. generators), de-noising, …

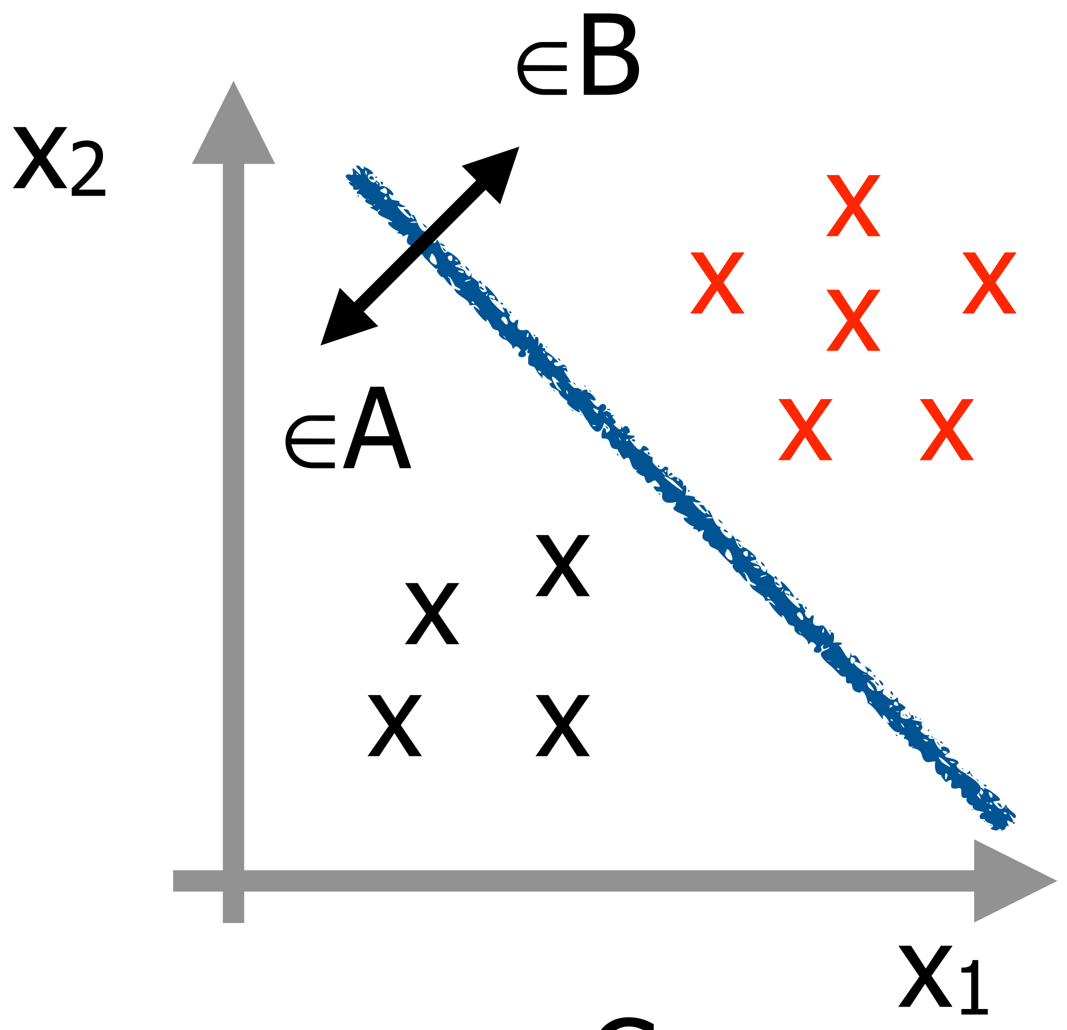
- Example/Experience E:
  - represent the set of empirical information from which the algorithm learn
    - training set (i.e. the data)
    - prior knowledge: invariants, correlations, …

- Performance measure P: to evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance. Usually this performance measure P is specific to the task T being carried out by the system
  - accuracy (fraction fo examples for which the algorithm produce the correct output), error rate, statistical costs, ROC, AUC, …
  - must be always evaluated in a statistically independent data set (test sample)

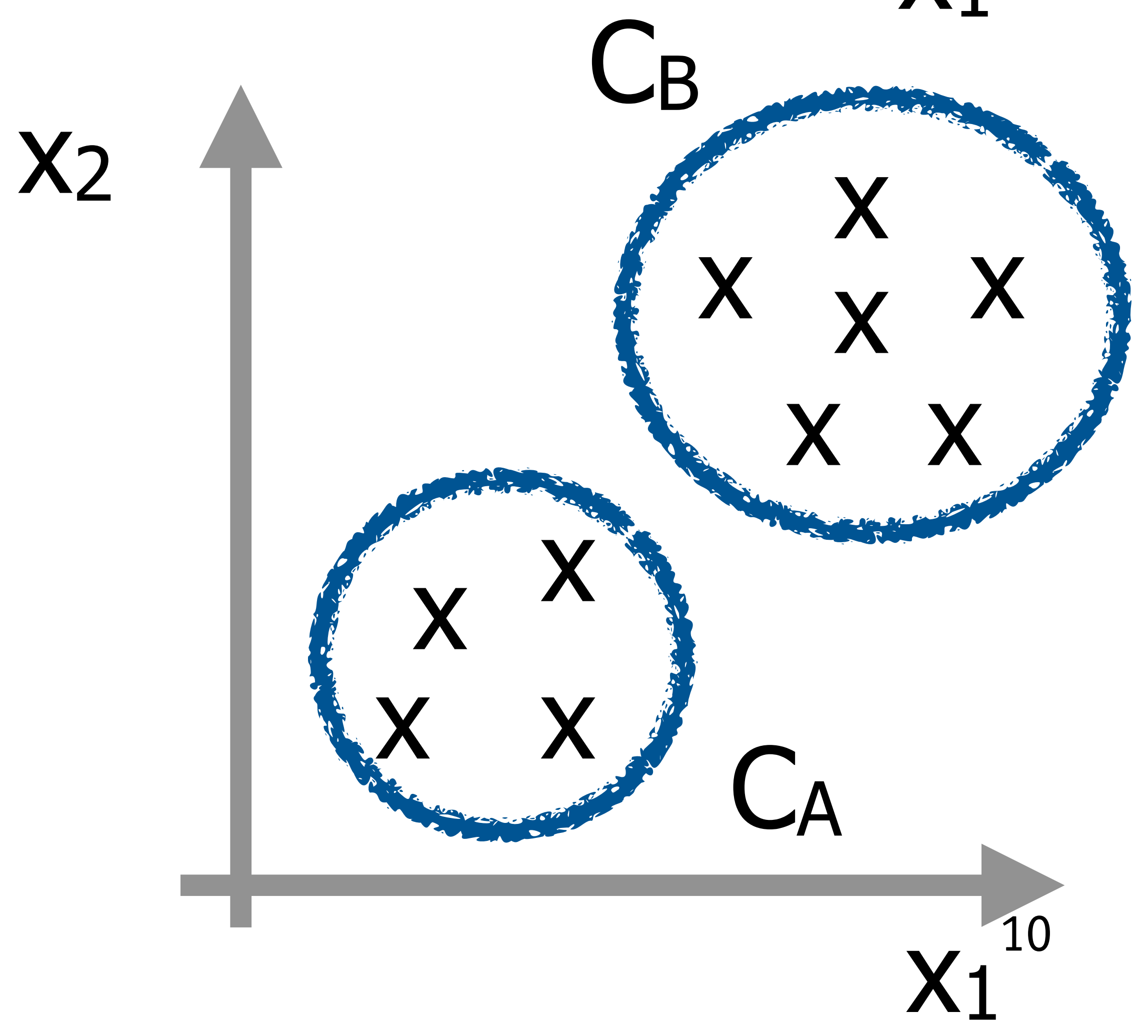
LEARNING PARADIGMS

- Learning algorithms can be divided in different categories that defines which kind of experience is permitted during the training process

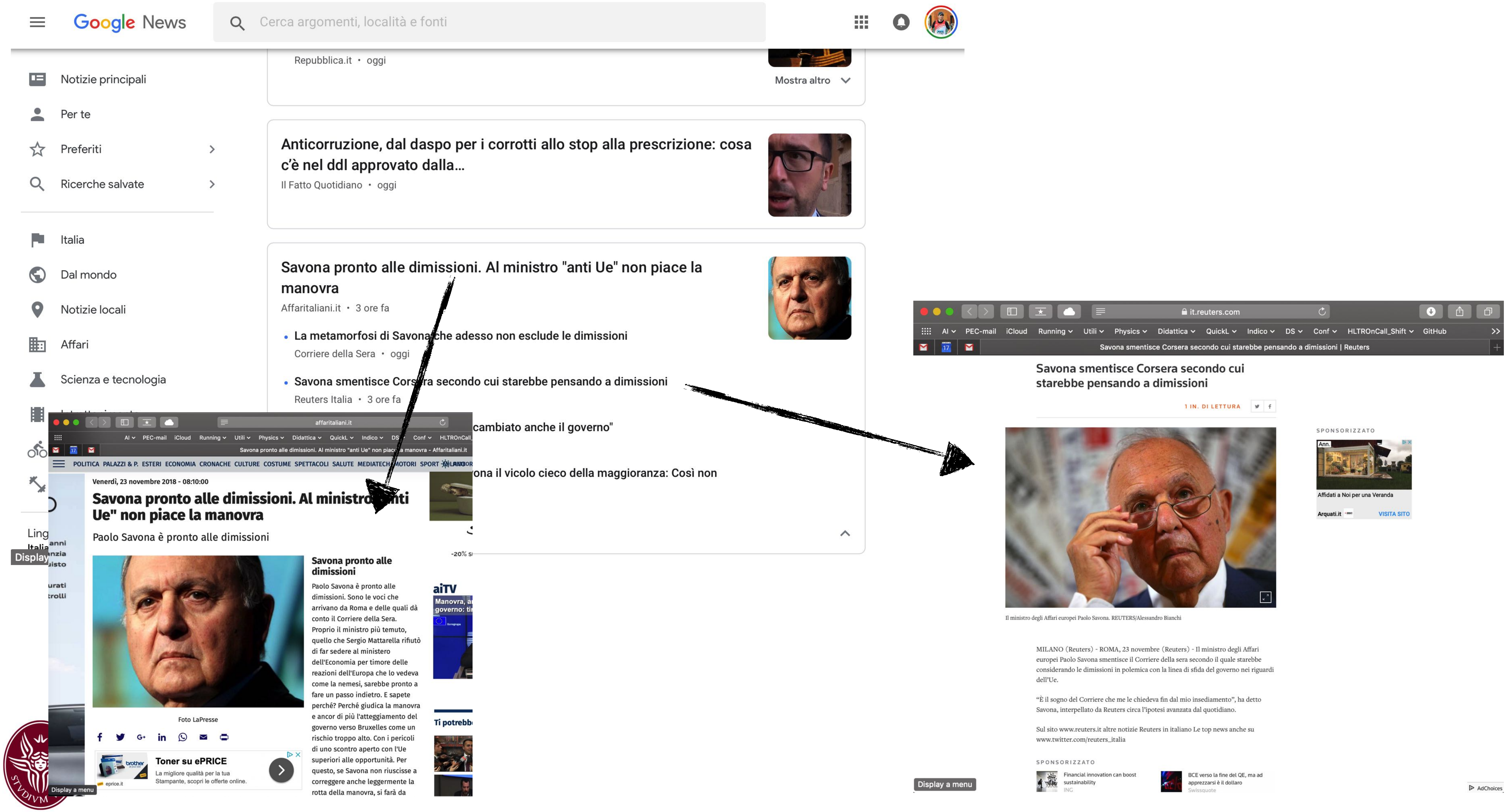
- supervised learning (i.e. there is a teacher):
  - for each example of the training set is provided the true answer (for example the corresponding class) called label
  - Typical target of the training process: to minimise the classification error or the accuracy



- unsupervised (or better: auto-supervised) learning:
  - no explicit information on the true answer for the training set examples is given
  - typical target of the training process: create groups / clusters of the input objects, generally on the base of similarity criteria



UNSUPERVISED LEARNING ALGORITHM EXAMPLE: GOOGLE NEWS

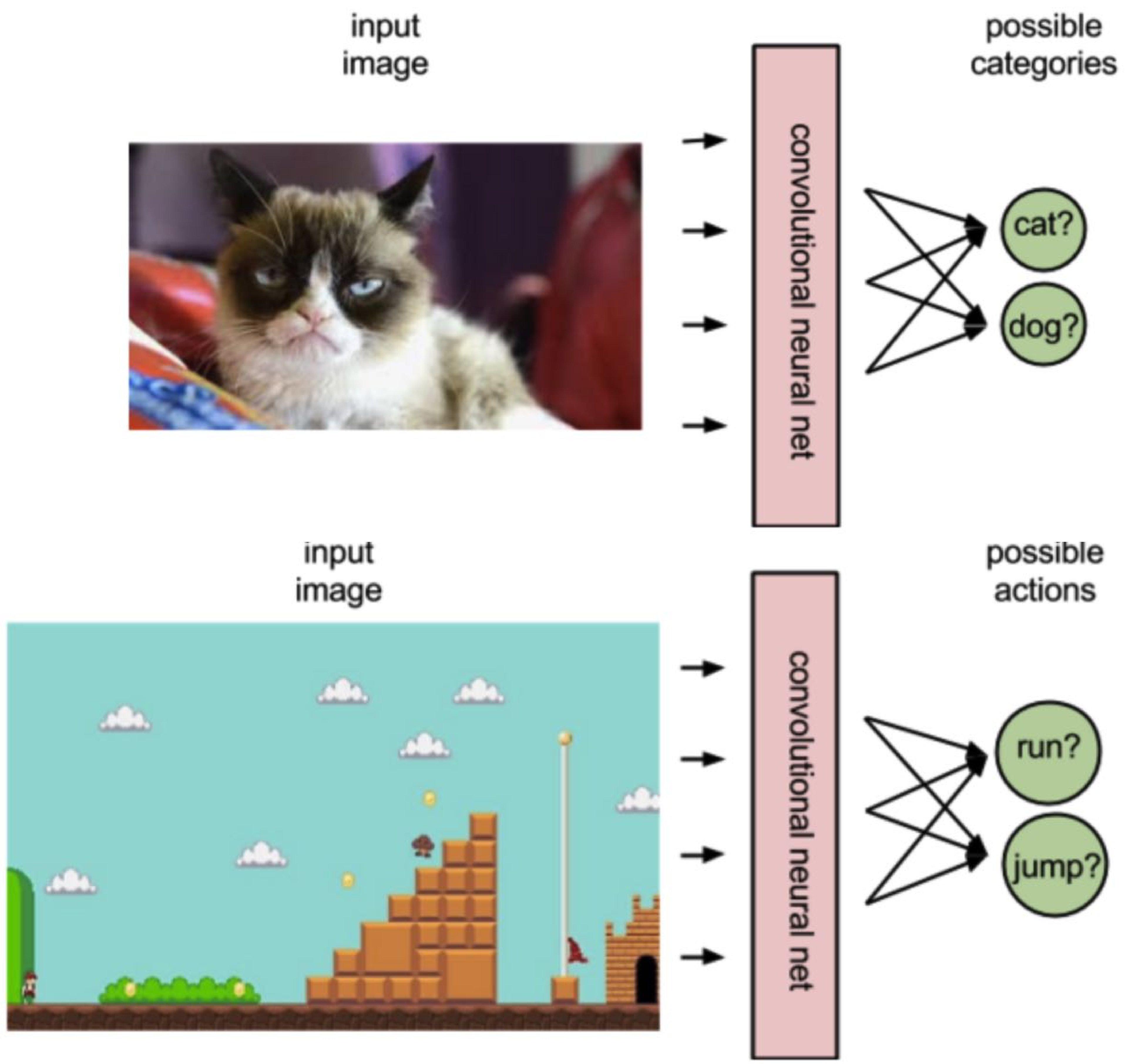


- Reinforcement learning:

inspired by behavioral psychology: is not used a fixed set of examples/experiences, but the algorithms adapts to teh ambient with which interacts via a continuous feedback between system and examples and through the distribution of a sort of reward (reinforce) that acts on the performance measure P

Solve the complex problem of relating instantaneous actions with the effect that they may produce at a later time

example: to maximise the score in a game that develop over multiple moves



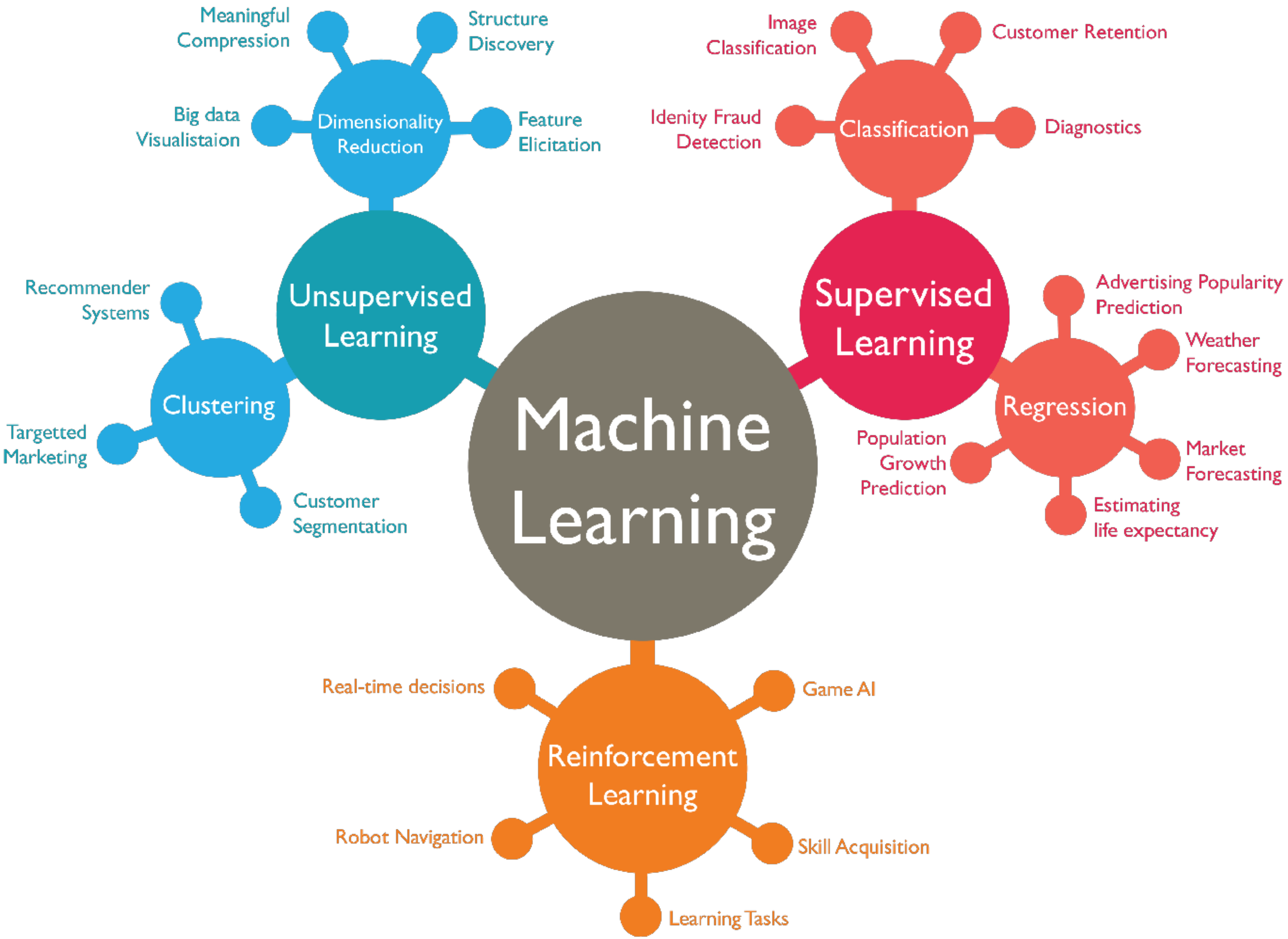
Supervised ConvNet

Associate a label to an image

Convolutional agent

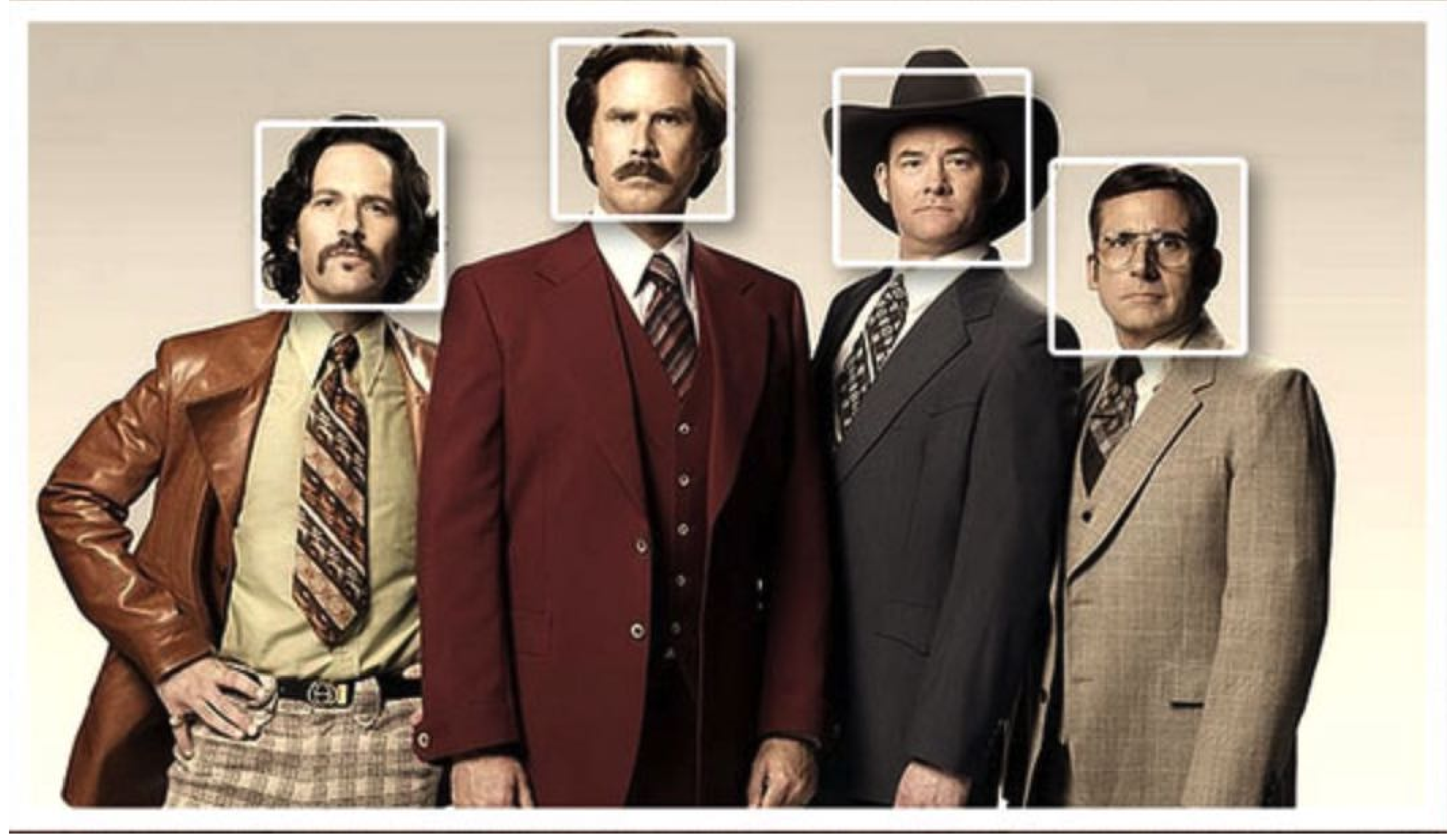
Maps a state to the best possibile action

ML: LEARNIGN PARADIGMS AND TASKS

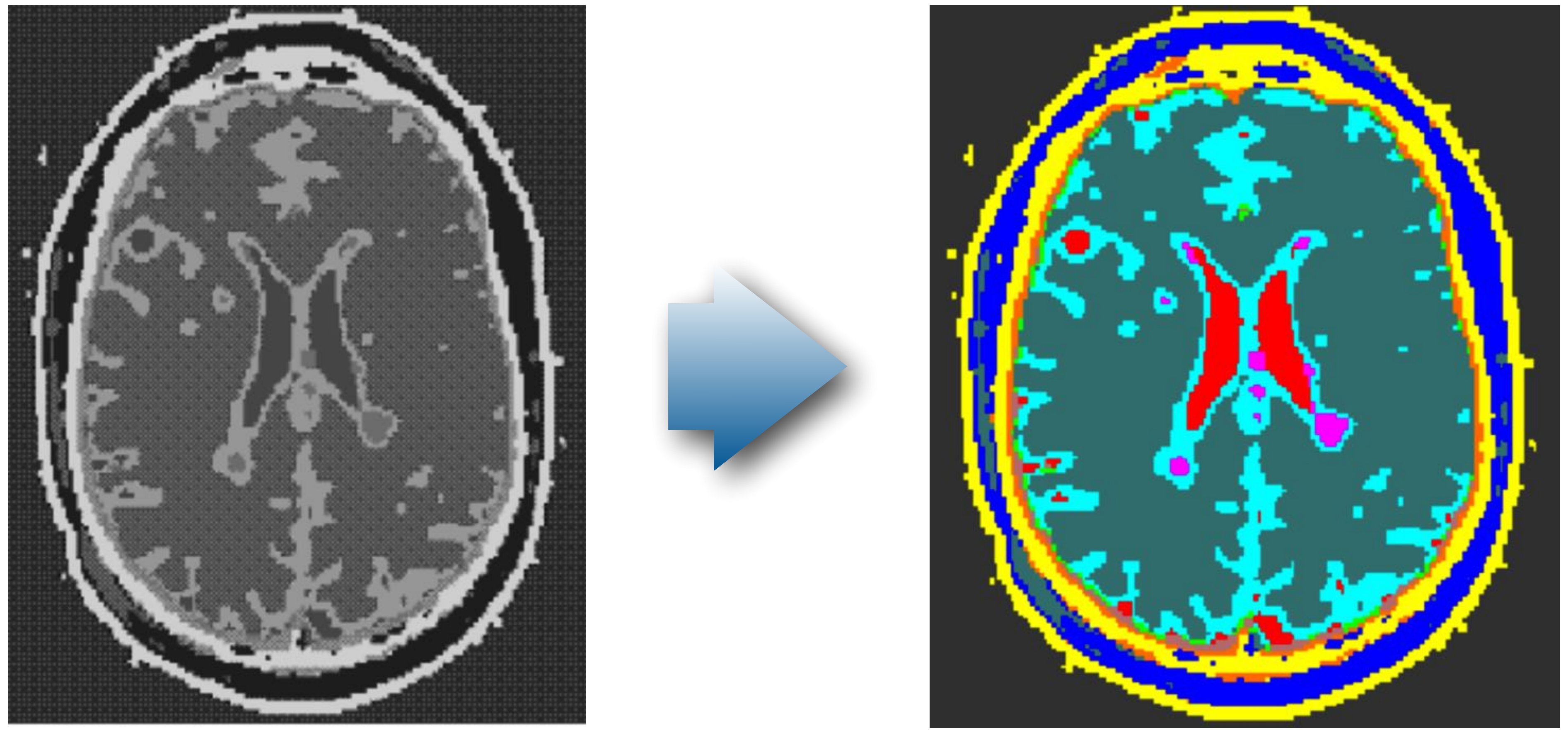


AI/ML APPLICATIONS EXAMPLES

Face/Object Detection:
- static: ex. facebook photos
- real time: cameras, autonomous driving systems
- experience: portion of images
- task: face or not-face



Medical Image Detection e Segmentation:
- experience: images (list of pixels)
- task: identify different biological tissues, disomogeneities ...

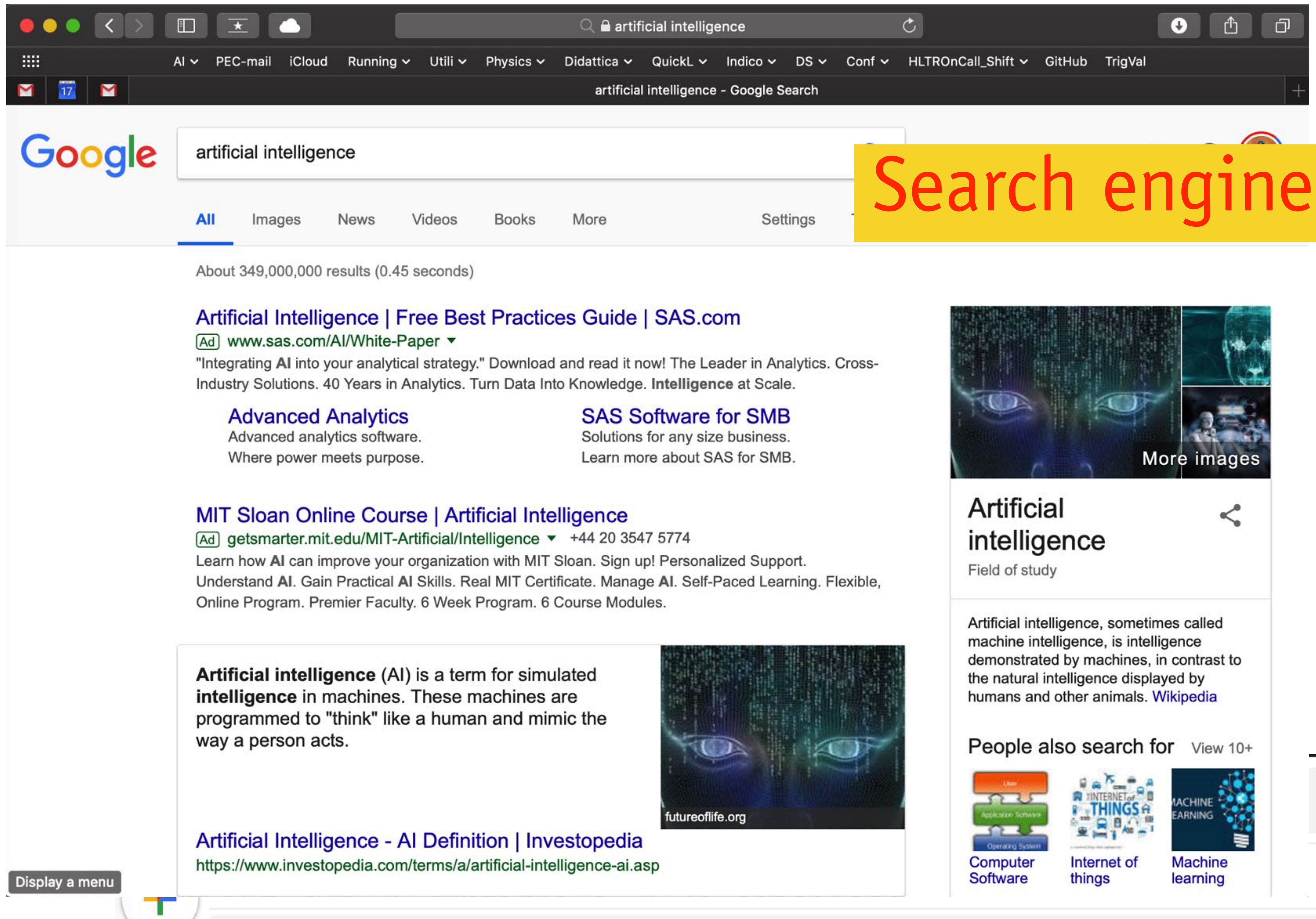


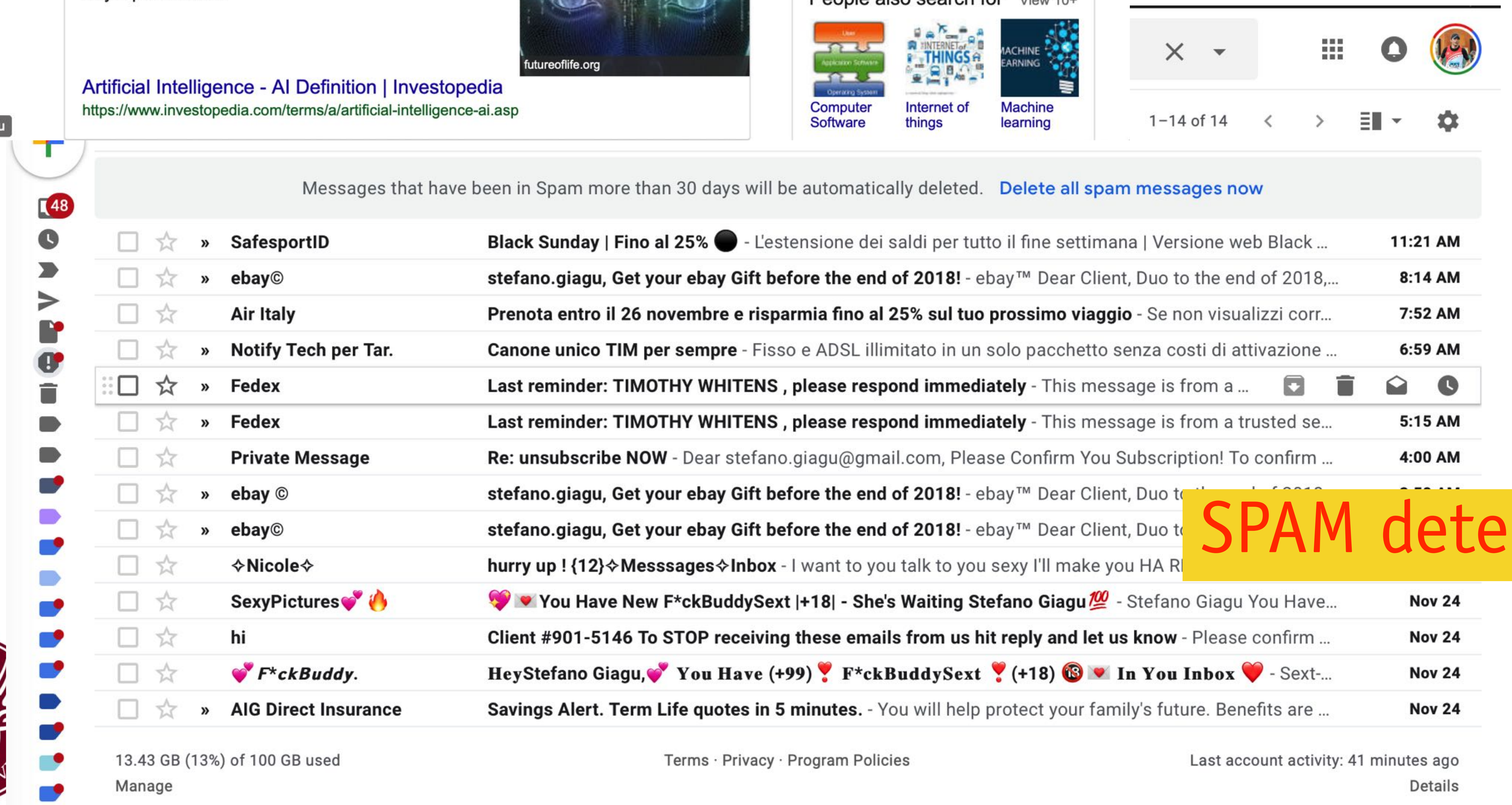
Voice recognition:
- experience: acoustical signals
- task: identify phonemes

ma-chin-le-ar-nin-g

AI/ML APPLICATIONS EXAMPLES

Search engines





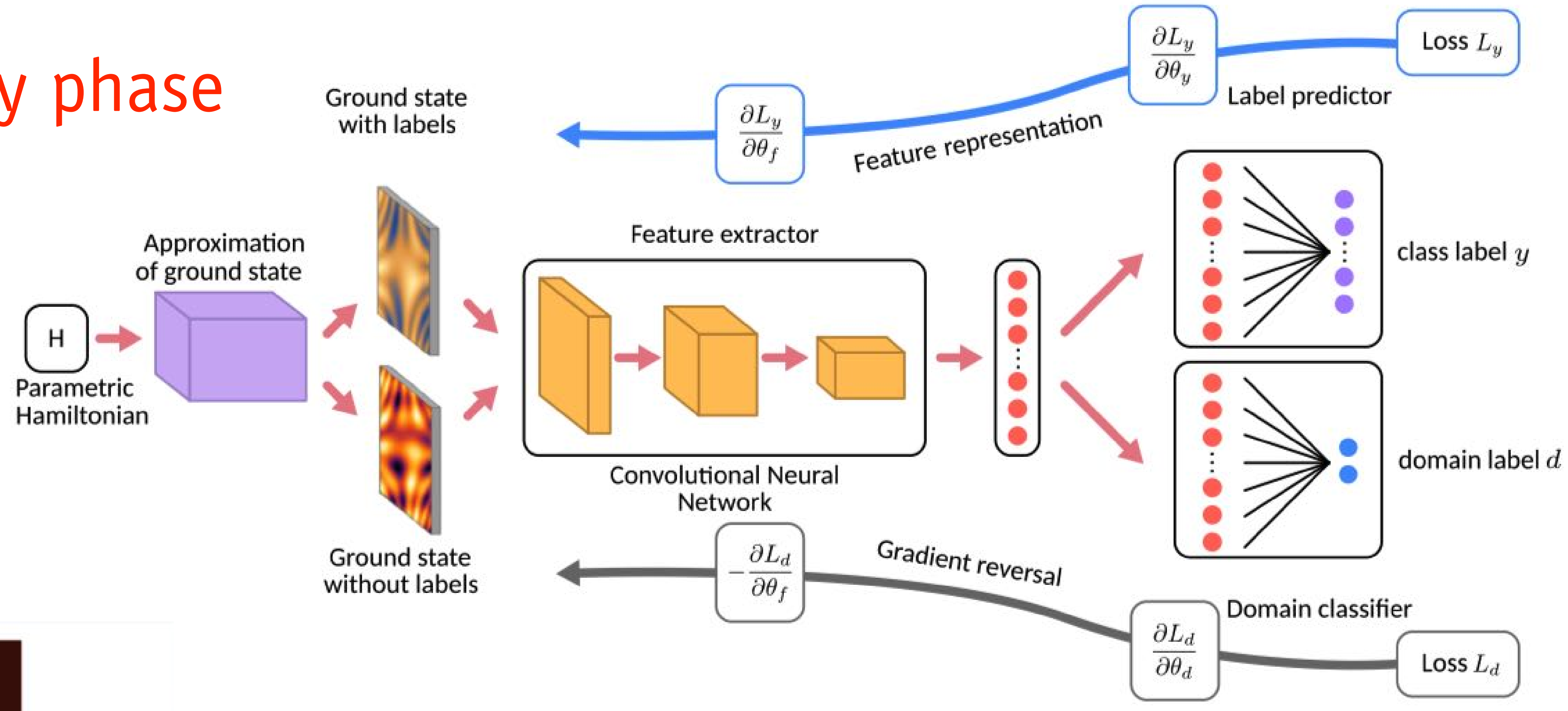
SPAM detection

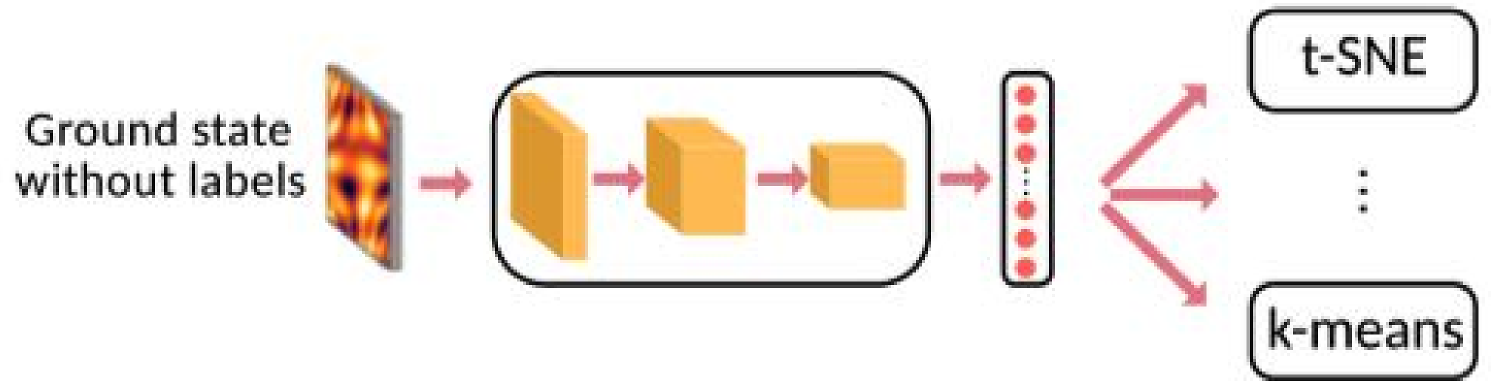
Autonomous drive

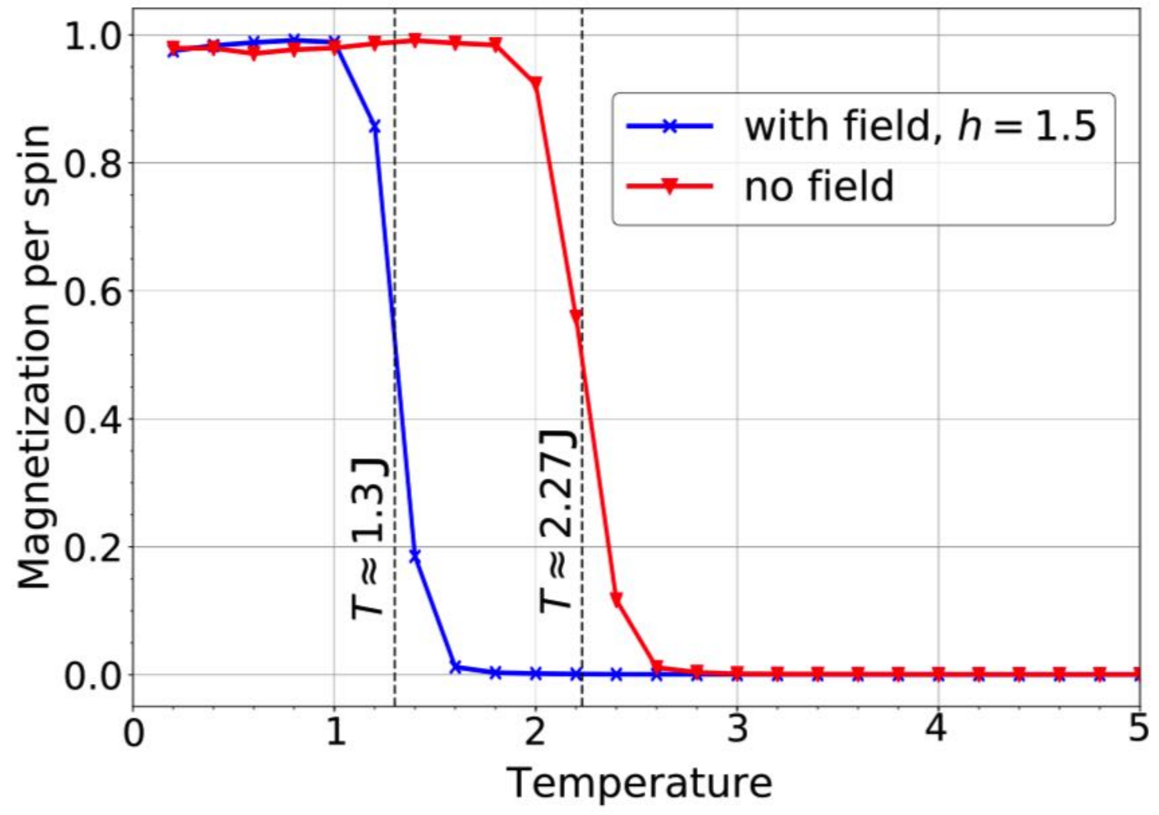
Autonomous Drones

APPLICATIONS IN PHYSICS

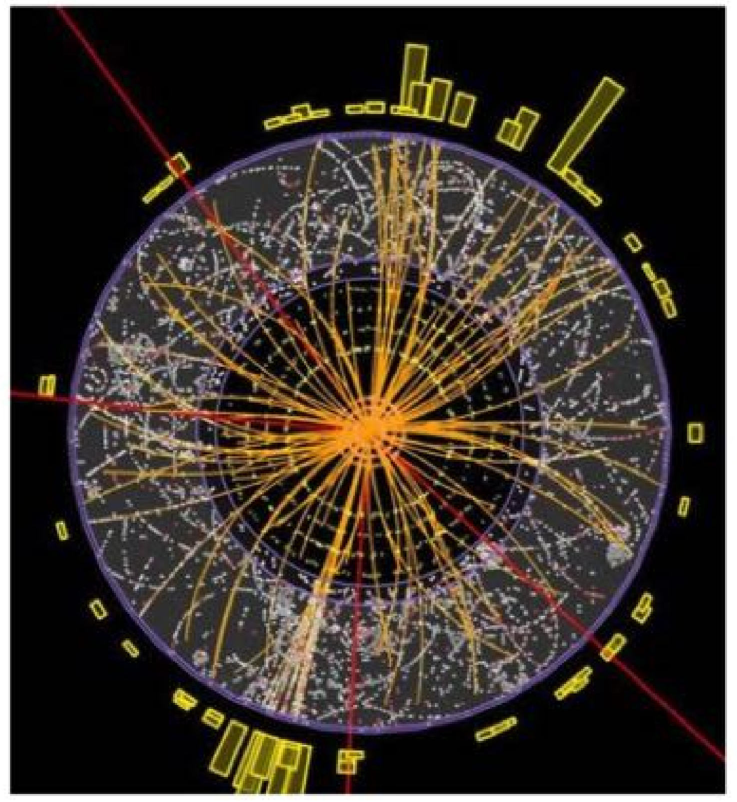
Adversarial CNN to identify phase transitions in matter



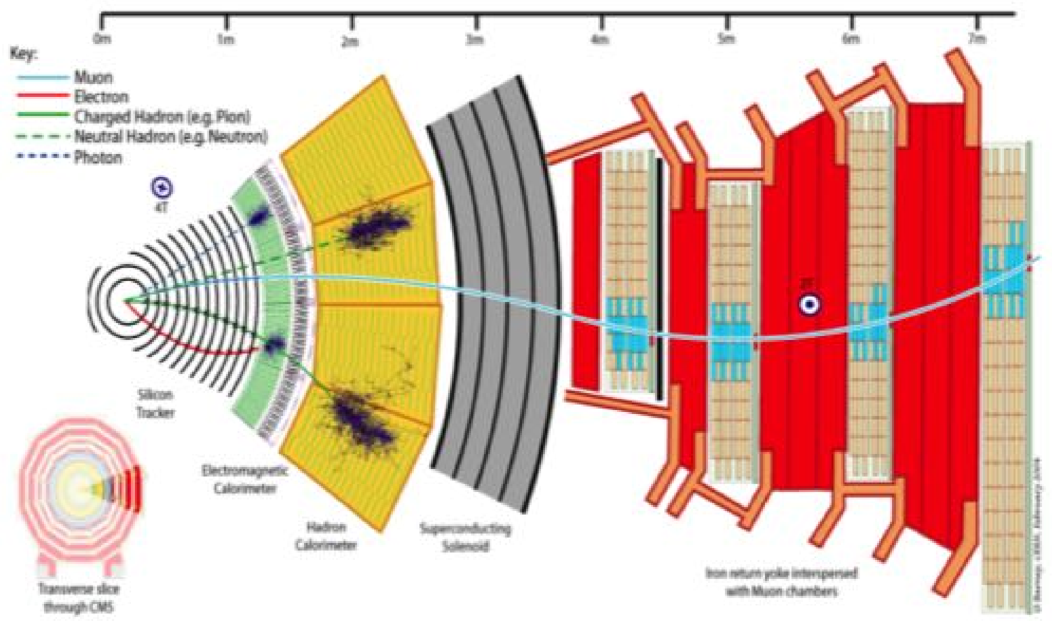




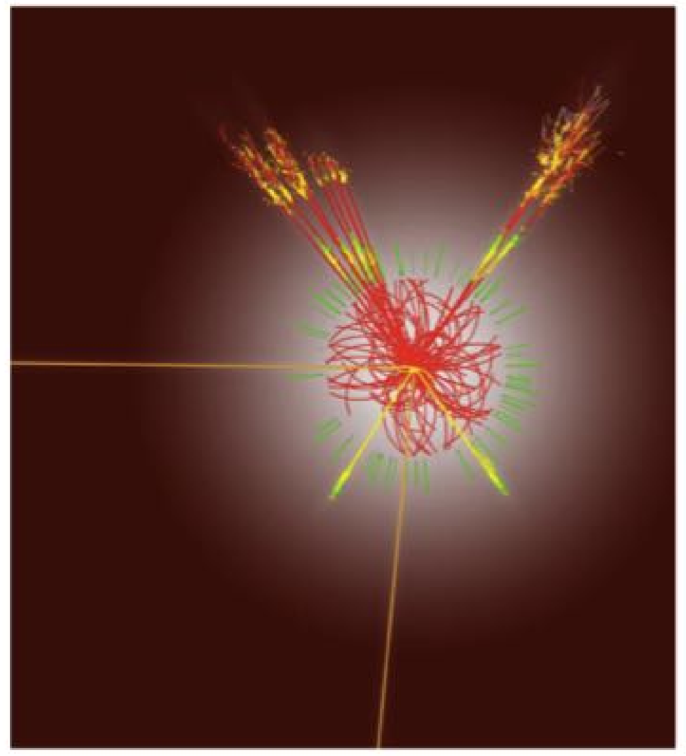
high energy physics applications



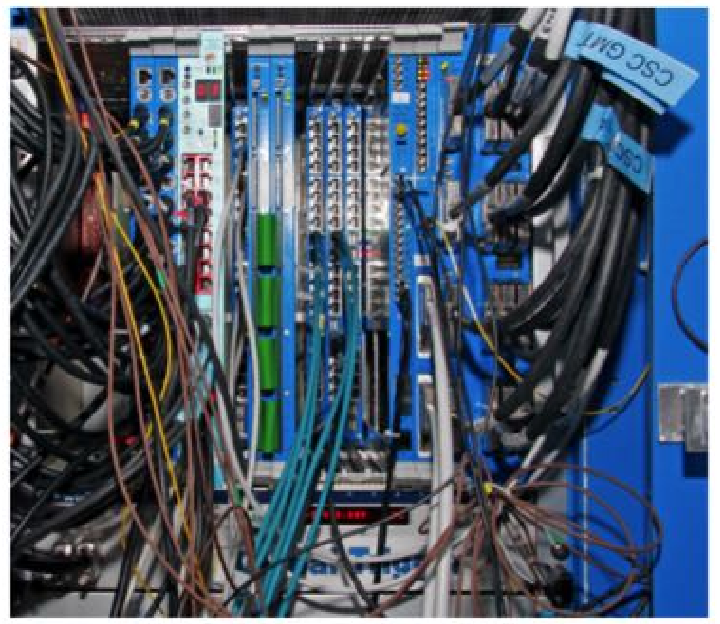
Deep Kalman RNNs



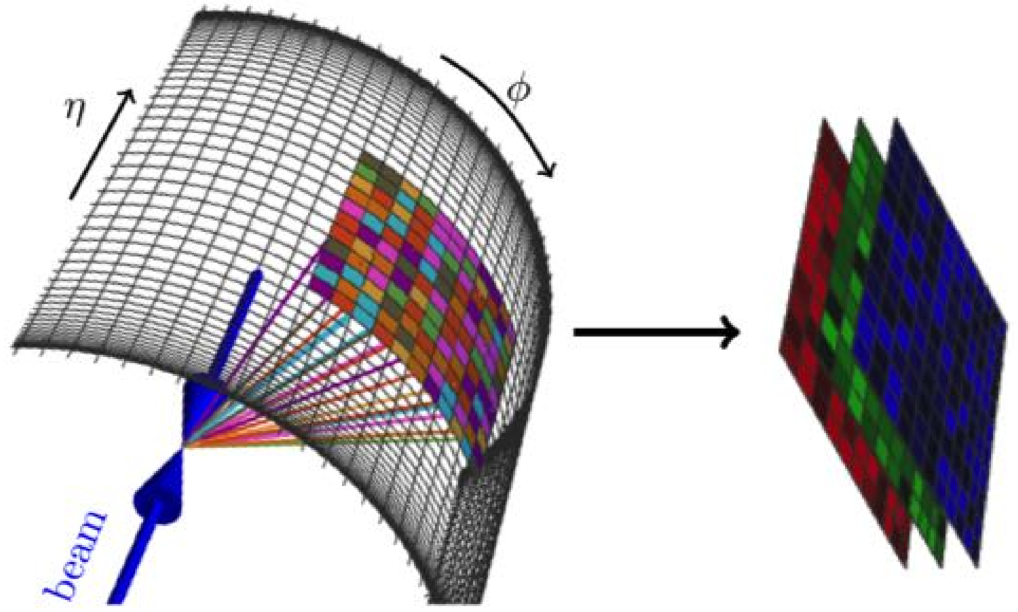
Generative Models, Adversarial Networks



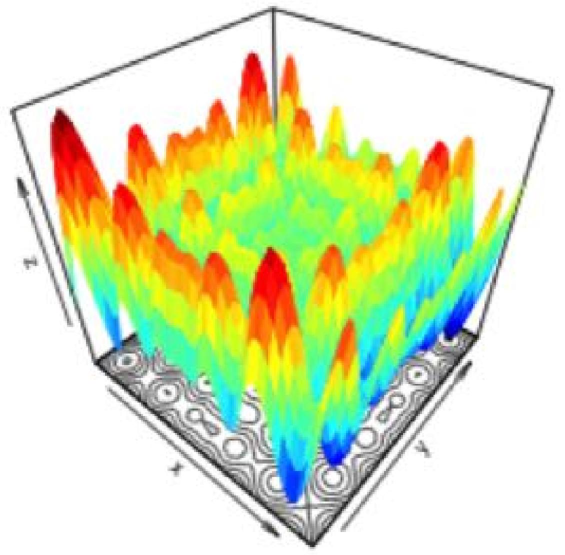
FCN, Recurrent, LSTM NN



Deep ML +FPGA



Convolutional DNN



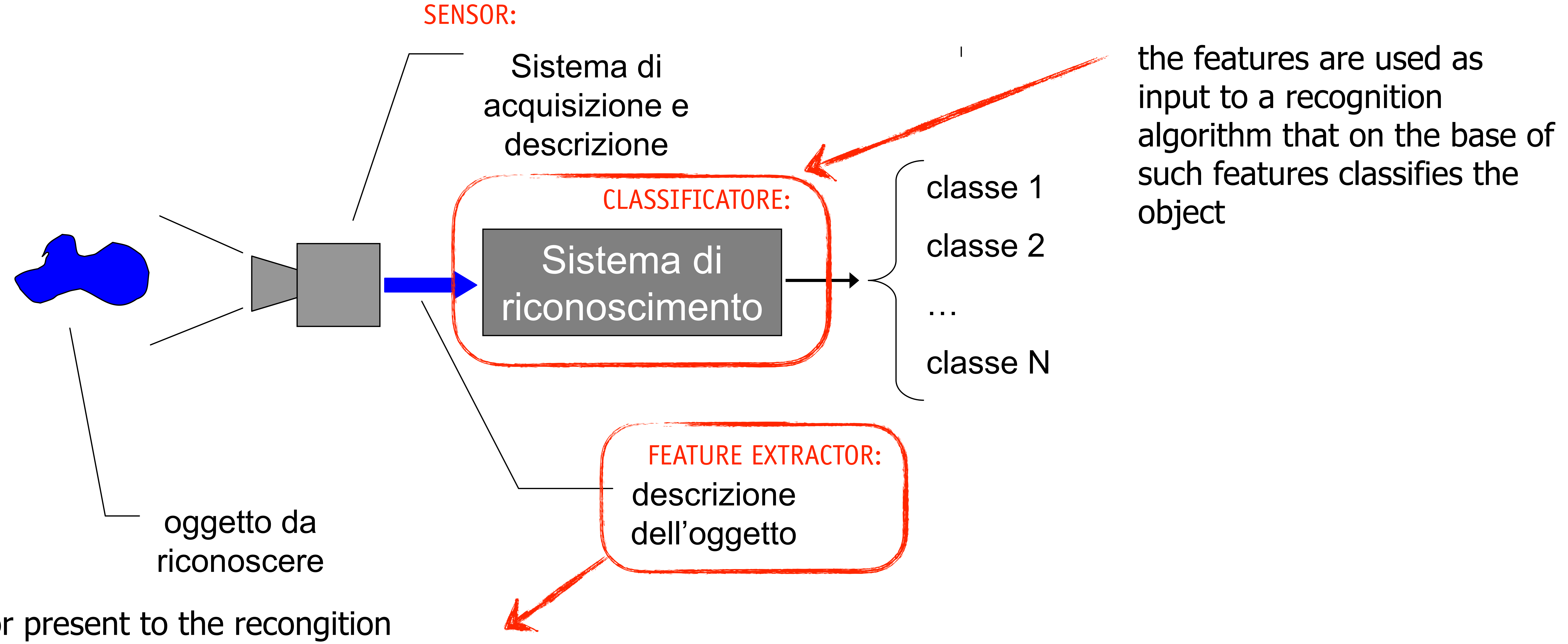
Multiobjective Regression

credits: D.Glayzer

And many more ...

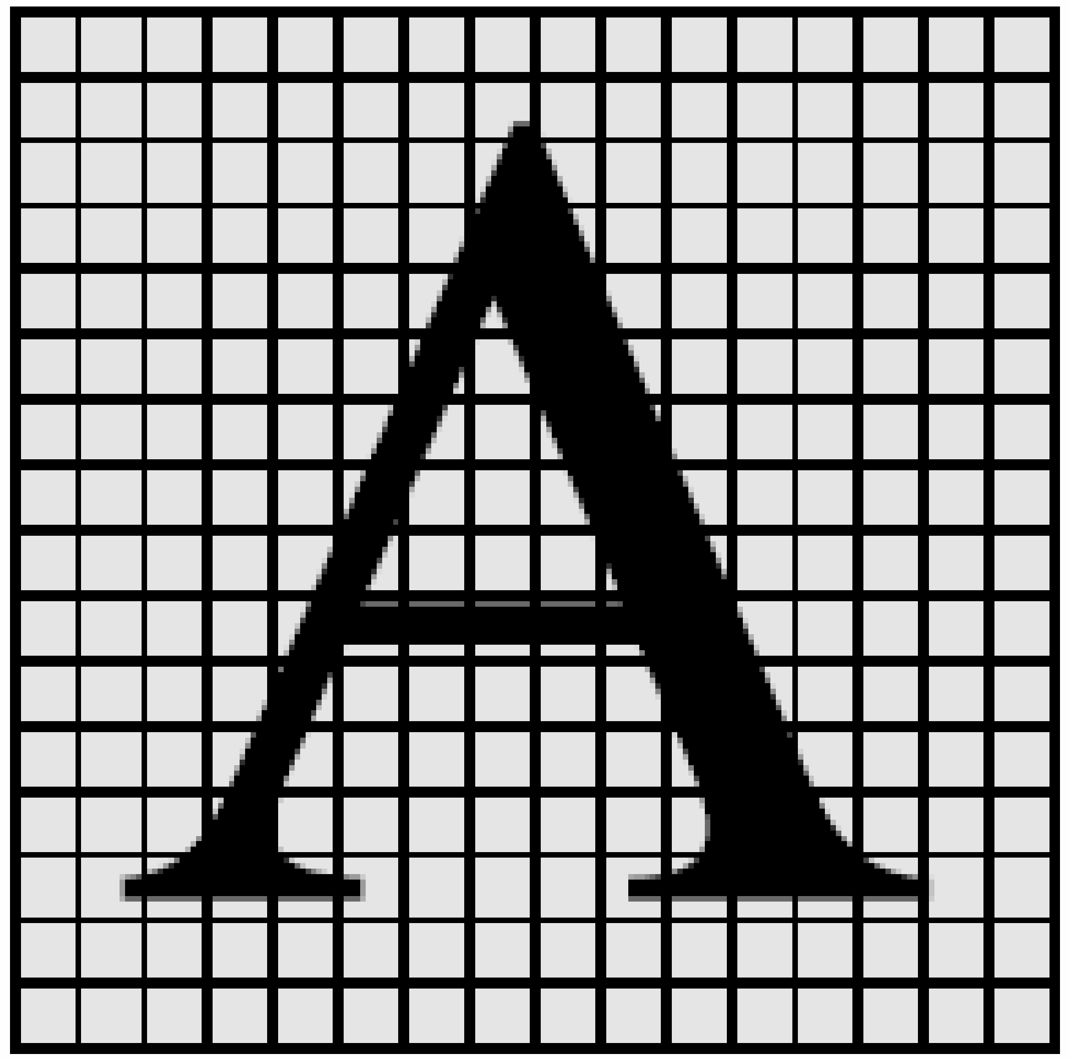
CONCEPTUAL SCHEME OF THE SIMPLEST CLASSIFICATION SYSTEM

- Given the description of an object that can belong to N possible classes, task for the system is to assign the object to one of the classes (o to assigna probability to each class) by using the knowledge base build during the training phase



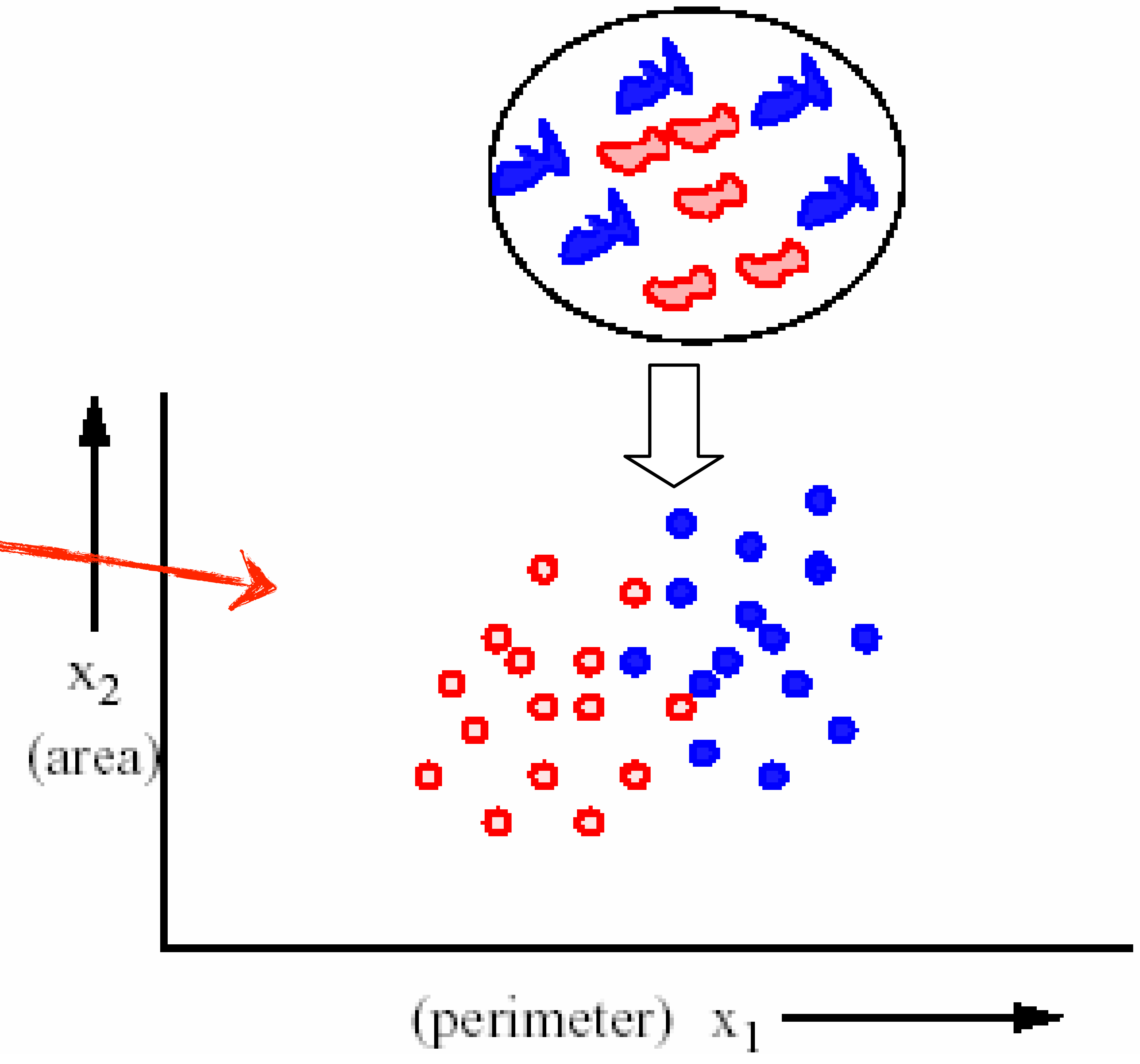
The feature estractor present to the recongition system a description, i.e. a set of measures (features) that characterise the object to be recognised

RAPPRESENTATIONS



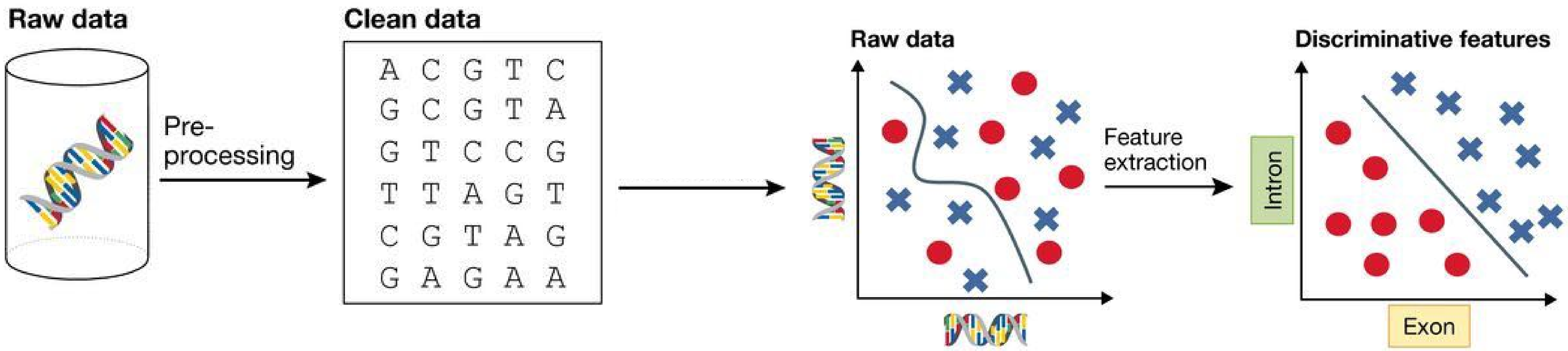
- 16x16=256 pixel represent the object
- From these pixels we can extract a set of measures (typically with a smaller dimension wrt the # of pixels) that describe the object (ex. Ratio between on/off pixels, area or perimeter of the clusters, ...): features

each object is then represented by a feature vector, that corresponds to a point in the feature space

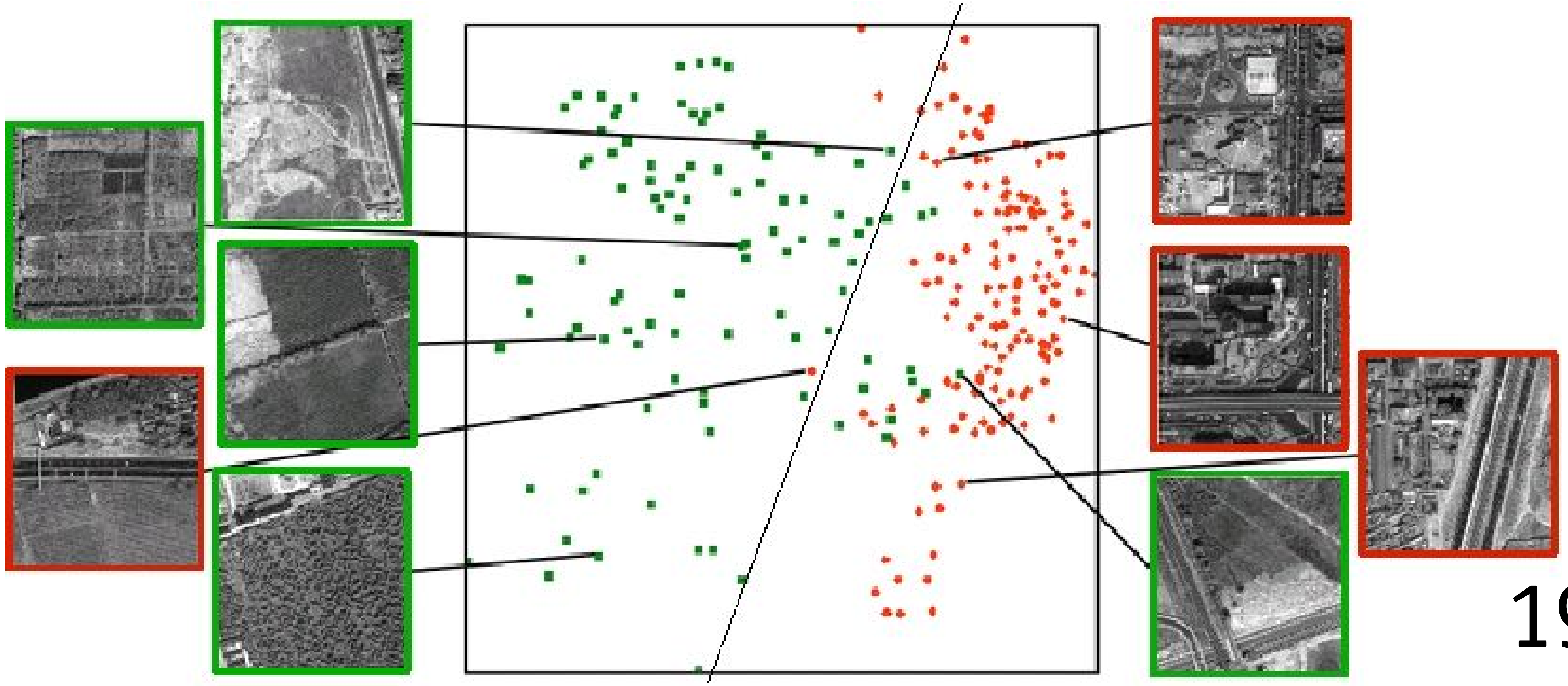


CHOICE OF FEATURES

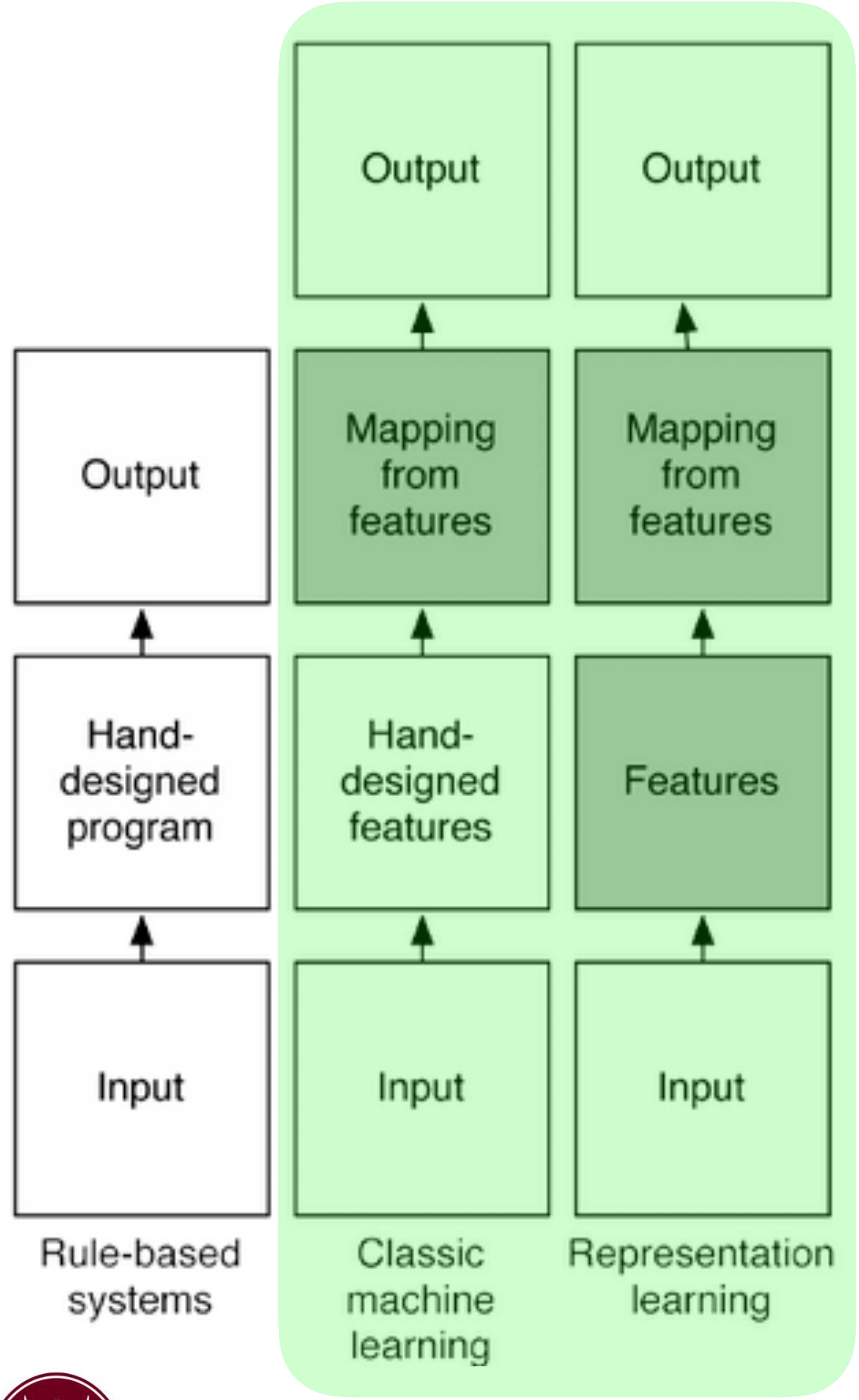
- the choice of the best representation of the data is one of the most crucial and important aspects of a ML algorithm



example: identification of rural and urban areas in satellite images based on spectral properties of the images



LERN THE DATA REPRESENTATION

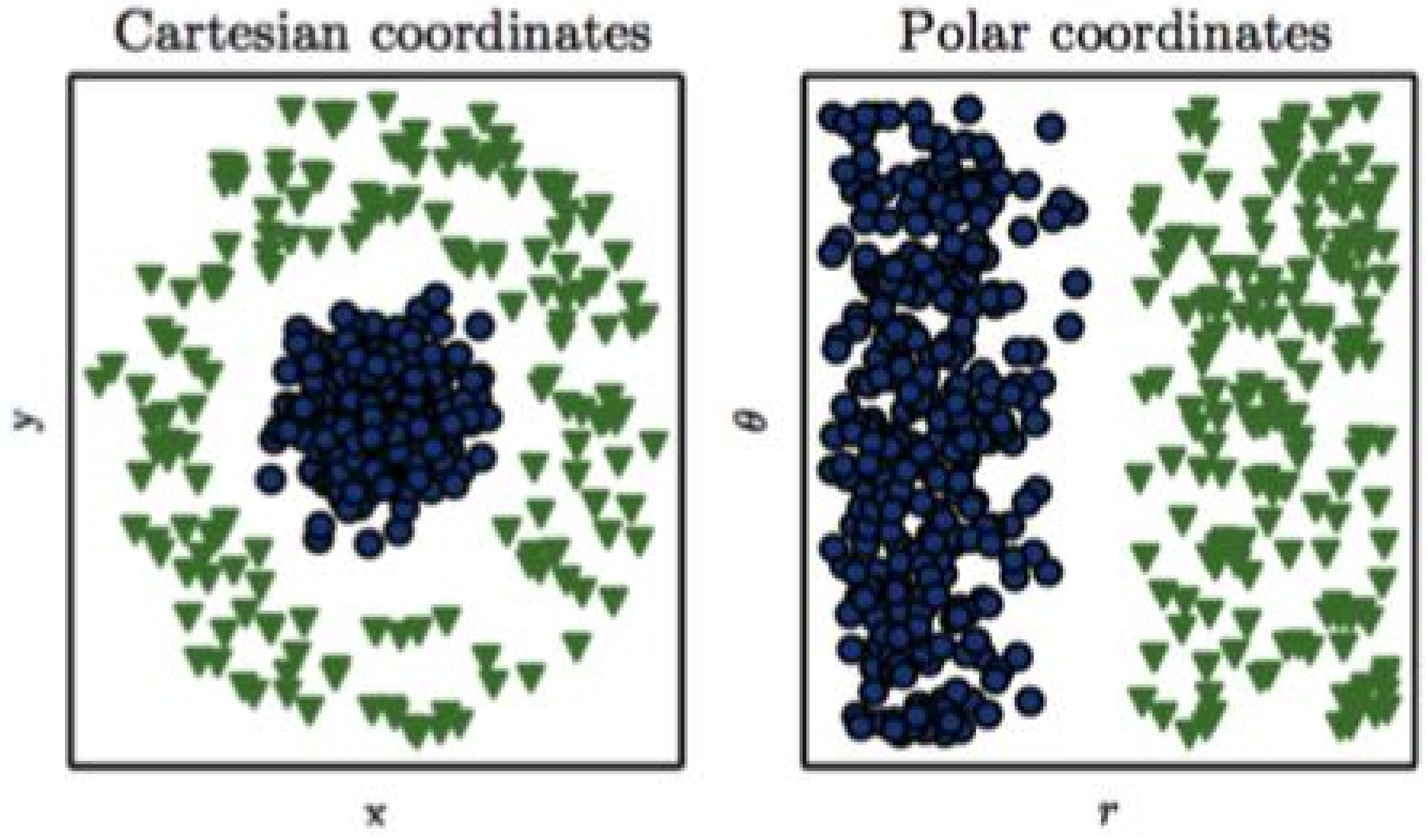


in first generation (classic ML): the feature set were build and chosen by the operator on the base of prior knowledge of the problem itself

- human: identify best features
- algorirthm: identify the best mapping between features and output

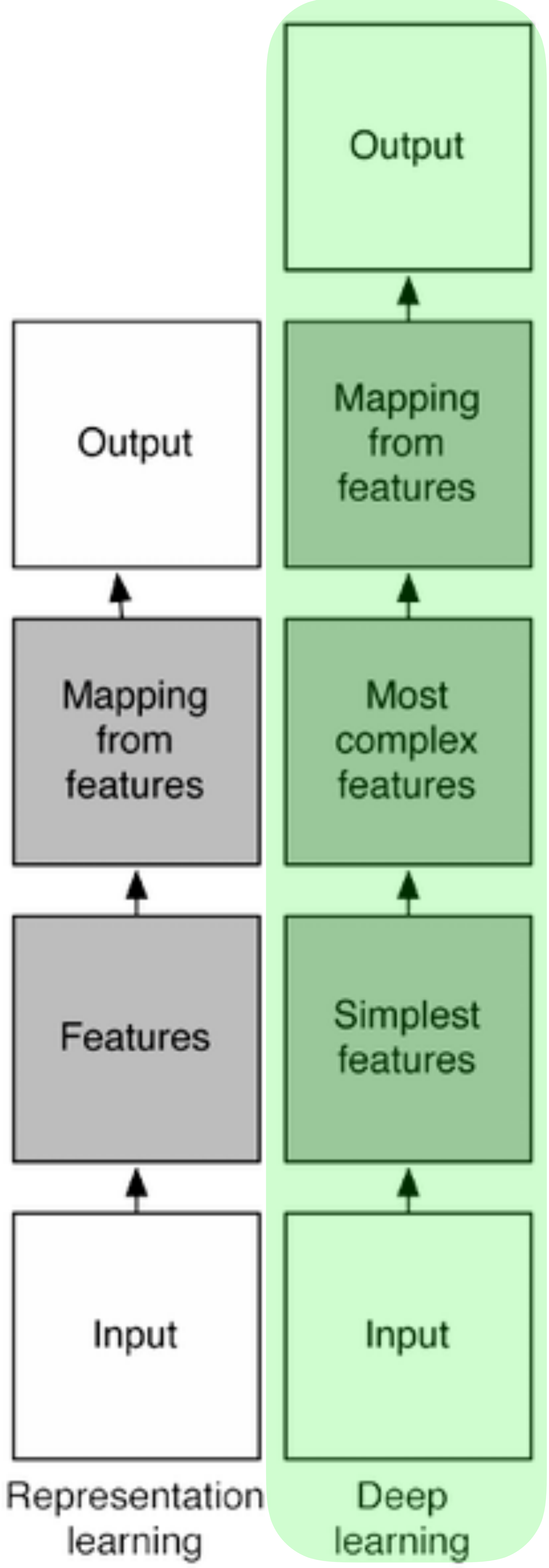
second generation ML: Representation Learning

- the algorithm scope is expanded by performing also the task to find in an automatic way a better representation of the data with respect to the one available with the input features



DEEP LEARNING (DL)

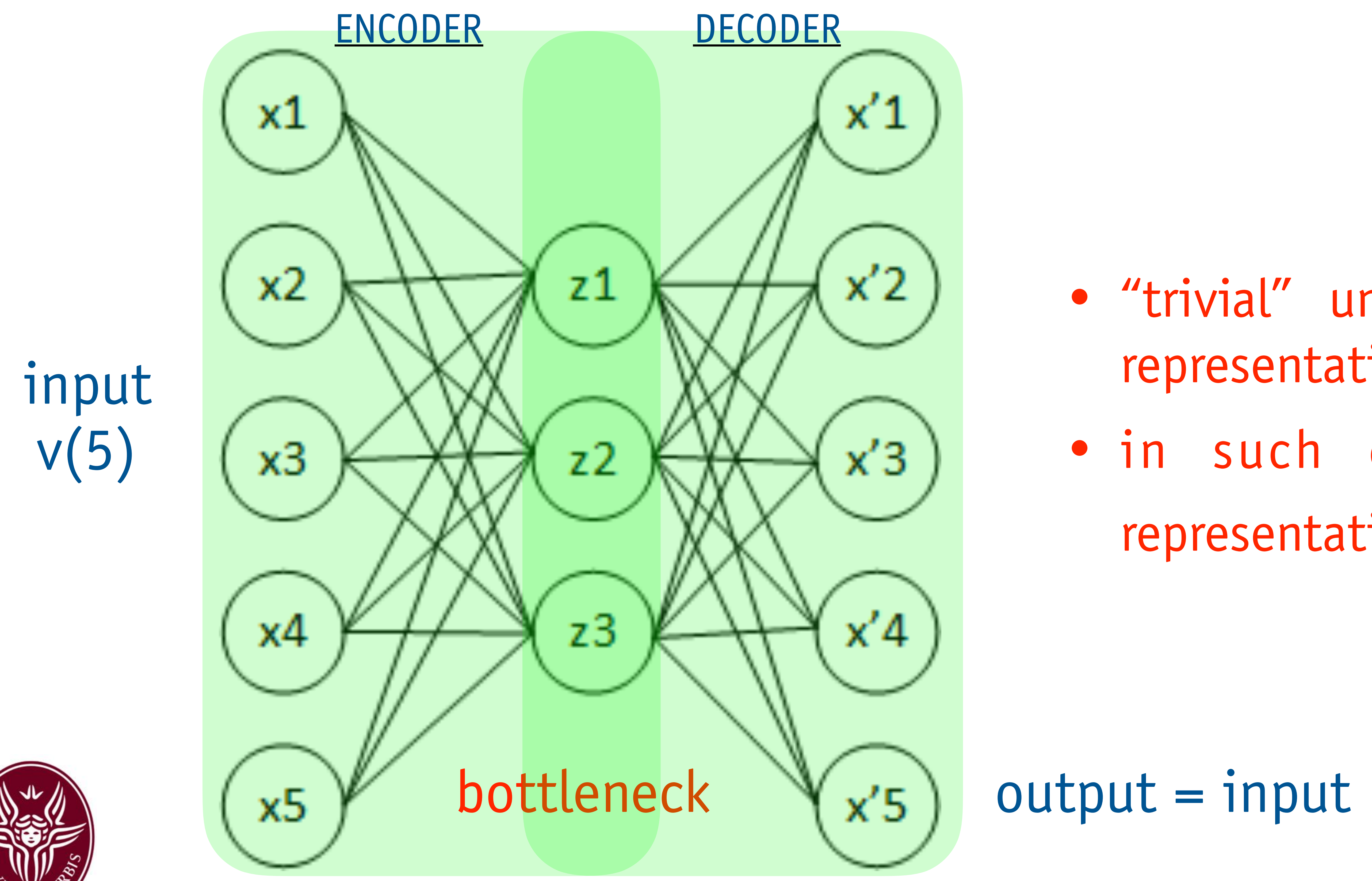
- the traditional ML algorithms were not very “cerative” in finding better representations
- basically they just searched the best possible transformation in a predefined set of operations called ”hypothesis space” of the algorithm. Search guided by the training examples
- The Deep Learning evolution solve this limitation by organising ideas and concepts in a hierarchical way and building new complex representations based on simpler ones
  - example: a person face can be presente by combining simpler features: eyes, mouth, hears …, that can be represented in trun by combining basic features: edges, contours, lines, …
- DL == HIERARCHICAL REPRESENTATION LEARNING



Extremely powerful, but requires huge training sets
and a lot of computing power …

AUTOENCODER: A BASIC EXAMPLE OF REPRESENTATION LEARNING

- non-supervised algorithm that try to identify common and fondamentali characteristic in the input data
- combines and encoder that converts input data in a different representation, with a decoder that converts the new representation back to the original input
- trained to output something as close as possible to the input (learn the identity function)



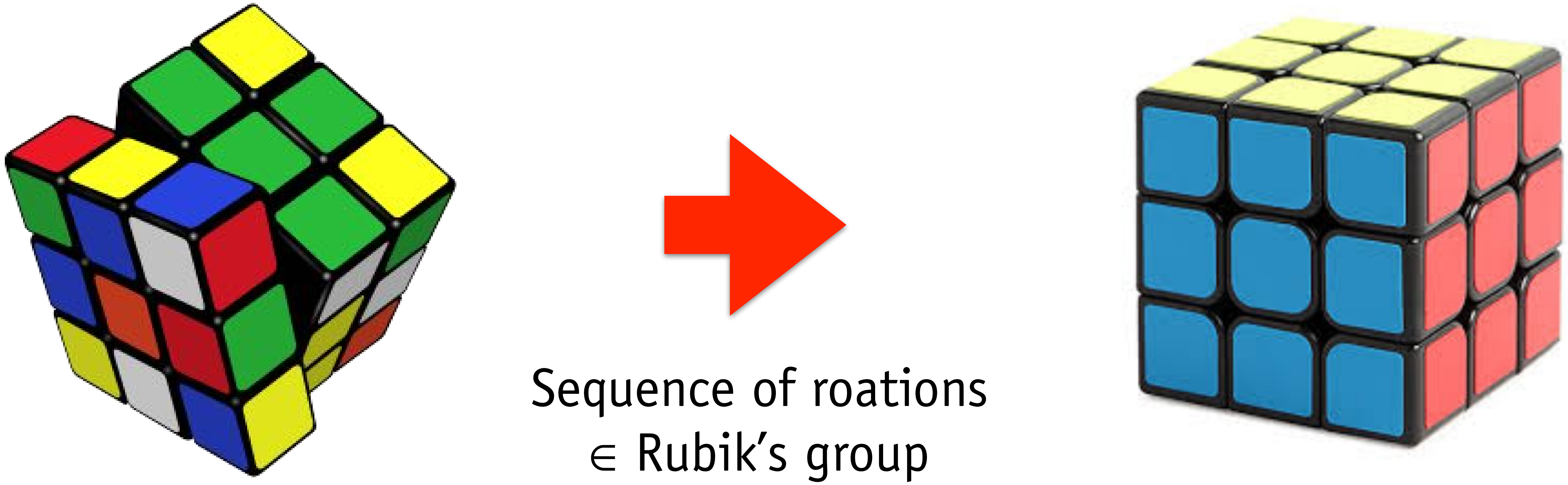
- "trivial" unless to constrain the network to have the hidden representation with a smallare dimension of the input/output
- in such case the network build (learn) "compressed" representations of the input features: $x \in R^5 \rightarrow z \in R^3 \rightarrow \cdots$

DL: INTUITIVE EXPLANATION ...

tt is like to solve the Rubik

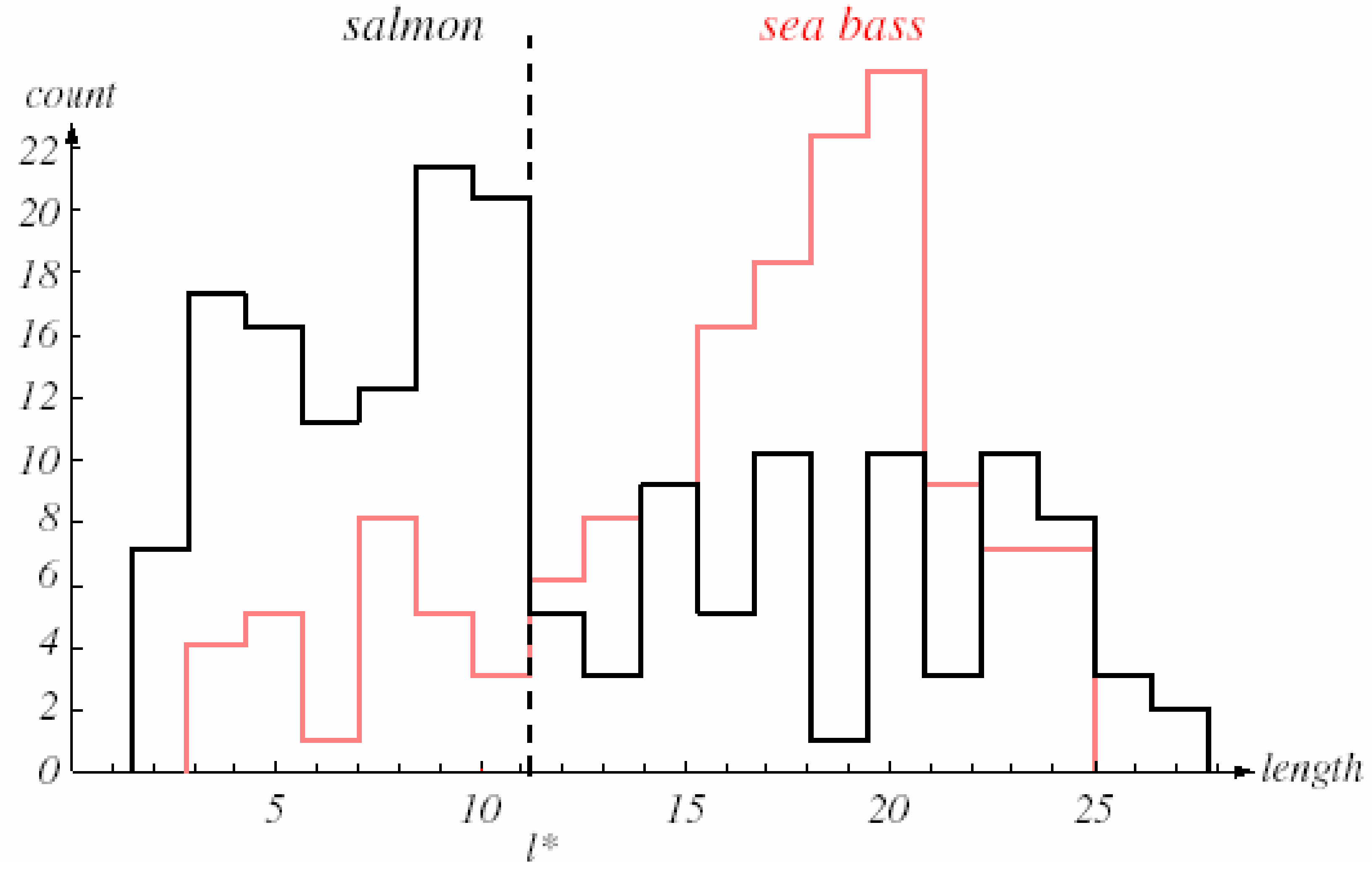
goal of the rappresentation-learning algorithm is to find the best transformation

in the DL this is implemented by a sequence of simpler transformations (in the specific example rotations in the 3D space)

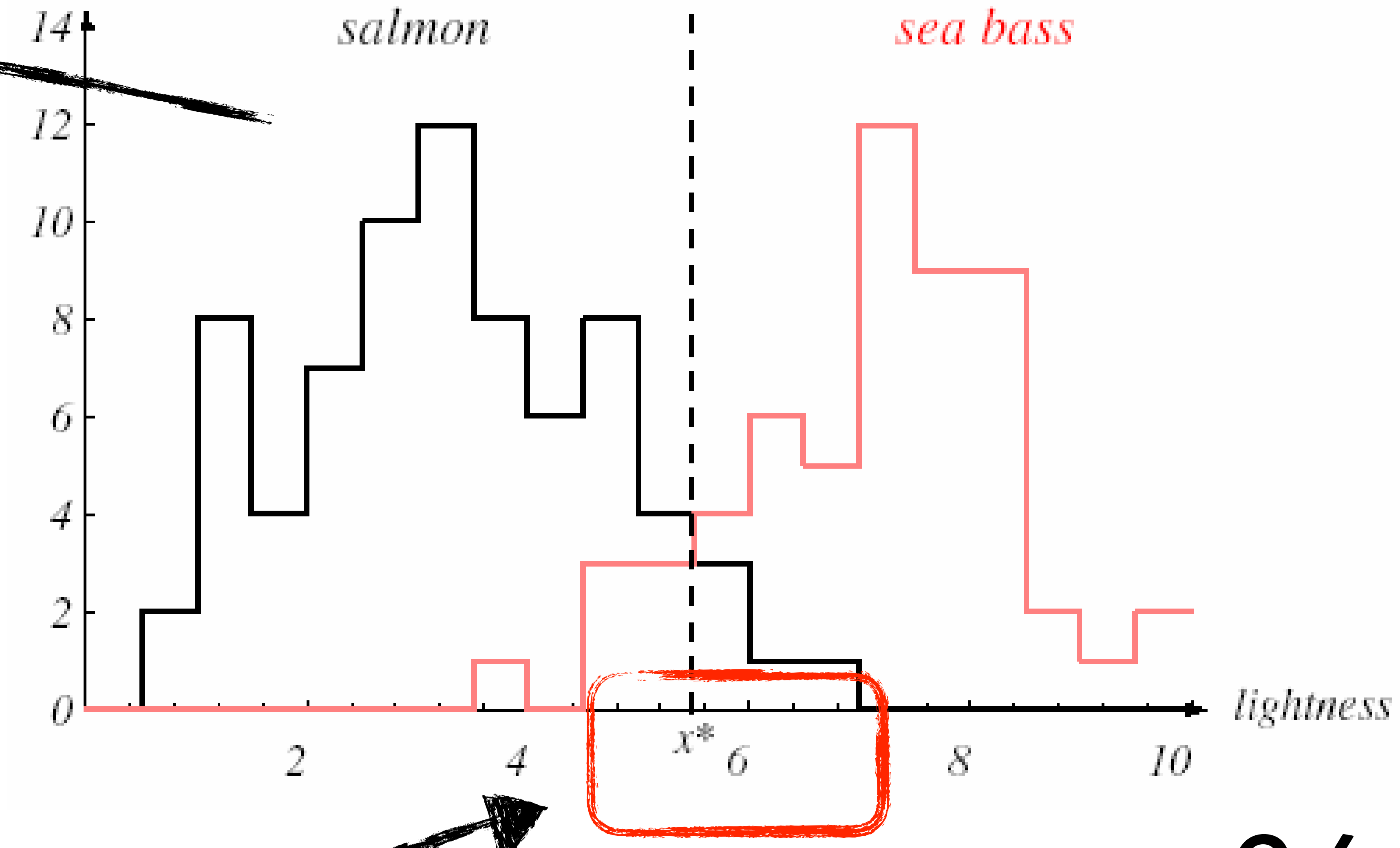


DECISION BOUNDARIES

- let’s assume that we have found that the two best features for our classification task are: length e lightness
- which one we should use for the classification? Which threshold?
- to decide this we make use of the traing set examples



Classification rule: if $x > x^*$: object $\in$ class A
else: objetc $\in$ class B



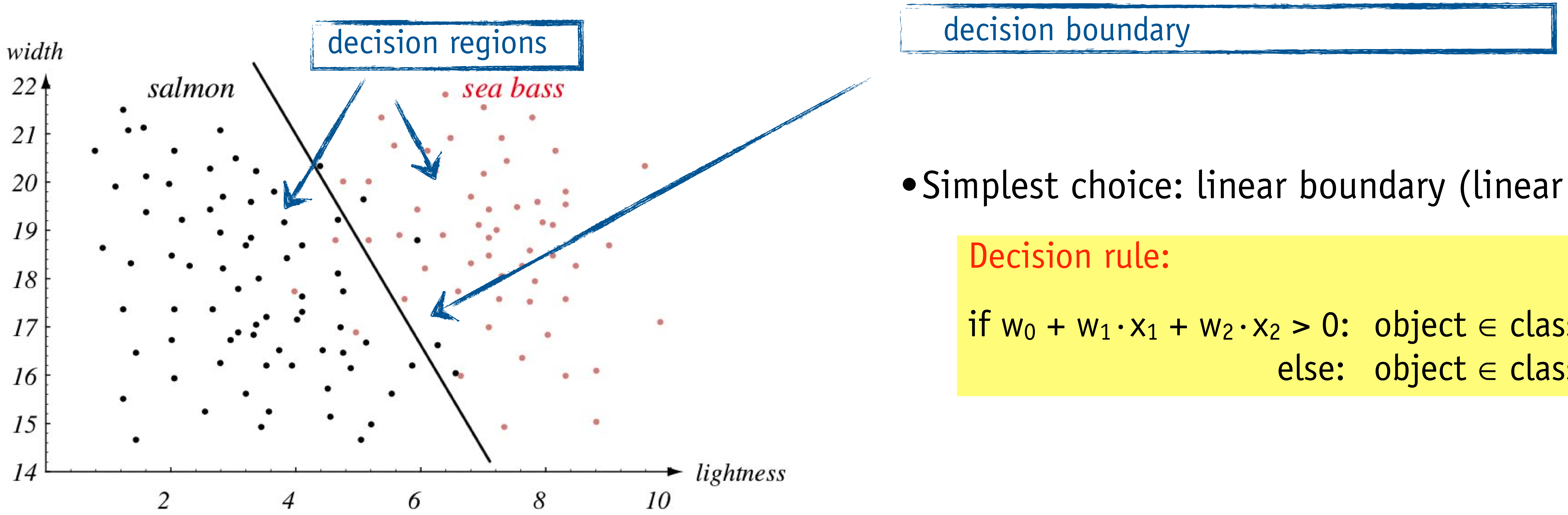
the threshold x^* is chosen in order to optimize an appropriate performance measure

example: accuracy, probability of misclassificantion, statistical risk …

decision boundary

DECISION BOUNDARIES

- to improve P a better strategy woudl be to use more than one feature at the same time
- The classification problem becomes the problem to find the best partition of the feature space, so that the classification error is the smallest one



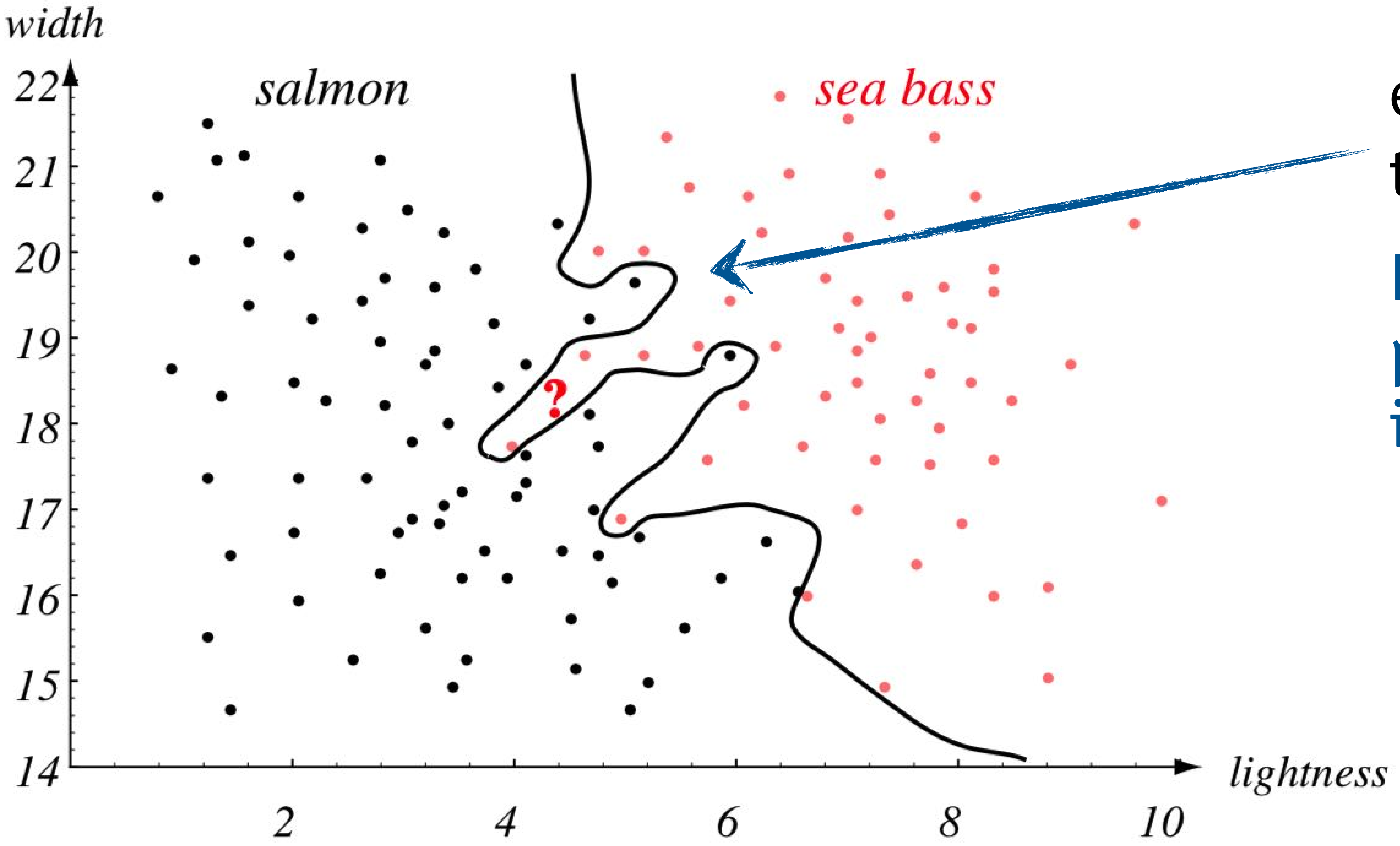
- Simplest choice: linear boundary (linear classifier)

Decision rule:

if $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 > 0$: object $\in$ class A
else: object $\in$ class B

COMPLEX DECISION BOUNDARIES ...

- question: it is possible to get rid of all errors with a complex decision boundary?



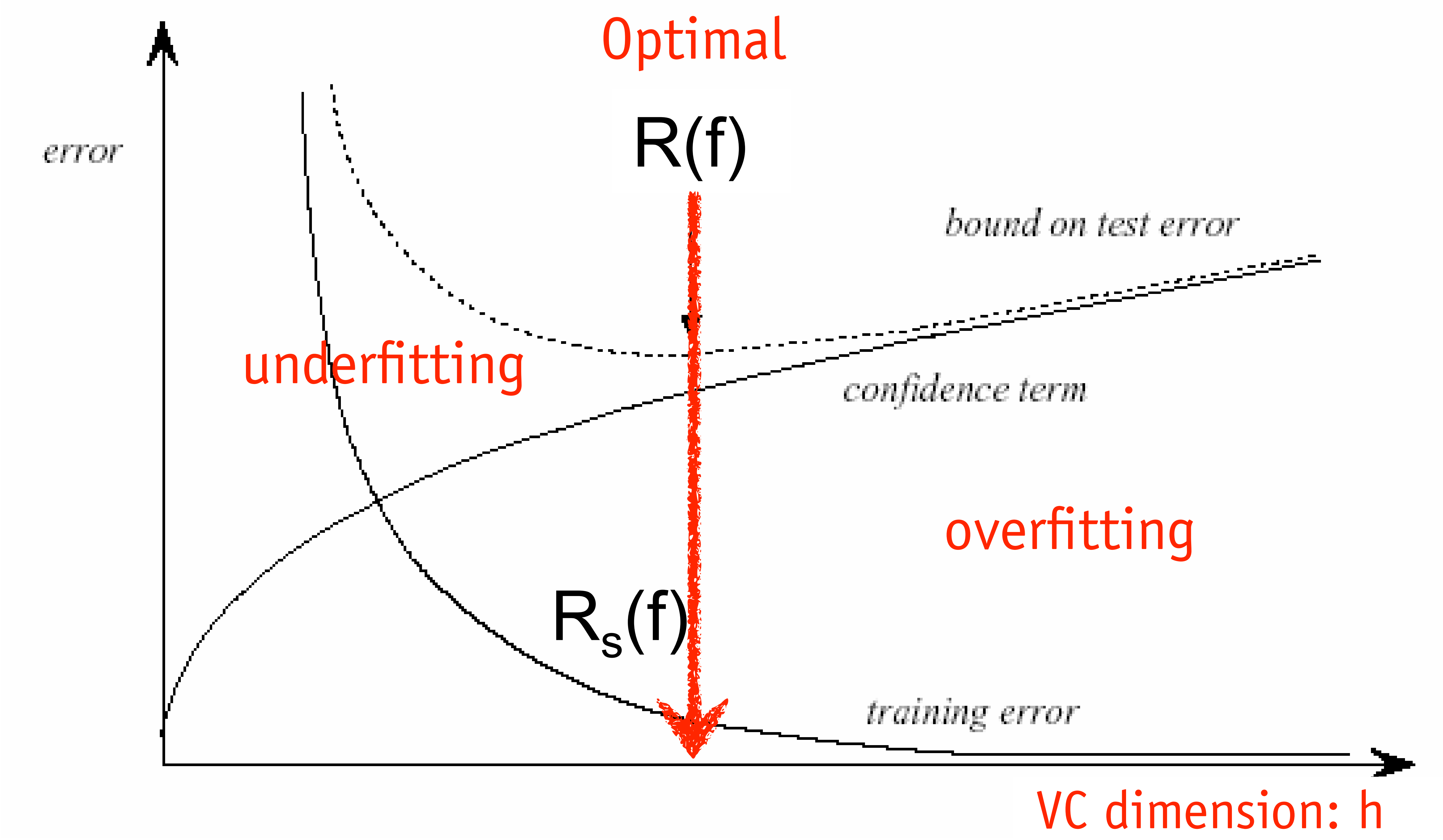
example: this boundary correctly clasify all the events of the trining set

PROBLEM: this way we are NOT guarantee a good performance of the algorithm when applied to events from independent samples wrt the training set (overfitting)

- The training set has finite dimension and the decision boundary is sensitive to the statistical fluctuation in the training set
- This aspect is called generalisation problem, and sis the crucial aspect in the design and training of any ML algorithm!

VAPNIK THEORY

bias-variance tradeoff: if we use a more complex model we pay a price of a larger variance ...



possible strategies

1. Mantenere fisso il termine di confidenza (scegliendo una struttura appropriata del classificatore) e minimizzare il rischio empirico.
2. Mantenere fisso il rischio empirico (p.es. uguale a zero) e minimizzare il termine di confidenza.

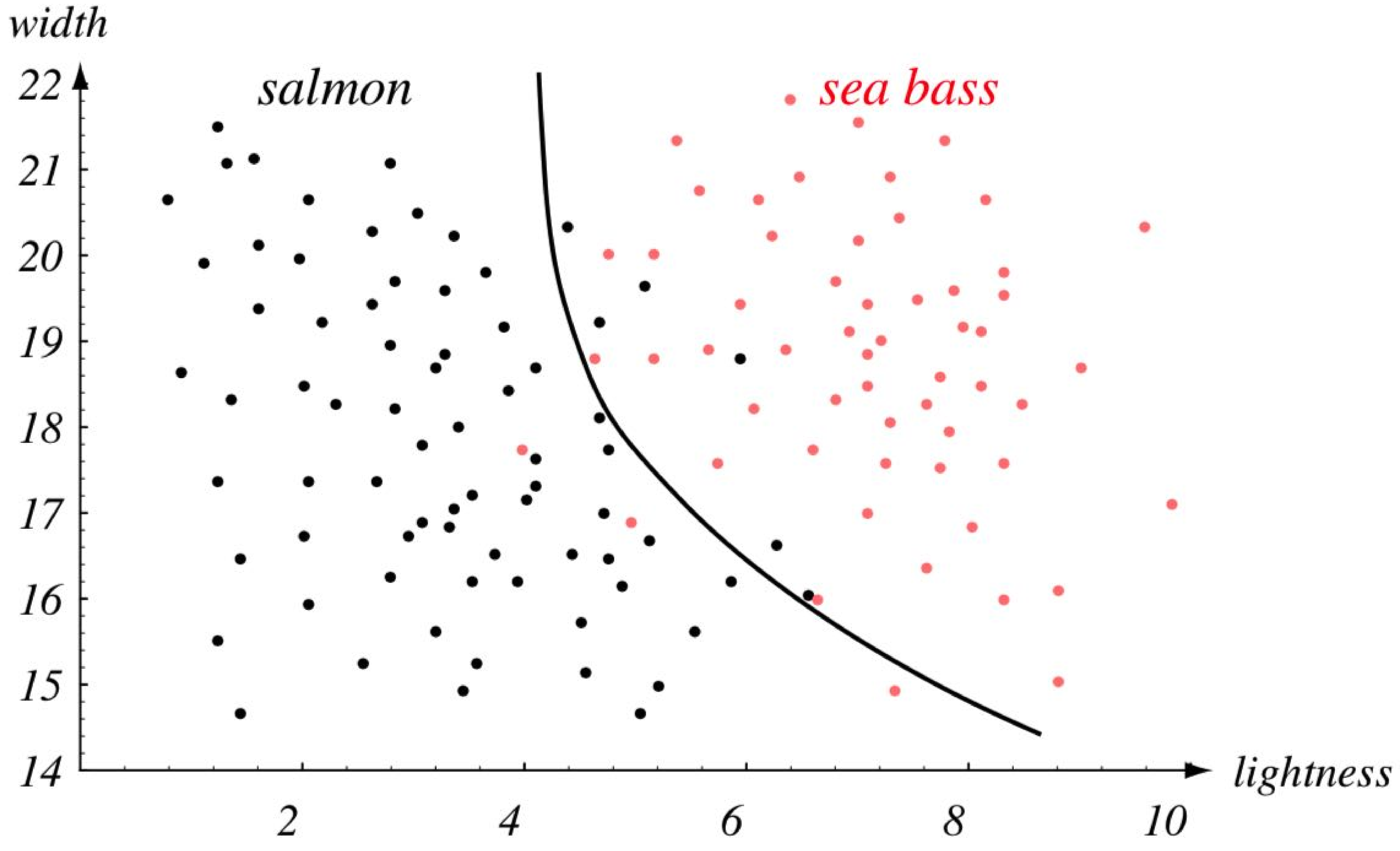
Shallow Neural Networks

Support Vector Machines

GENERALISATION PROBLEM

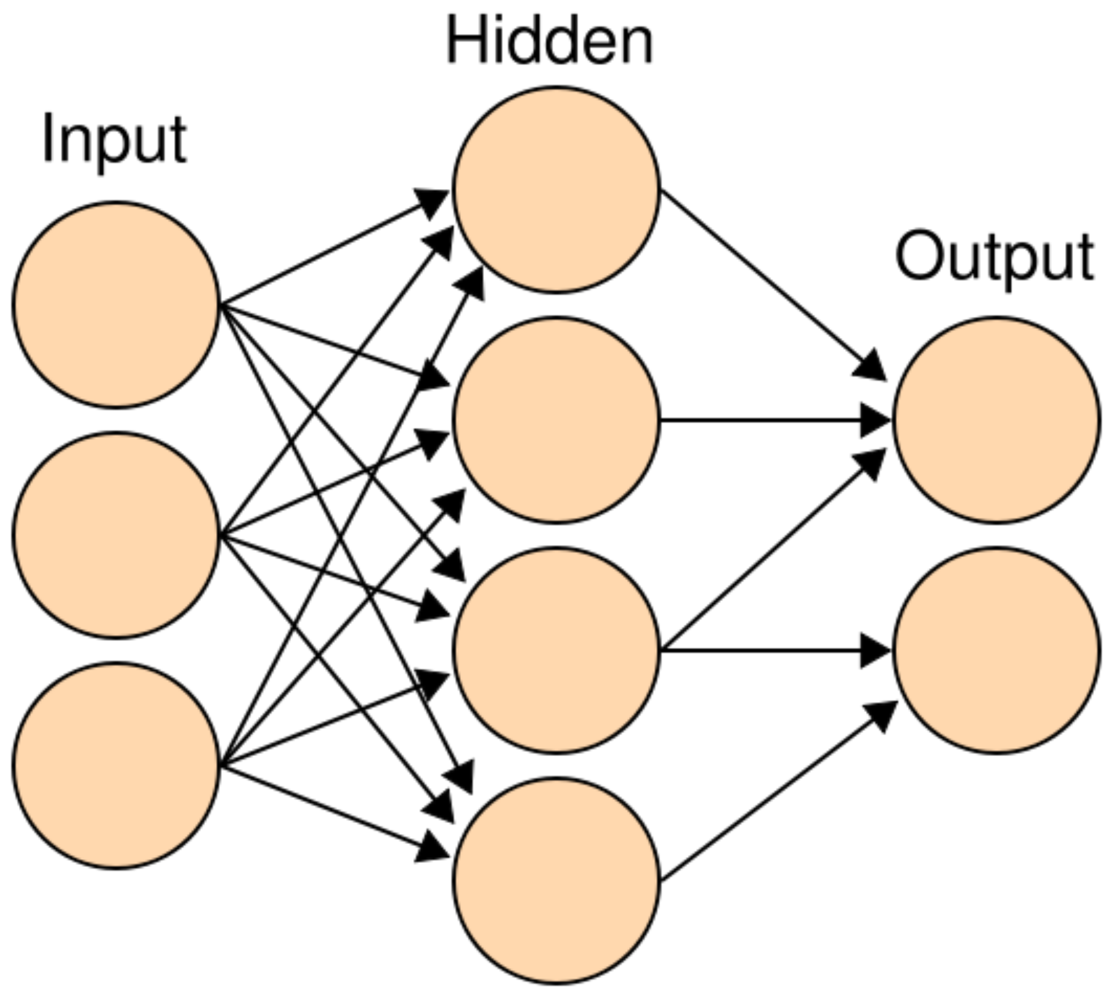
- at the end the choice of the decision boundary ia a trade off between:
  - Performance of the classifier on the training-set
  - Generalisation capacity of the classifier on the validation-set
- it is always preferable to accept a certain margin of error on the trining set if this allows to a better generalisation of the algorithm

example: a decision boundary with good performance and good generalisation capacity



ARTIFICIAL NEURAL NETWORKS

- the most popular approach to machine and deep learning to date
- an ANN is a mathematical model based on similarity with biologica neural networks:
  - based on an interconnected group of identical units (neurons)
  - process input information accordino to a connexionist computational approach: → collective actions performed in parallel by simple processing units (neurons)
  - behave as anadaptive system: structure dynamically modified during the learning phase based on a set of examples that flow through the network during the training step

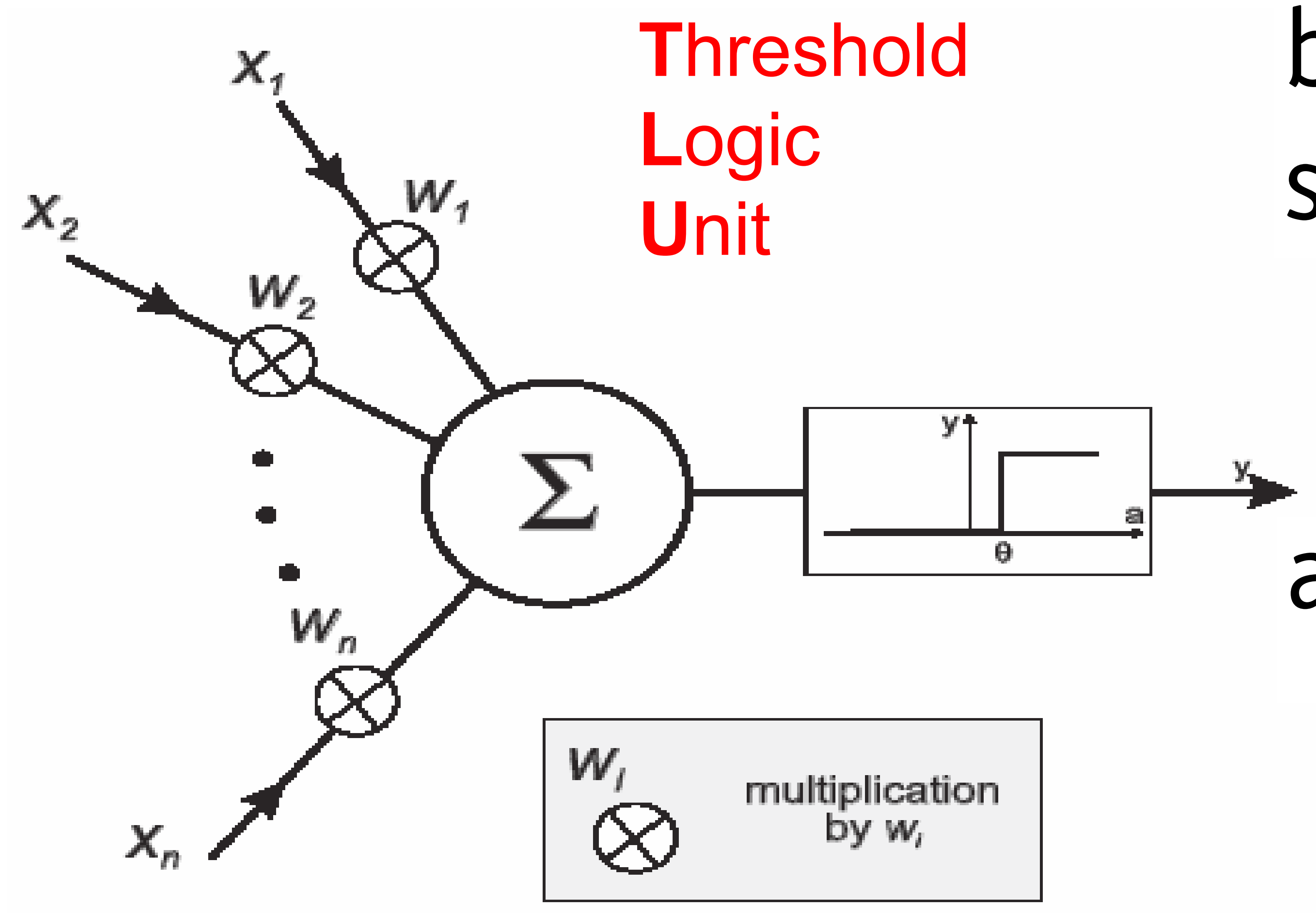


- non linear response obtained by non linear activation functions used as output of each neuron
- hierarchic representation learning obtained by implementing complex architectures with multiple layers of connected neurons (deep-NN)

ARTIFICIAL NEURON MODEL

Modello di McCulloch-Pitts (1943) / Rosenblatt (1962)

TLU:
Threshold
Logic
Unit



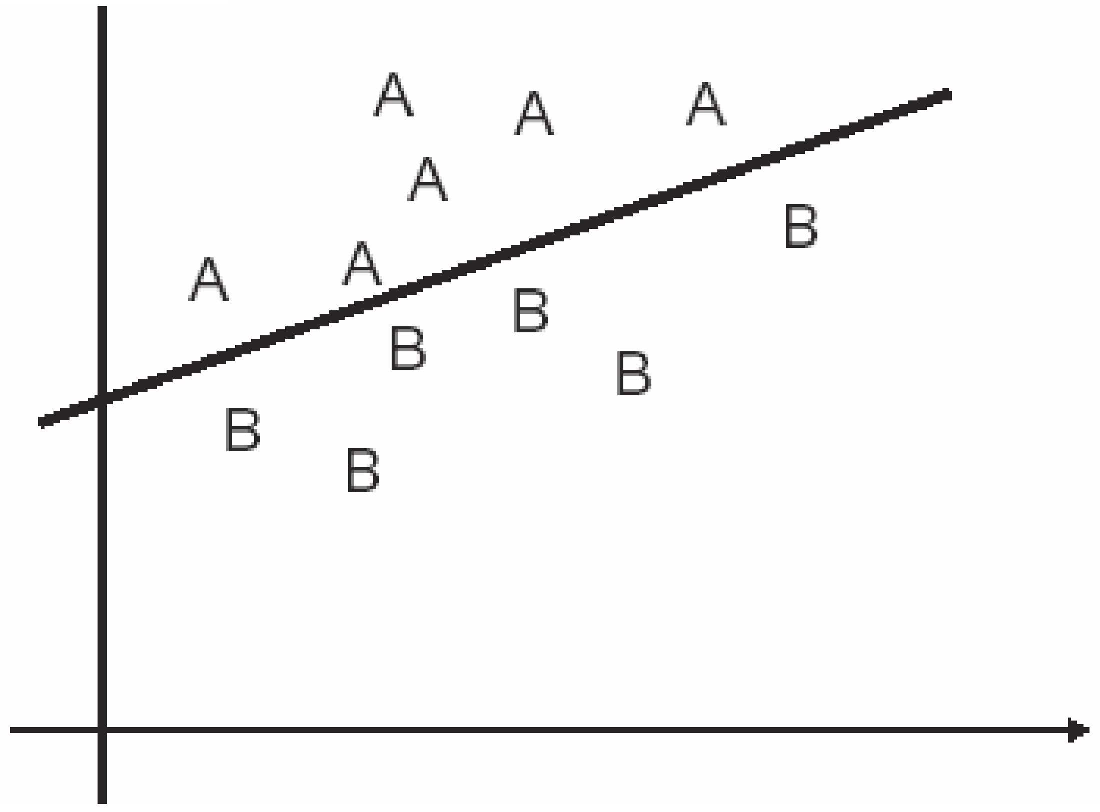
binary signals (x_i = 0,1)
synaptic function:

$$a = \sum_{i=1}^{n} w_i x_i$$

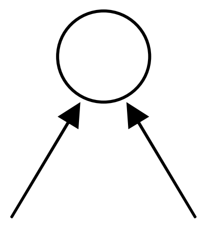
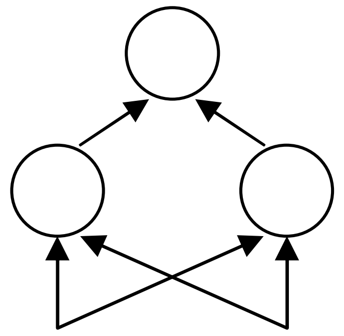
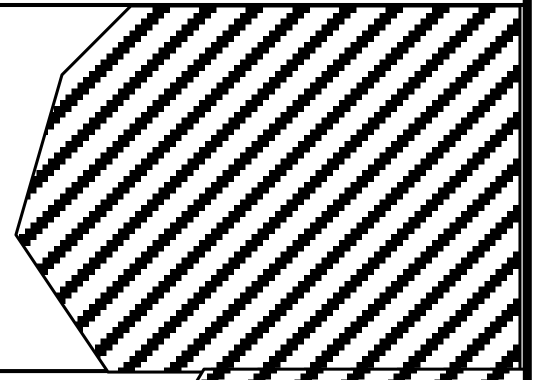
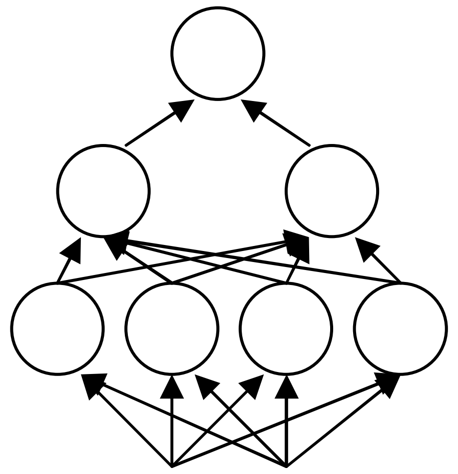
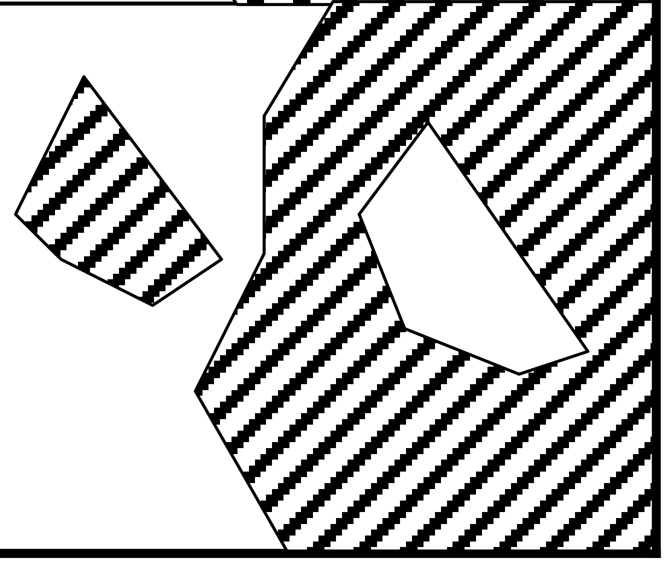
activation function (i.e. output):

$$y = \begin{cases} 1 & if \quad a \geq \vartheta \\ 0 & if \quad a < \vartheta \end{cases}$$

with a TLU it is possible to solve problems with linearly separable classes:



COMPLEX SEPARATION REGIONS

Struttura	Regioni di decisione	Forma generale
	Semispazi delimitati da iperpiani	
	Regioni convesse	
	Regioni di forma arbitraria	

Universal Approximation Theorem

a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of Rn, under mild assumptions on the activation functio

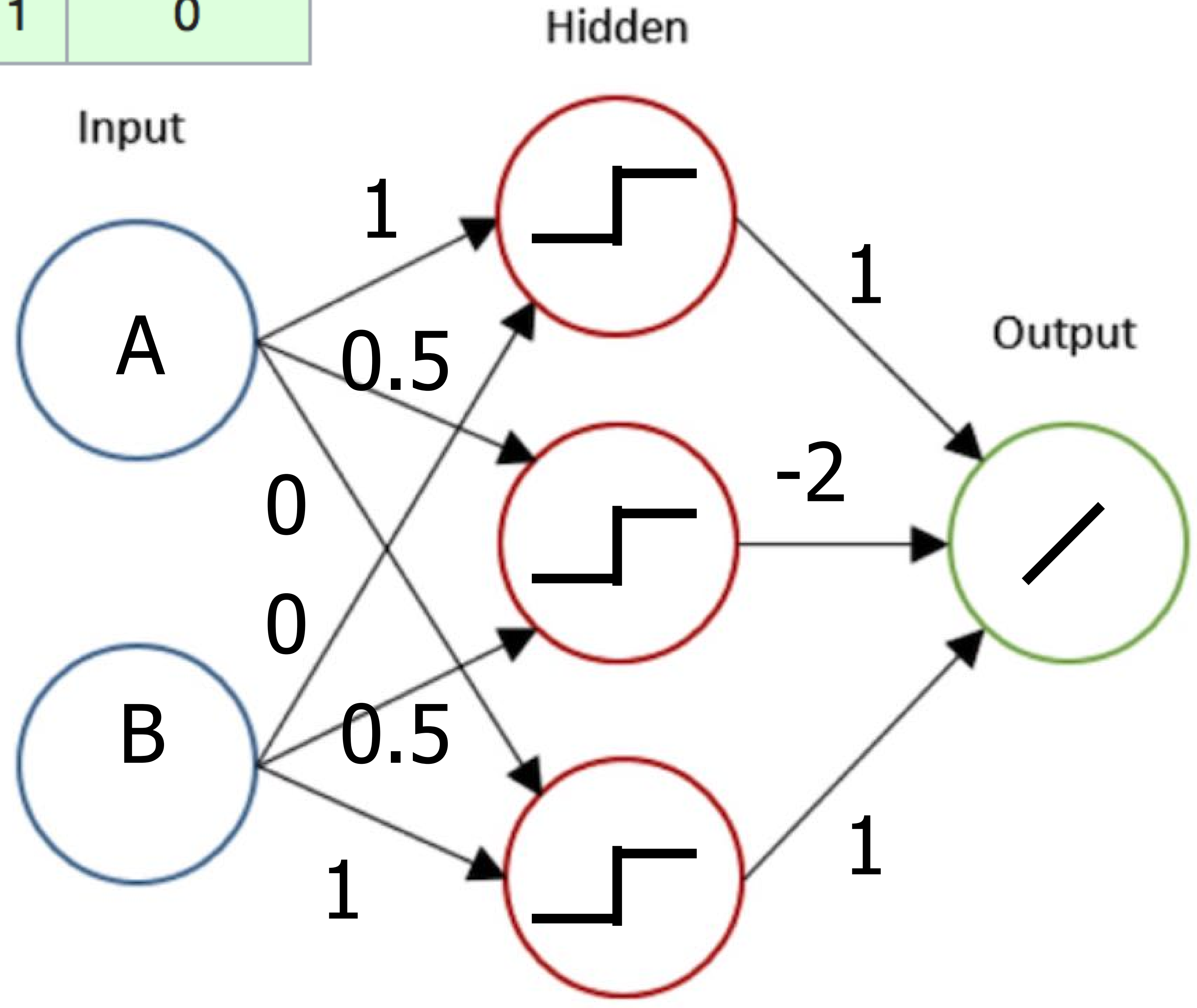
$$F(x) = \sum_{i=1}^{N} v_i \varphi \left(w_i^T x + b_i \right)$$

NOTE: the theorem does not say anything on the effective possibility to learn in an easy way the parameters of the network!

SIMPLEST NN EXAMPLE: XOR GATE

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

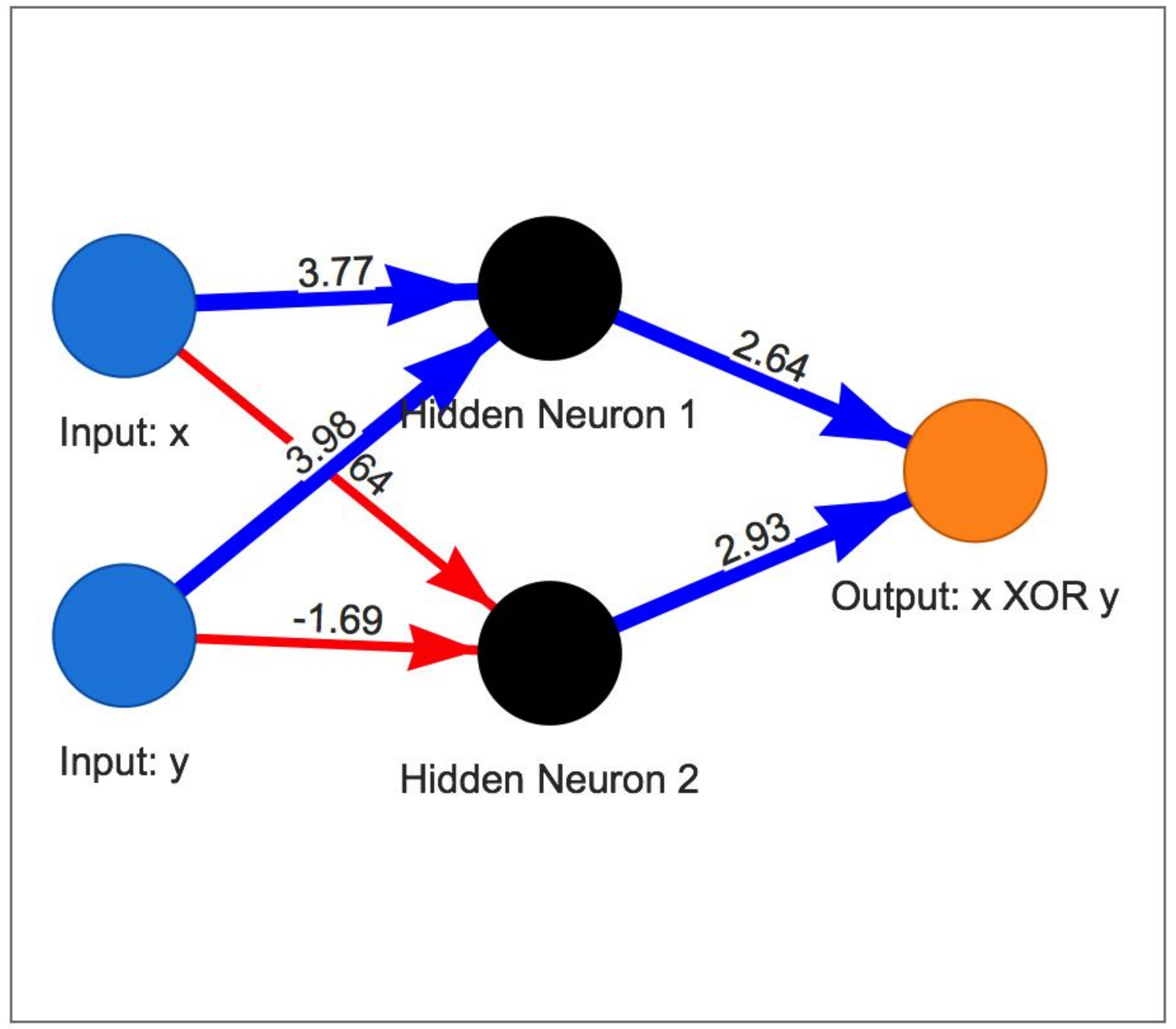
$(A\cdot\overline{B})+(\overline{A}\cdot B)$

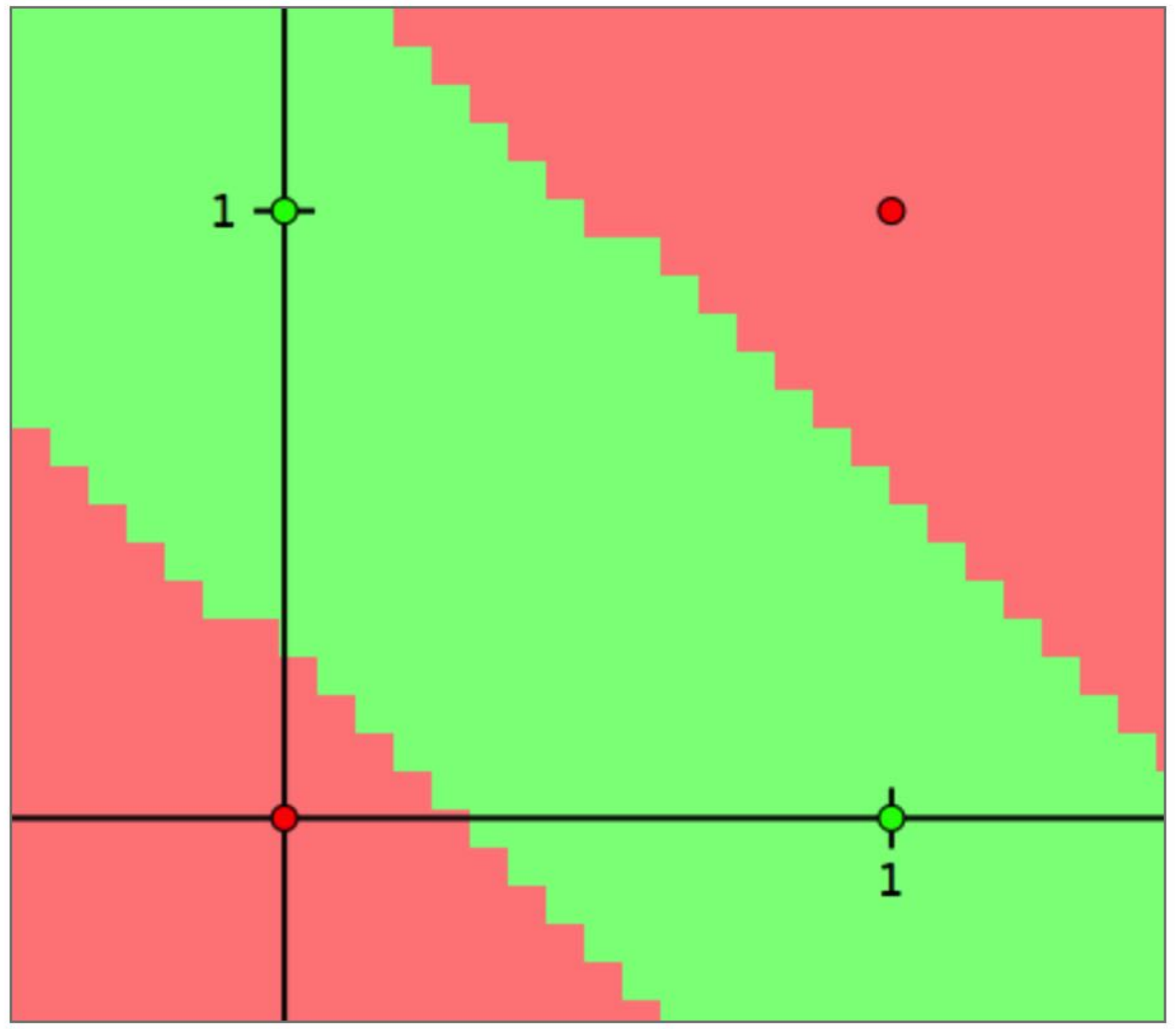


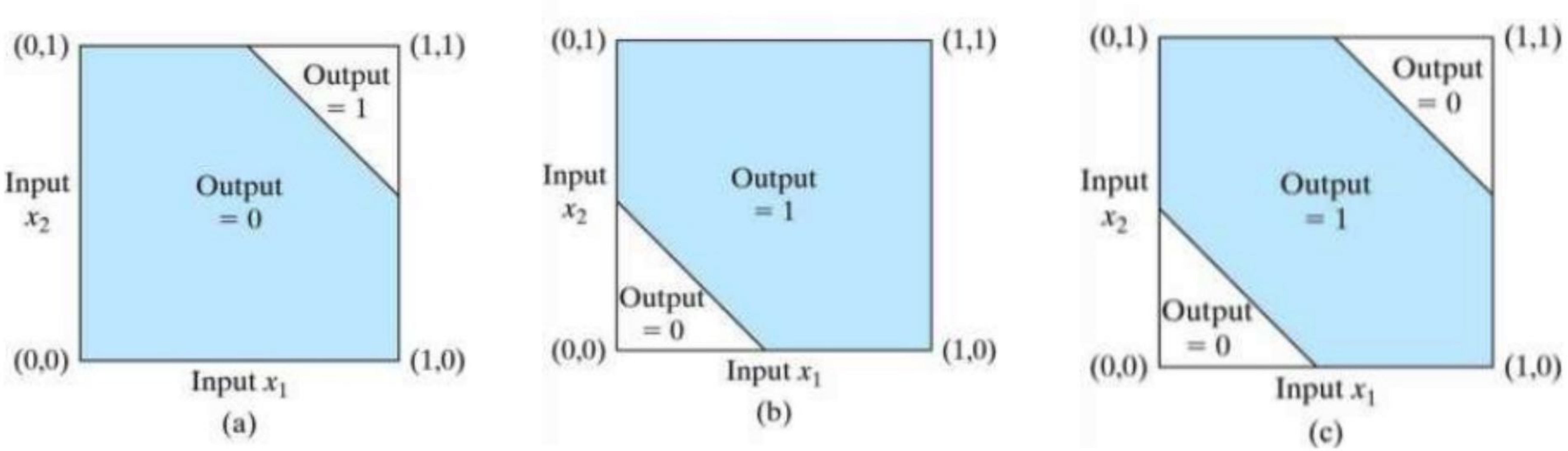
TLU

out = $\theta(\Sigma w_i x_i)$

MLP (sigmoid)







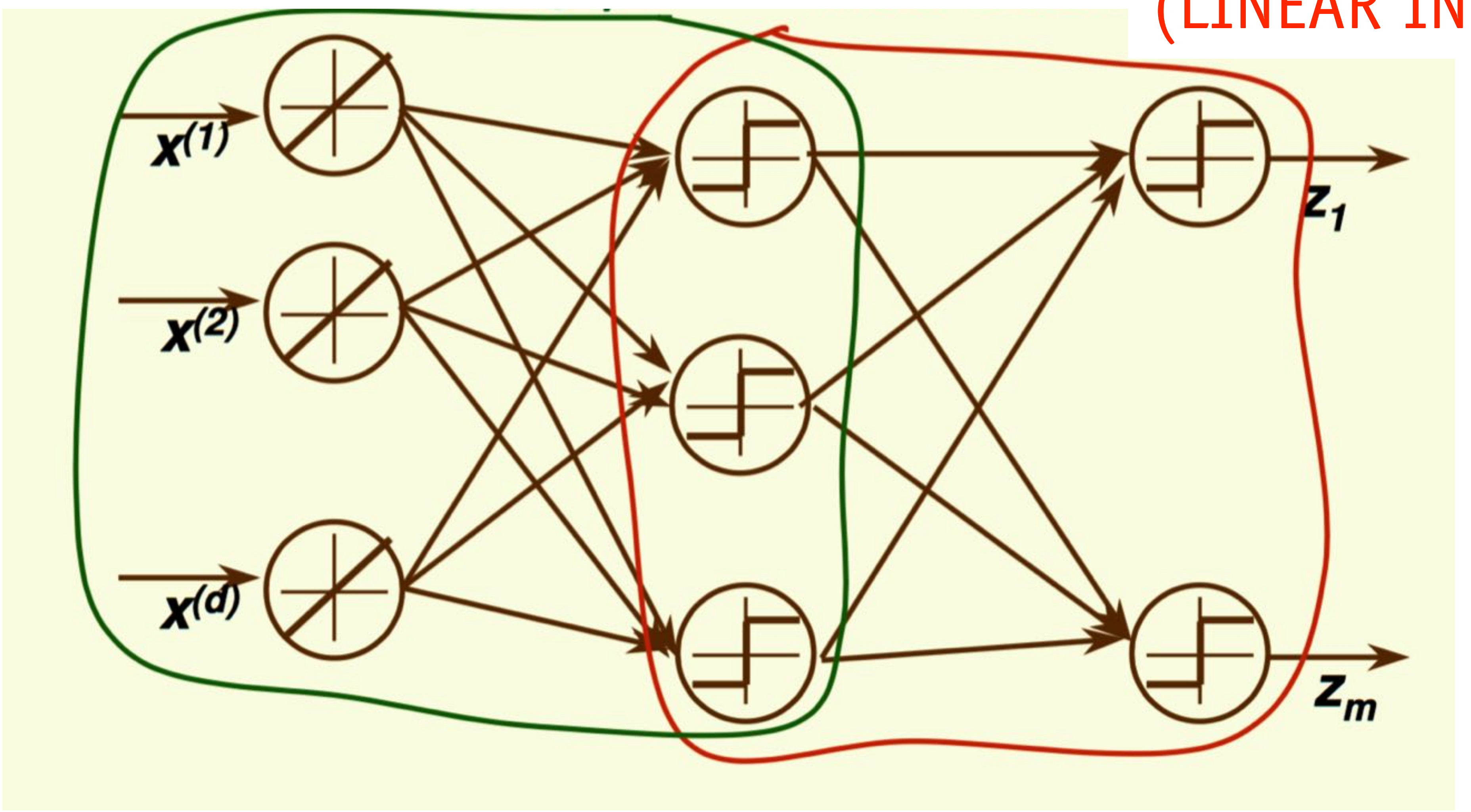
first neurone secondo neurone

ANN: INTERPRETATION AS NON LINEAR MAPPING

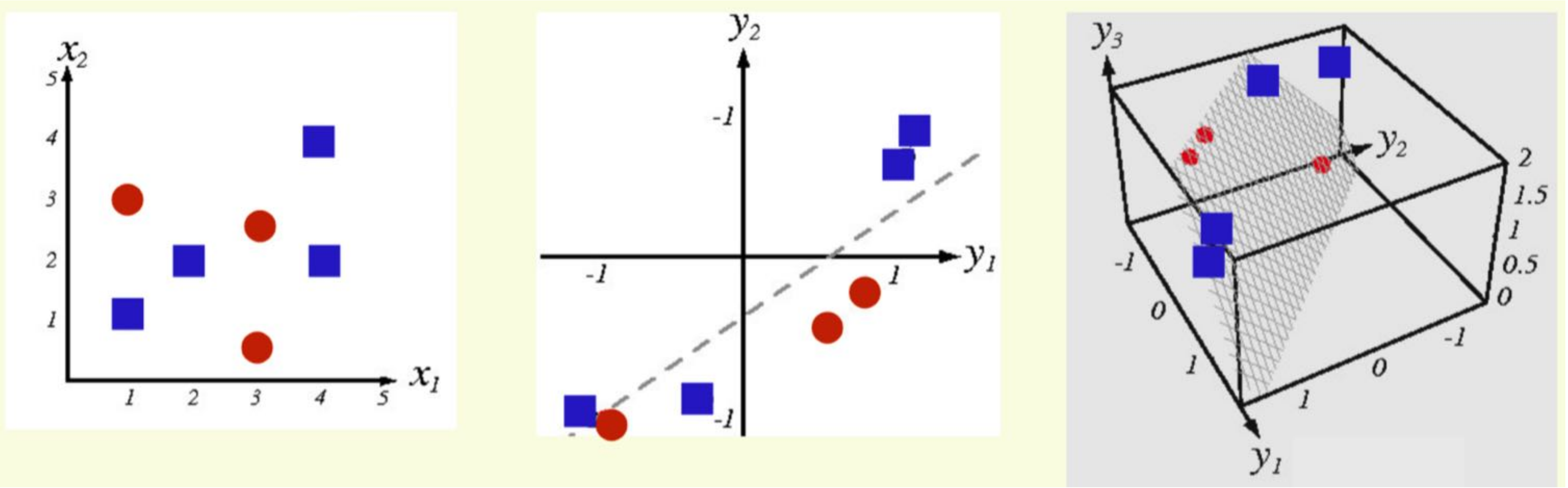
- A NN can be thought as an algorithm that learn two tasks at the same time:

THIS MODULE LEARN A (NON LINEAR) MAPPING
OF THE INPUT

THIS MODULE LEARN A CLASSIFIER
(LINEAR IN CASE OF A PERCEPTRON)



original space:
non linearly separable patterns x:

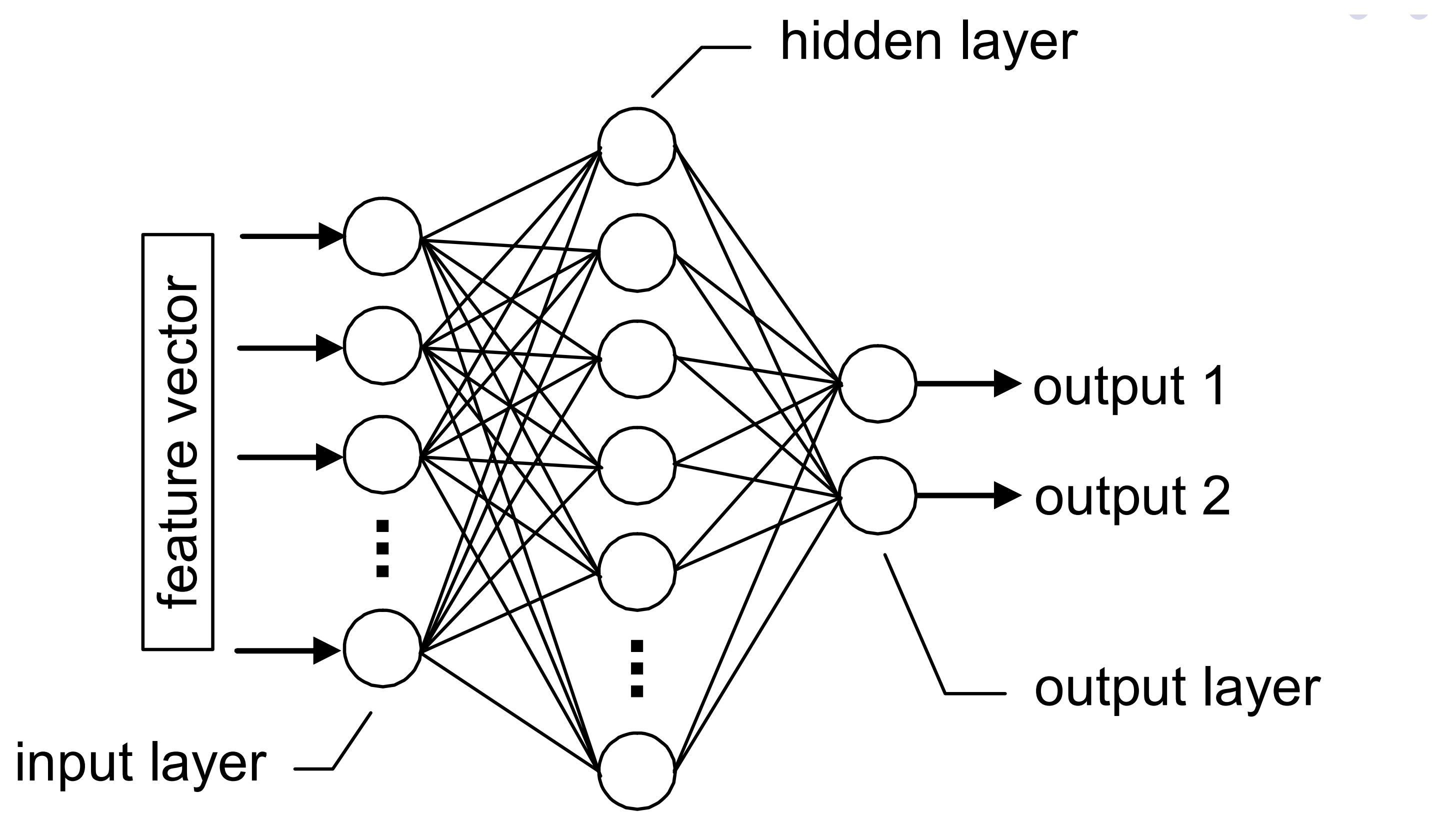


NN: finds the non linear mapping y=Φ(x) in 3-dimensional space (three hidden nodes) in which the patterns are linearly separable

NN: finds the non linear mapping y=Φ(x) in 2-dimensional space (two hidden nodes) which the patterns are “almost” linearly separable …

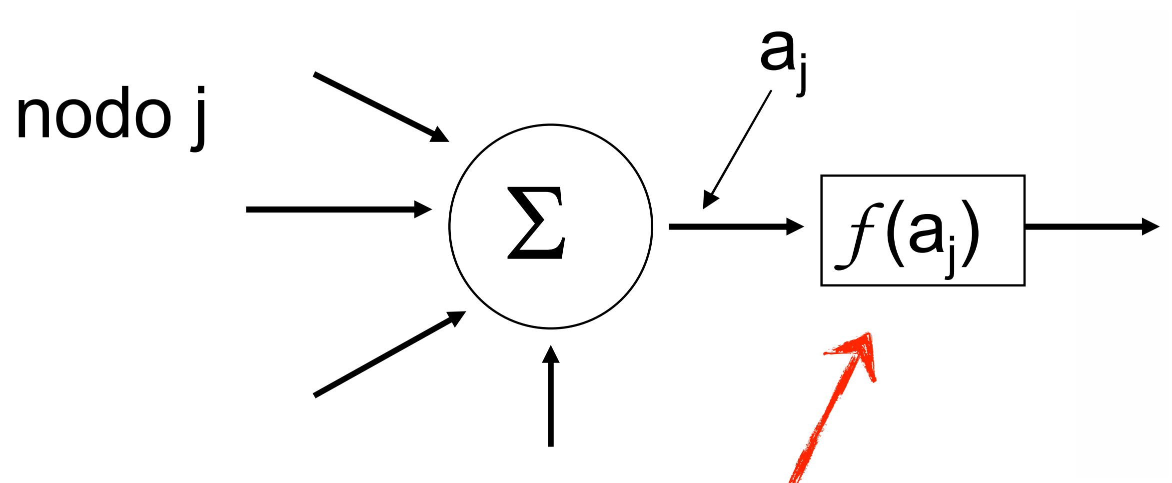
FEED-FORWARD ANN

- the most used ANN have a Feed-Forward multilayer structure:

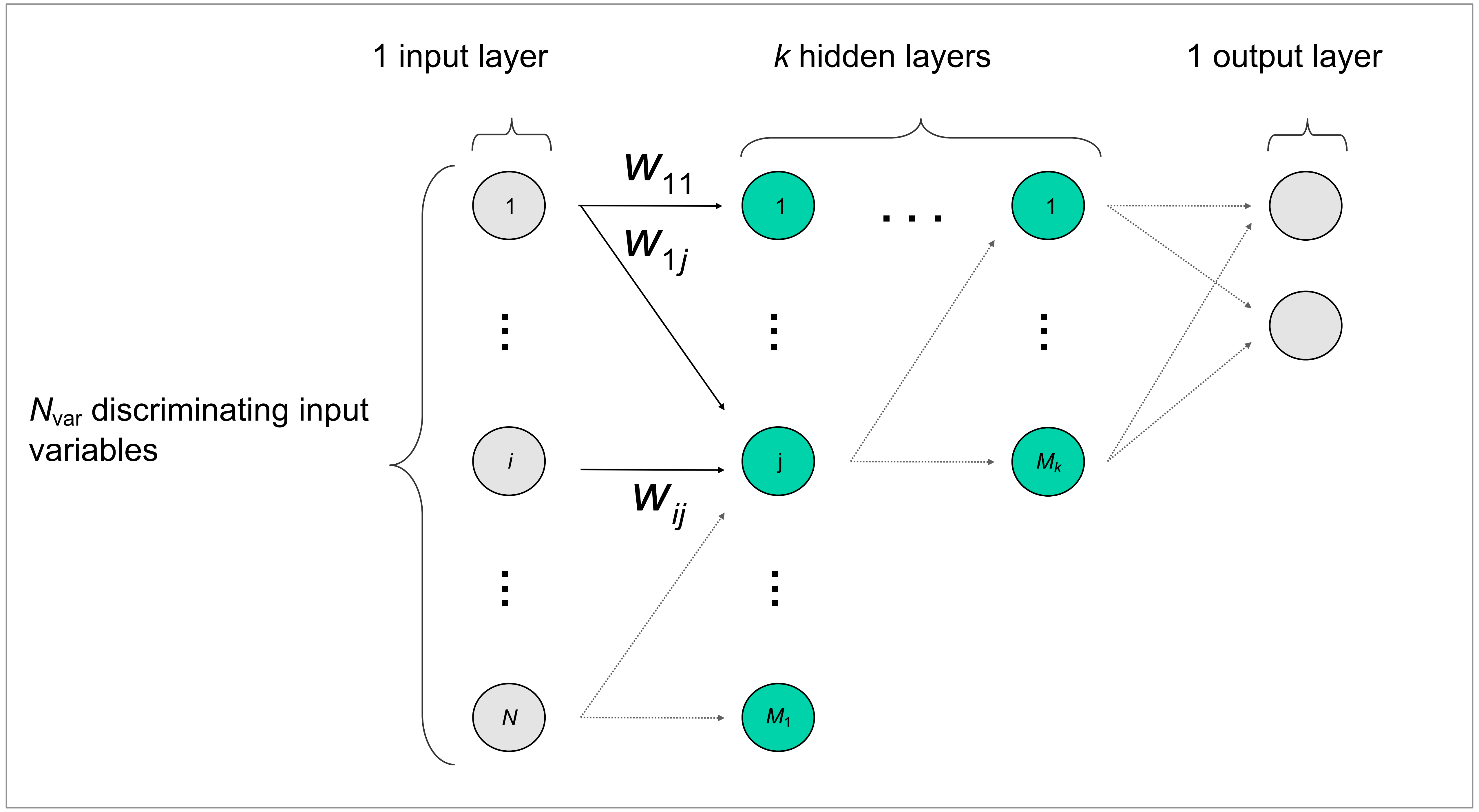


- neurons organised in layers: input, hidden-1, ... , hidden-K, output
- only connections from a given layer to the next following one are allowed

Nodo



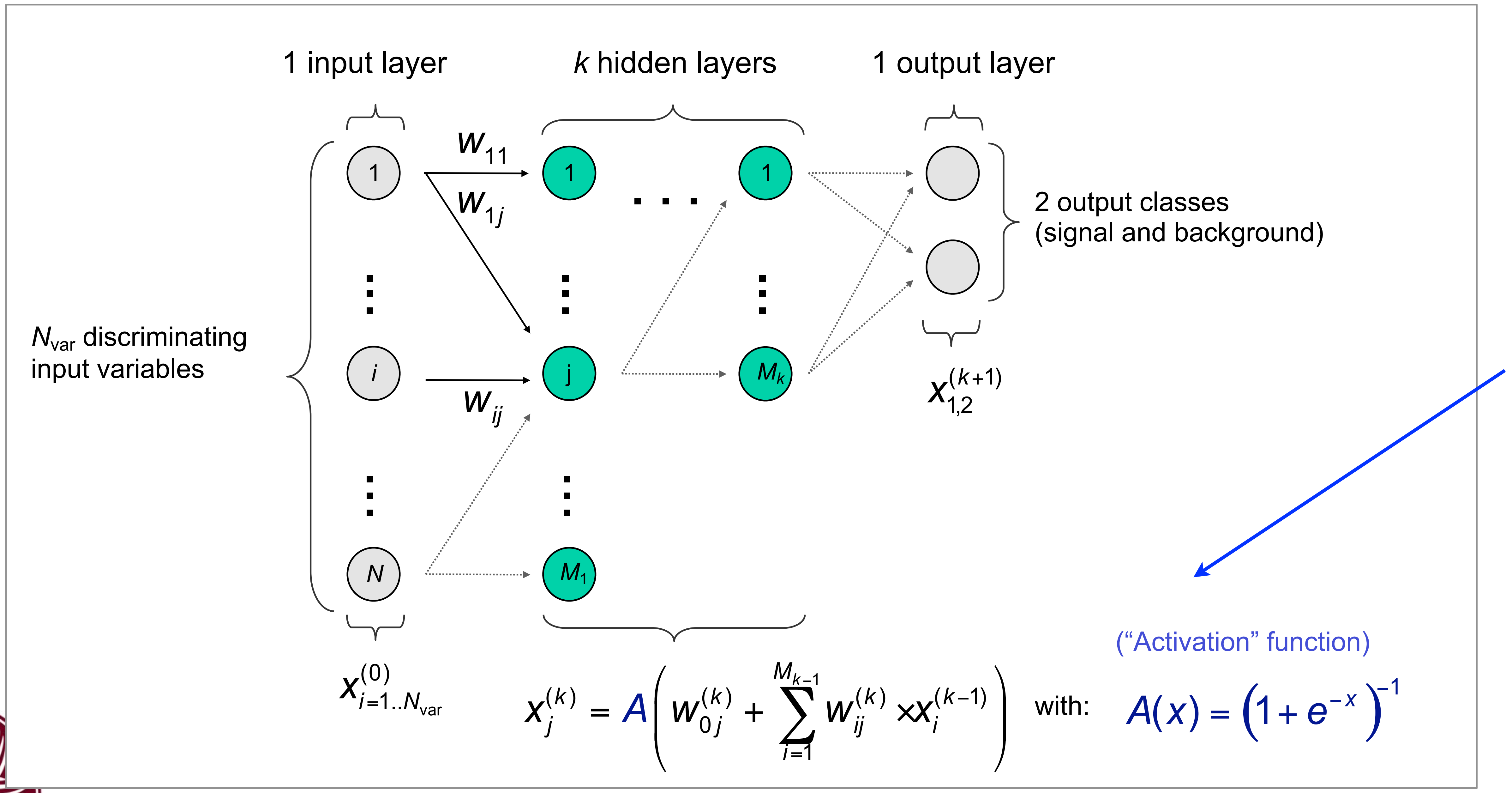
activation function (or output function)

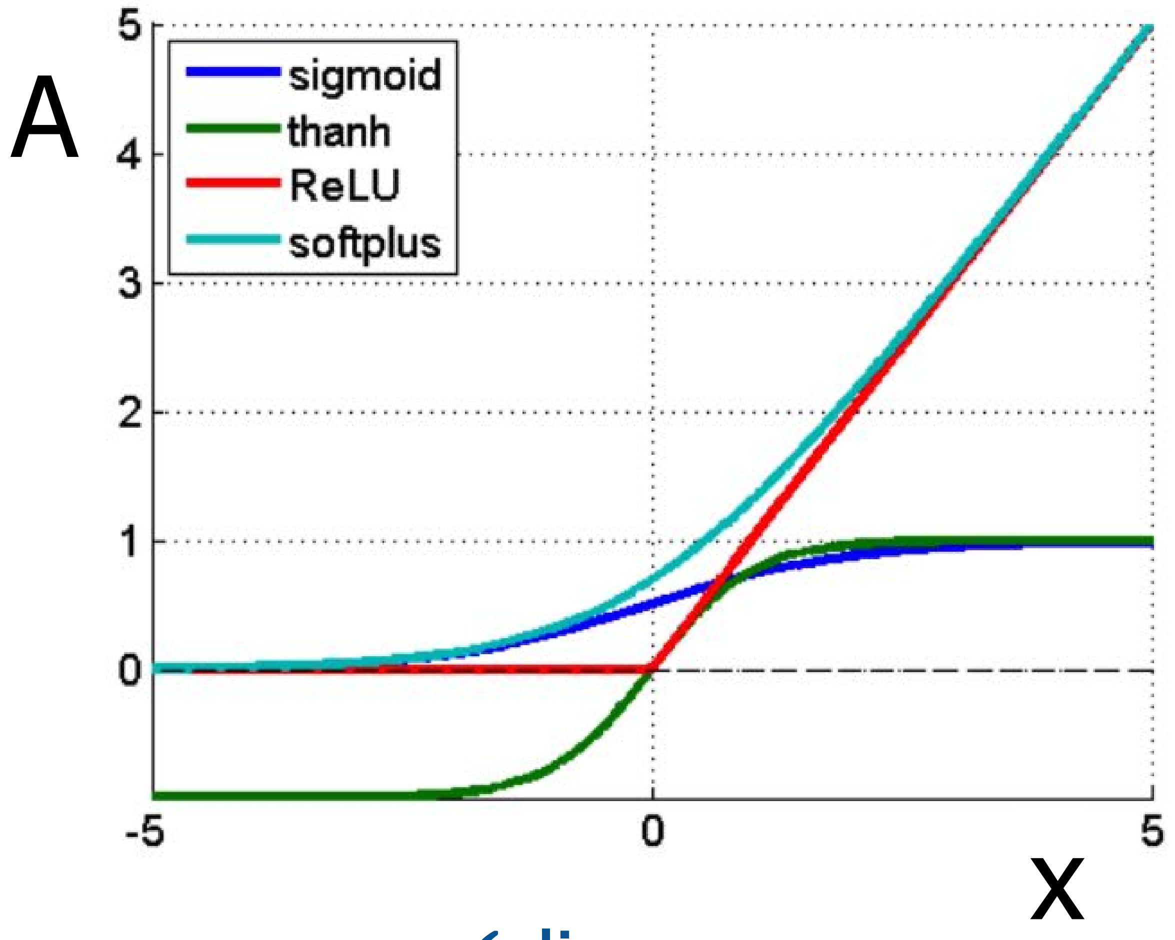


Feed-forward Multilayer Perceptron

RESPONSE FUNCTION

- behaviour of the NN determined by:
  - topological structure of the neurons (architecture)
  - Weights associated to each connection
  - response function of each neutron to the input data
- Response function ρ:
  - maps the input of the neuro n:$x^{(k-1)}{}_1,\ldots,x^{(k-1)}{}_n$ to the output $x^{(k)}{}_j$
  - normally divided in two parts: synaptic function $k:R^n\rightarrow R$ and the neural activation function $A:R\rightarrow R$: $\rho = k \bullet A$





$$A : x \rightarrow \begin{cases} \text{linear: } x \\ \text{sigmoid: } 1/(1+e^x) \\ \text{Tanh}(x) \\ \text{ReLU: } \max(0,x) \\ \text{softplus: } \log(1+e^x) \end{cases}$$

TRAINING

- The training of the NN consists in adjusting the weights (and the other hyperparameters) according to a given loss function in order to optimise the performance of the algorithm wrt a specific task
- most used technique: Back-propagation

Output for an ANN with:
- a single hidden layer with A: tanh
- an output layer with A: linear

n_h: number of hidden layer neurons

n_{var}: number of input layer neurons

$$y_{ANN} = \sum_{j=1}^{n_h} x_j^{(2)} w_{j1}^{(2)} = \sum_{j=1}^{n_h} \tanh\left(\sum_{i=1}^{n_{var}} x_i w_{ij}^{(1)}\right) w_{j1}^{(2)}$$

weight associated to the link between j-th neuron of the hidden layer and the output neutron

weight associated to the link between the i-th neuron of the input layer and the j-th neuron of the hidden layer

TRAINING

- during the training N examples are presented to the NN: $\boldsymbol{x}_a$ (a=1,..,N)
- for each event the output $y_{ANN}(a)$ is computed and compared with the expected target $Y_a \in \{0,1\}$ (0 class 2, 1 class 1 as example for a 2-class classification algorithm)
- A loss function is defined in order to measure the distance between $y_{ANN}(a)$ e Y_a:

$$\Delta(x_1, ..., x_N|\mathbf{w}) = \sum_{\mathbf{a=1}}^{\mathbf{N}} \boldsymbol{\Delta}_{\mathbf{a}}(\mathbf{x_a}|\mathbf{w}) = \sum_{\mathbf{a=1}}^{\mathbf{N}} \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{y_{ANN}}(\mathbf{a}) - \mathbf{Y_a})^{\mathbf{2}}$$

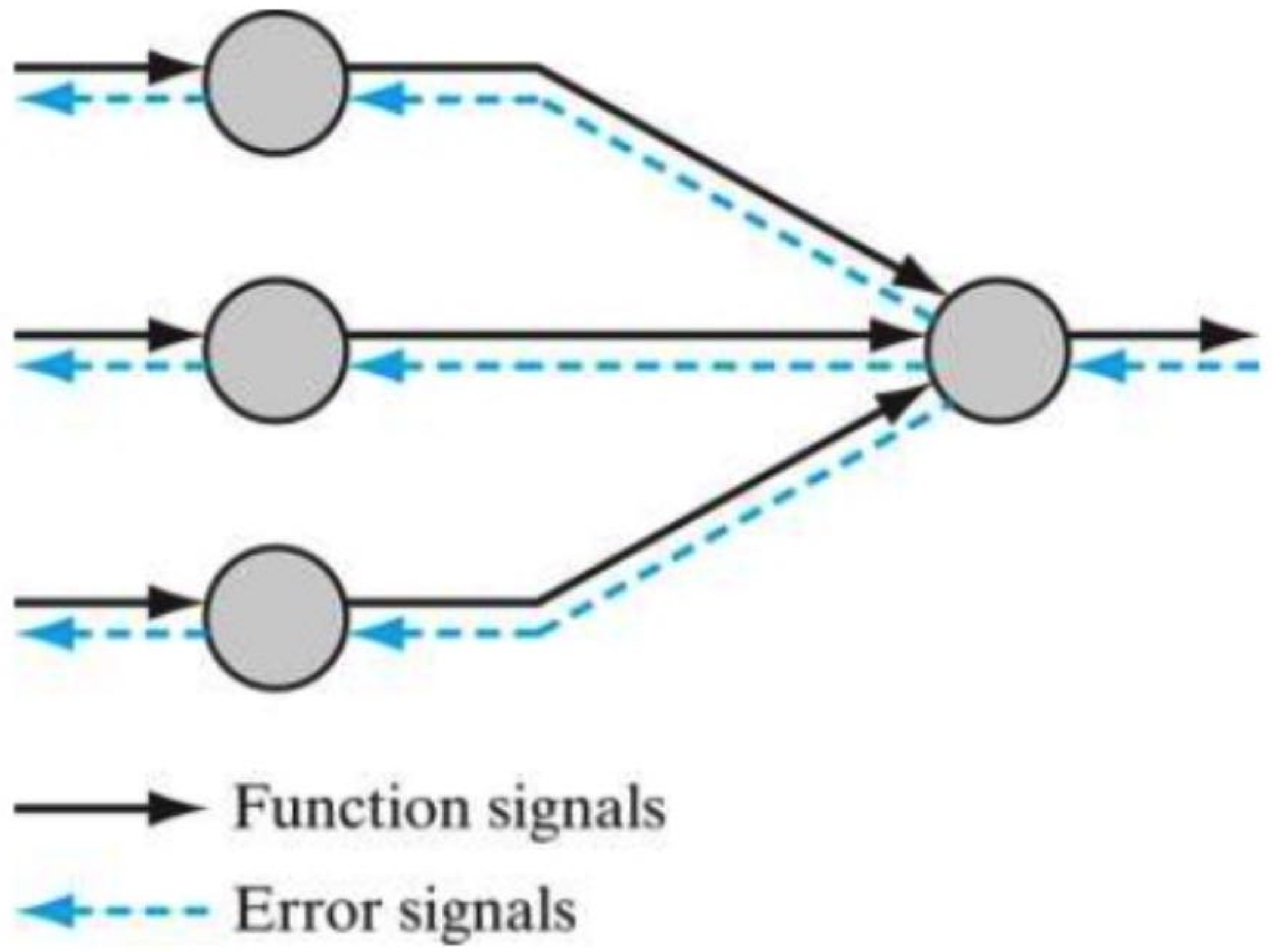
MSE

- and the weight vector is chosen as the one that minimise the error Δ
  - Minimisation obtained with the GD/SGD ...

$$\mathbf{w}^{(\rho+\mathbf{1})} = \mathbf{w}^{(\rho)} - \eta \nabla_{\mathbf{w}} \boldsymbol{\Delta}$$

BACK-PROPAGATION

- the training happens in two phases:
    - Forward phase: the weights are fixed and the input vector is propagated layer by layer up to the output neurons (function signal)
    - Backward phase: the error Δ obtained by comparing output with target is propagated backward, again layer by layer (error signal)

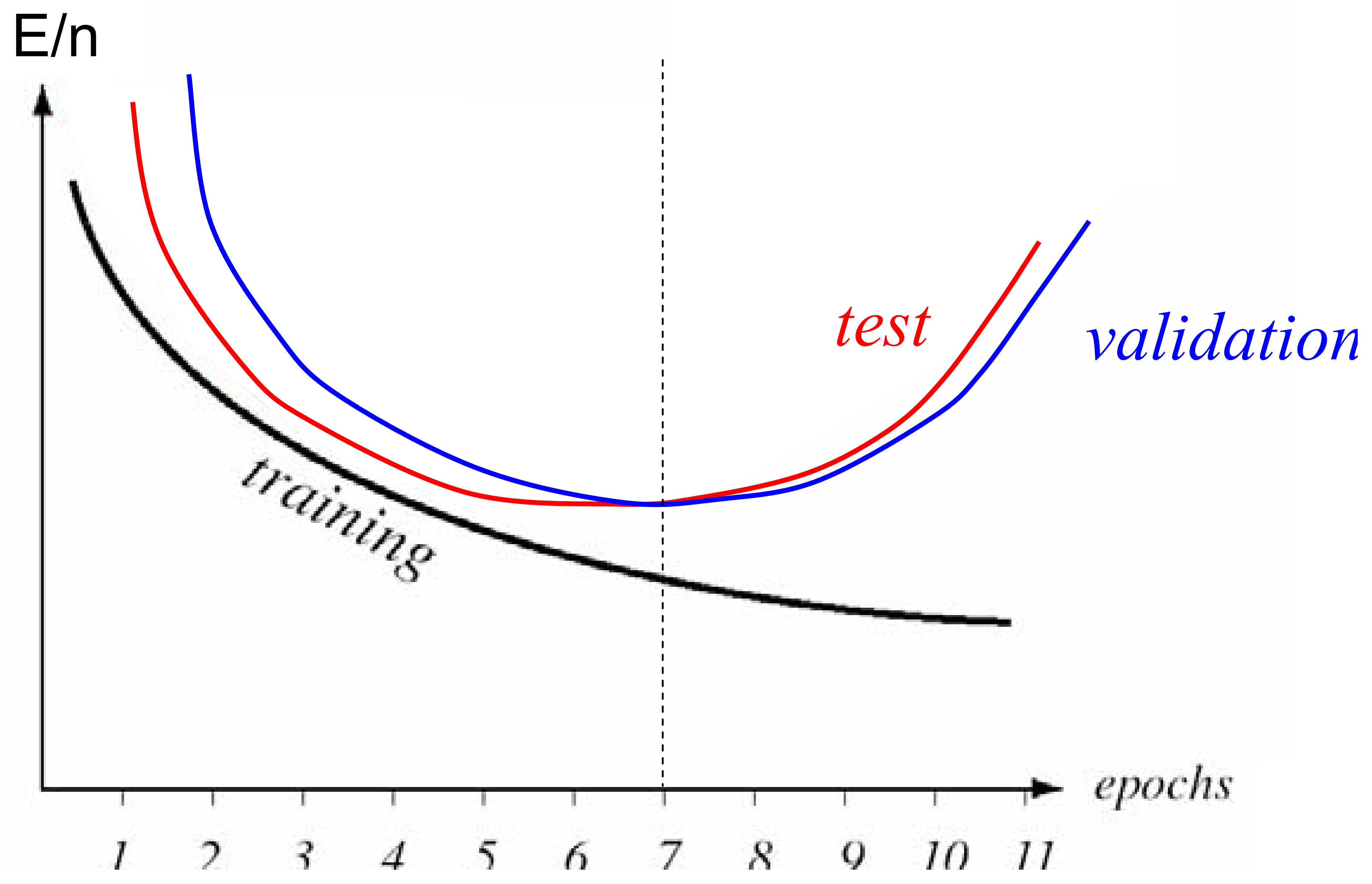


- every neuron (hidden or output) receive and compare the functioned error signals

- the back-propagation consists in a simplification of the gradient descent obtained by recursively applying the chain rule of derivatives

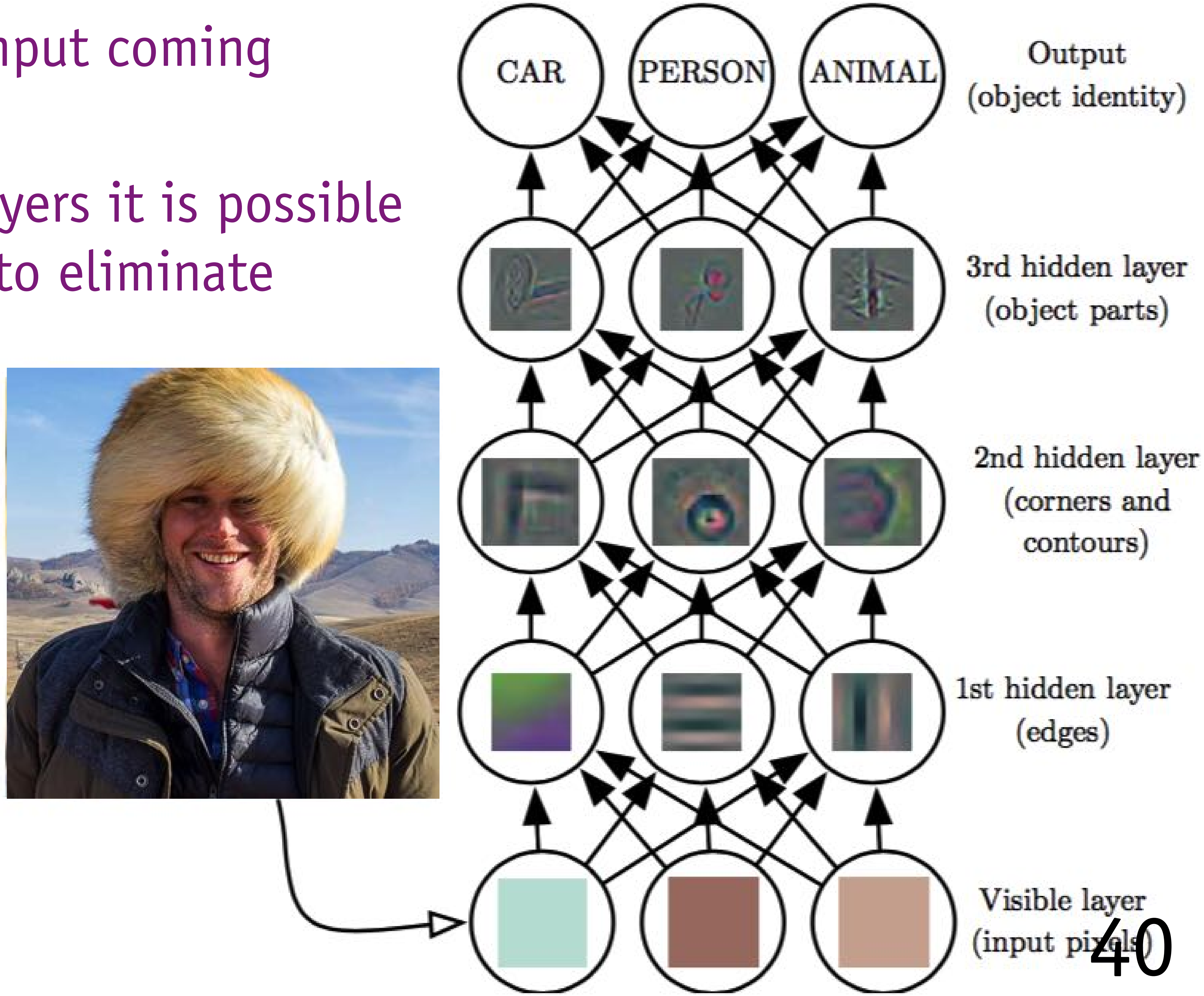
LEARNING CURVES

- at the start of the training phase the error on the training set is typically large
- with the iterations (epochs) the error tend to decrease until it reach a plateau value that depends on:
    - training set size
    - number of weights of the NN
    - initial value of the weights
- training progress is visualized with the learnign curve (error vs epochs)
- as usual multiple datasets (or cross validation) are needed to train the NN, decide the architecture, decide the stop criterion, and evaluate the final performances … etc..

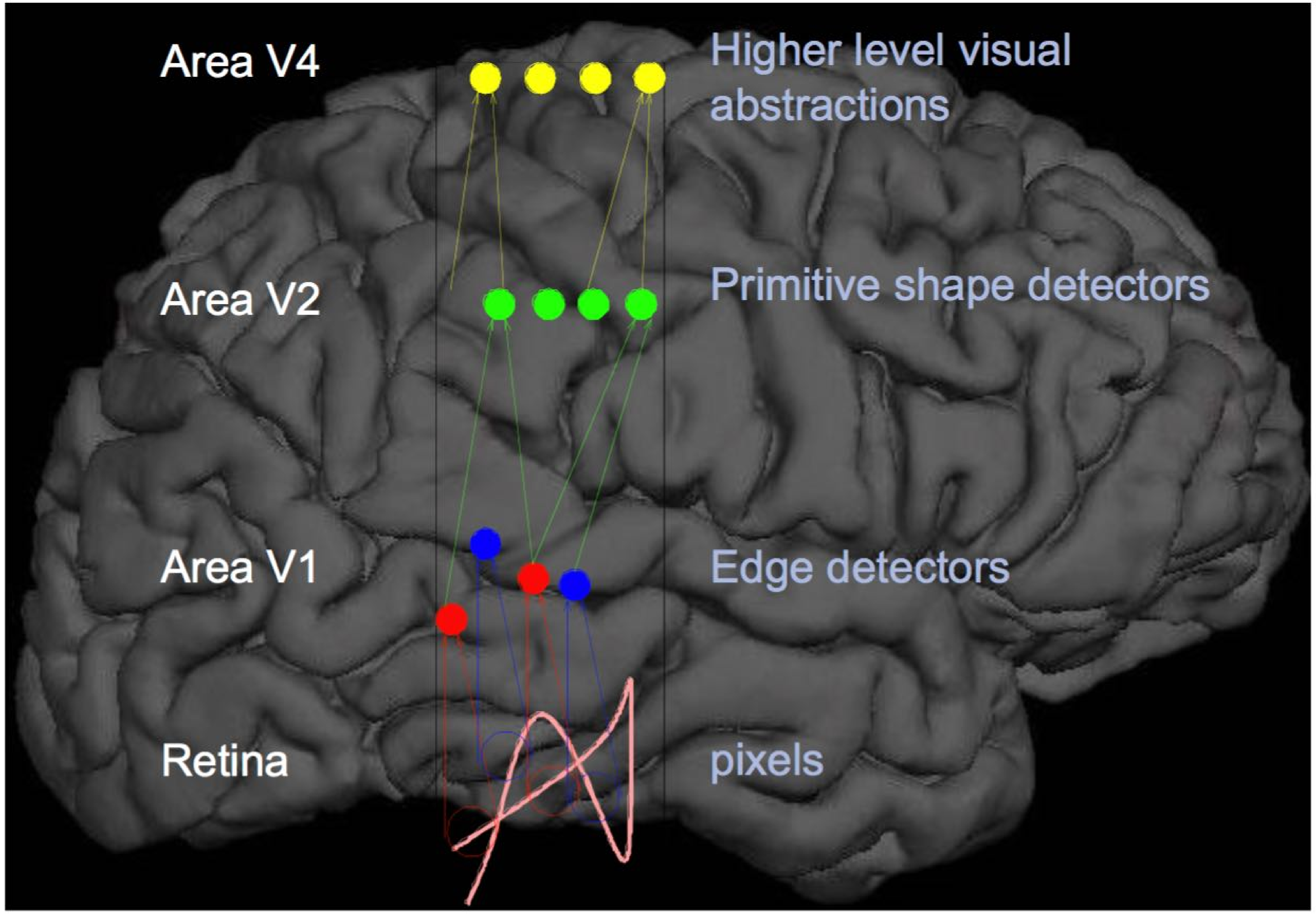


DEEP LEARNING AND ANN

- the different transformation/representation layers have a natural and intuitive implementation in multilayer neural-networks:
- each layer implements a transformation of the input coming from the preceding layer
- by using a sufficiently large number of hidden layers it is possible to learn extremely complex representations and to eliminate from the process irrilevante variations
  - example: image → array of raw pixels
  - first layer: find presence/absence of strong tonal Variations in specific points of the image (edges)
  - second layer: combines edges to find patterns like corners, contours
  - third layer: combines the previous patterns in complex objects (like faces, heads, …) that can be used to classify the content of the image …



DEEP ARCHITECTURE OF THE BRAIN



- we organise ideas and concept in hierarchical way
- first we learn simple concepts, then we compose them to represent more abstract concepts
- the DL try to emulate this behaviour …

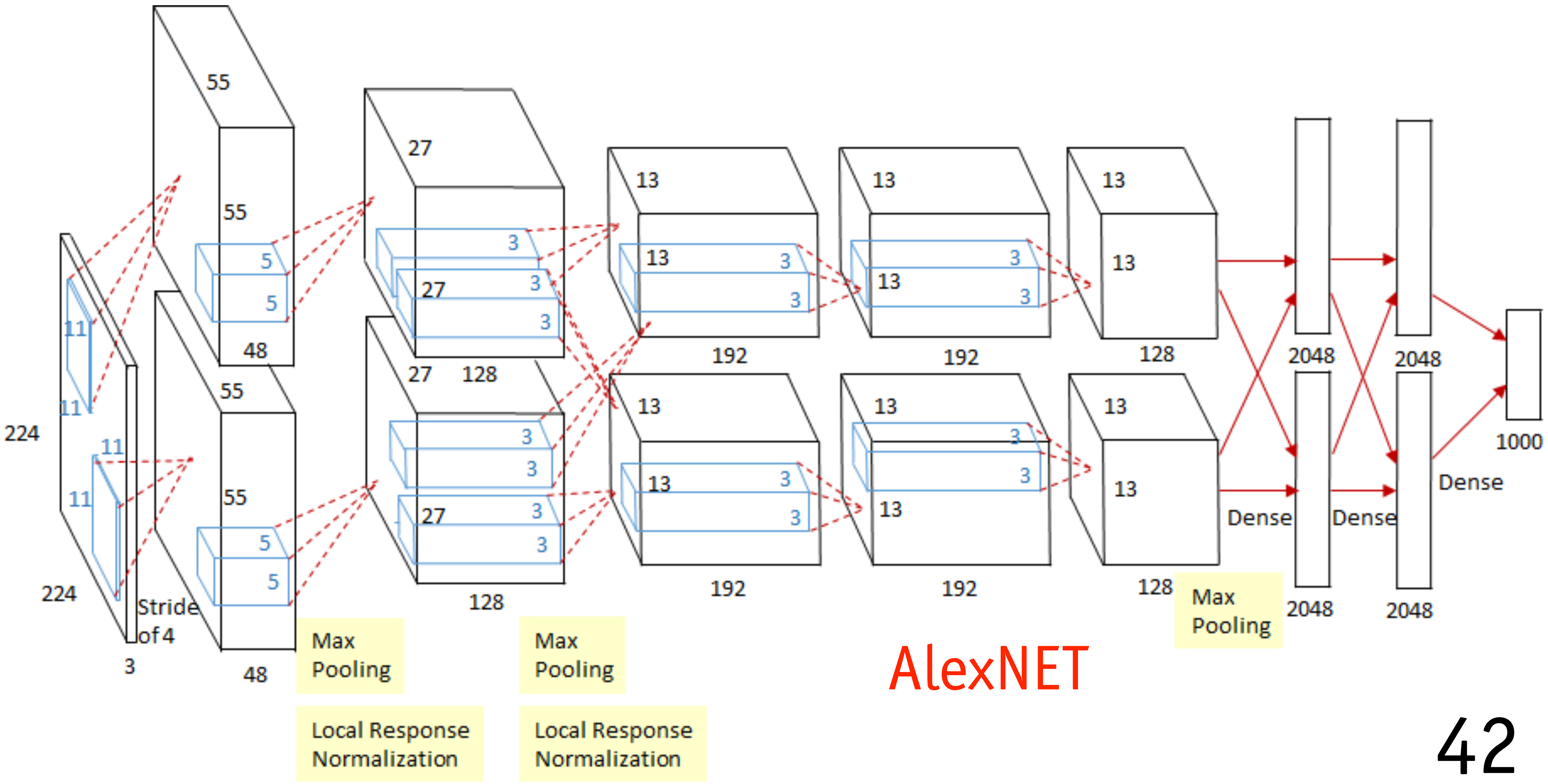
THE FIRST HIGH PERFORMANCE CNN: ALEXNET

- NN based on the architecture of Krizhevsky et Al. winner of Imagnet 2012 Contest
- developed under Caffe framework (Berkeley Vision Deep Learning framework: http://caffe.berkeleyvision.org)
- instructions to install it on mac os x: https://vimeo.com/101582001

same top-down approach as LeNet with successive filters designed to capture more and more subtle features

\+ improvements:

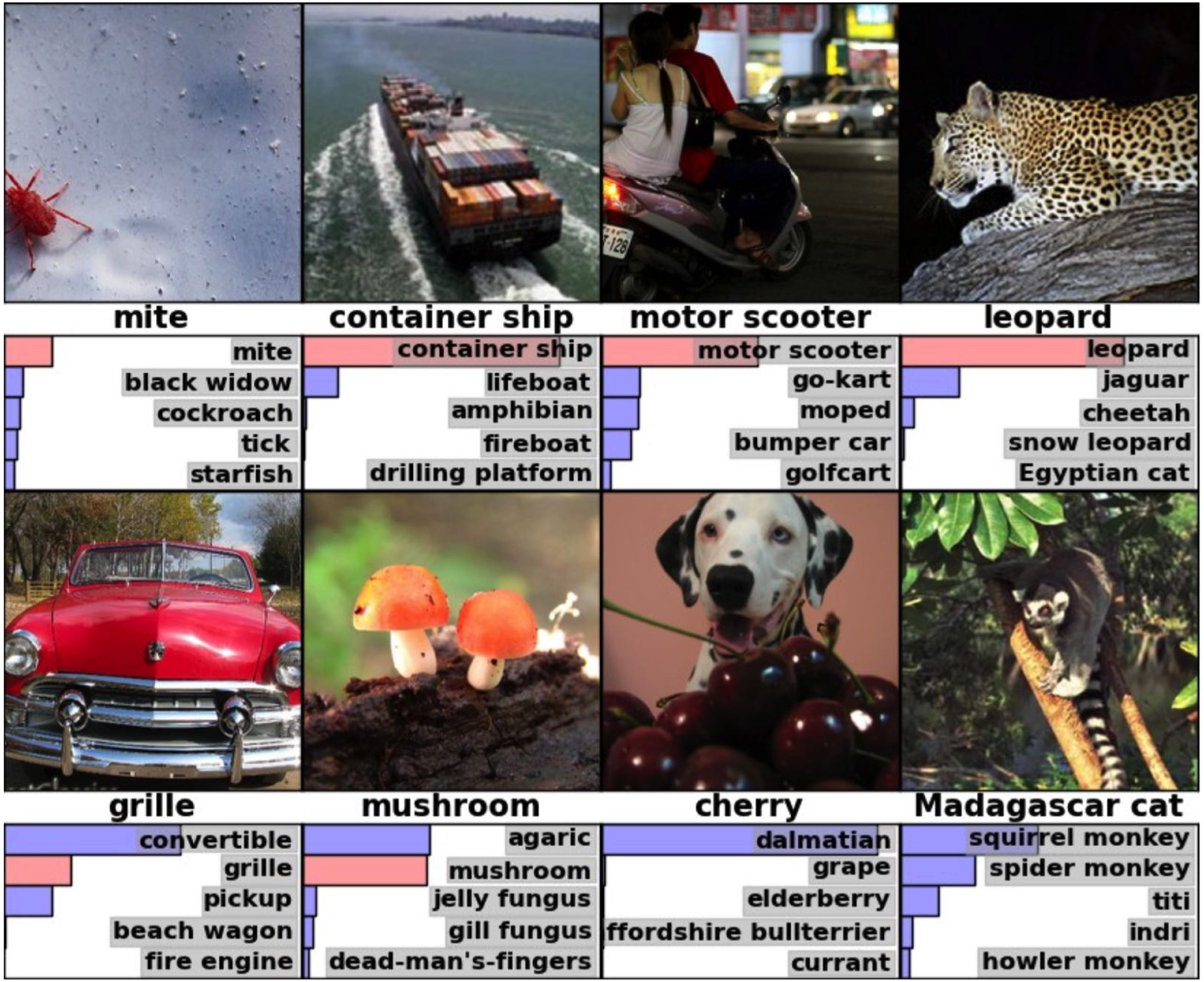
1. better back-propagation via ReLU
2. dropout based regularisation
3. batch normalisation
4. data augmentation: images presented to the NN during training with random translation, rotation, crop
5. deeper architecture: more convolutional layers (7), i.e. more finer features captured



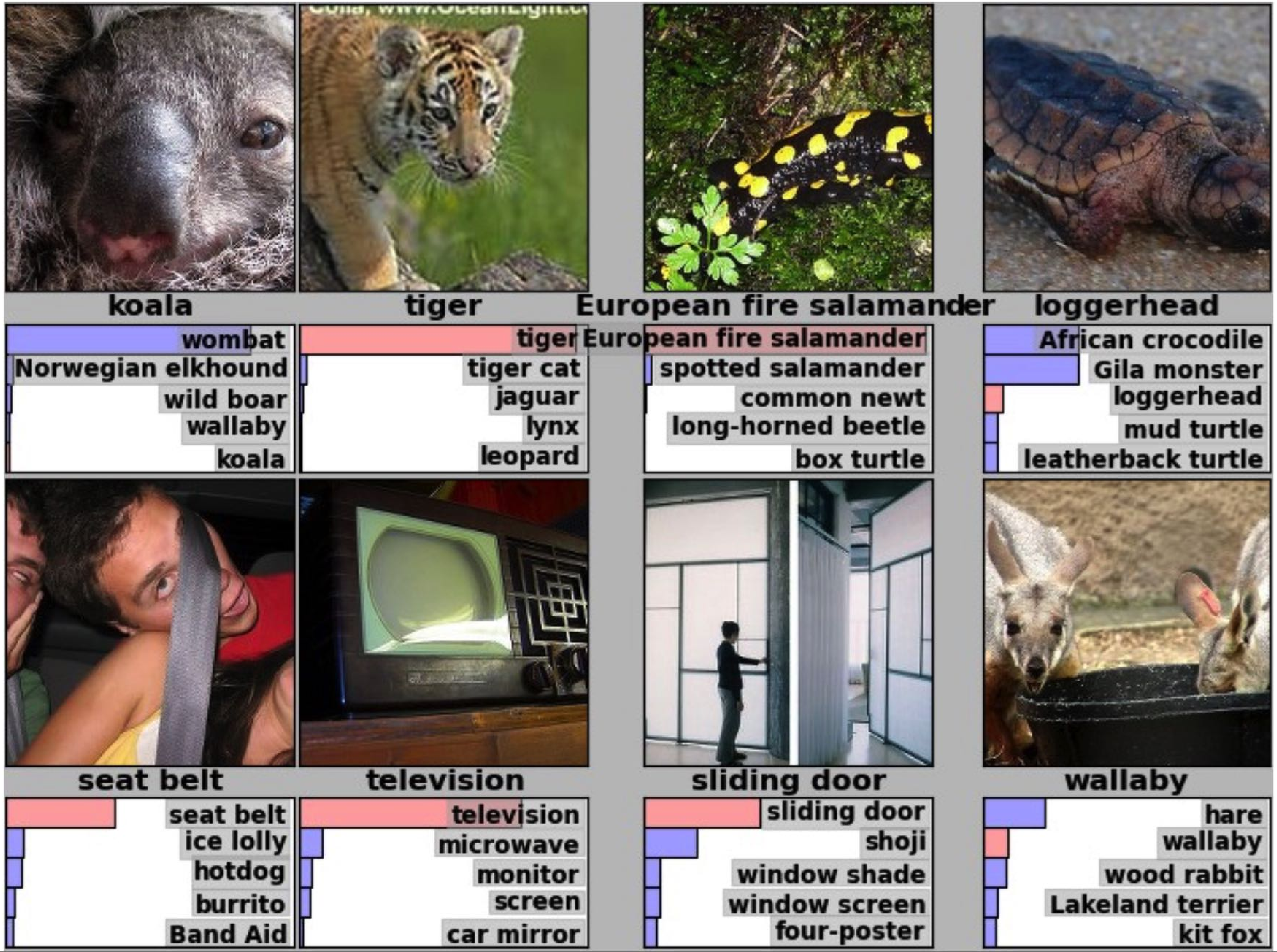
AlexNET

Validation classification

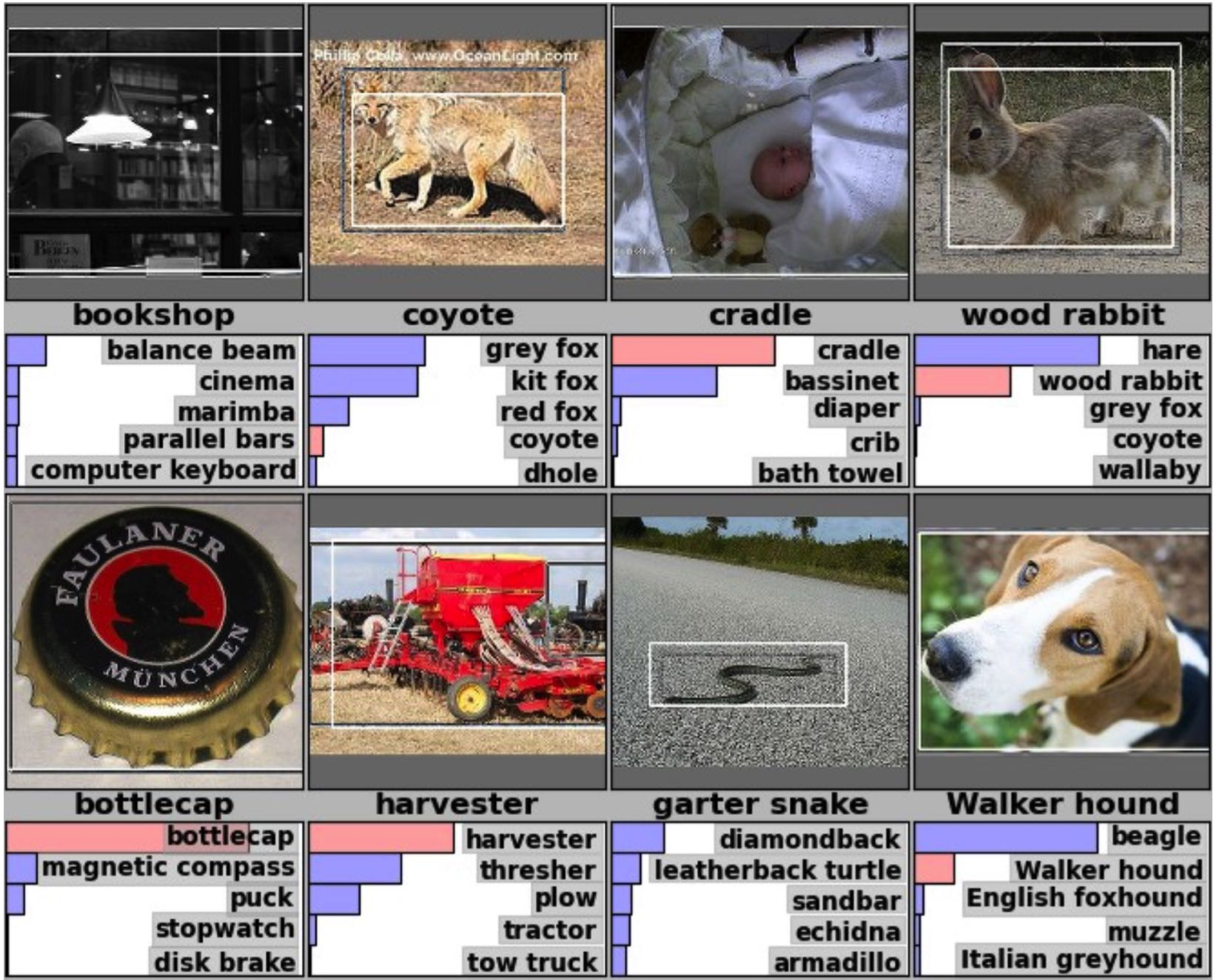
- feature initialised with white gaussian noise
- fully supervised training
- training on GPU NVIDIA for ~1 week
- 650K neurons
- 60M parameters
- 630M connections



Validation classification



Validation localizations



Retrieval experiments

First column contains query images from ILSVRC-2010 test set, remaining columns contain retrieved images from training set.