Lecture 1: Introduction

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ

Contents

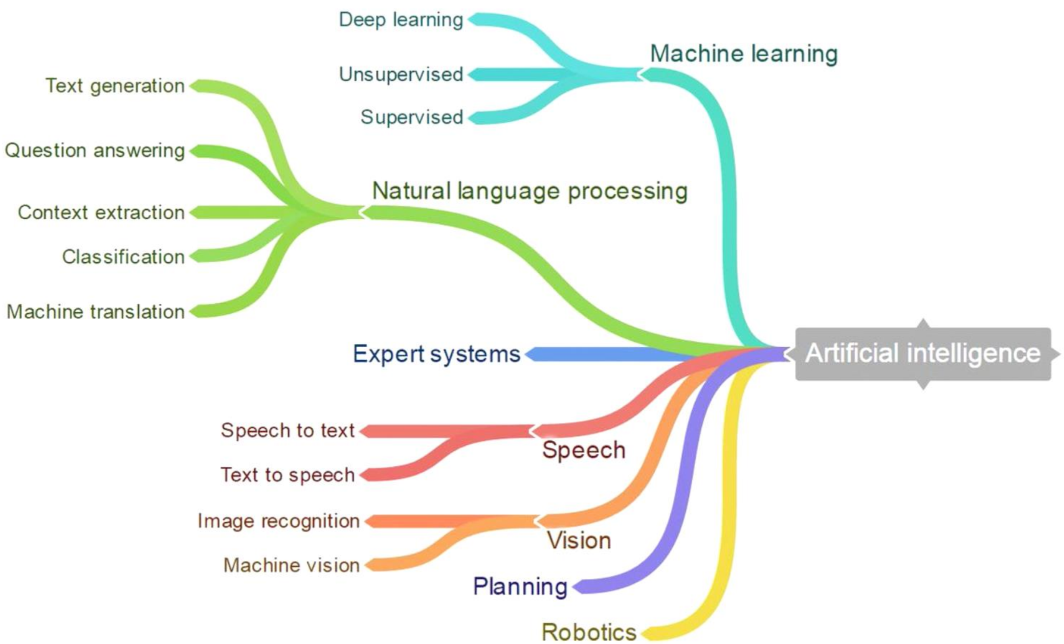
- Machine learning in the context of Artificial Intelligence
- Description of the problem and basic concepts
- Regression tasks
- ML design cycle
- Exploitation (maybe as part of a perception system)
- Flavours of machine learning
- Development framework (suggested)

Machine learning in AI

- **Artificial Intelligence**, a couple of definitions
  - AI as a **discipline**:

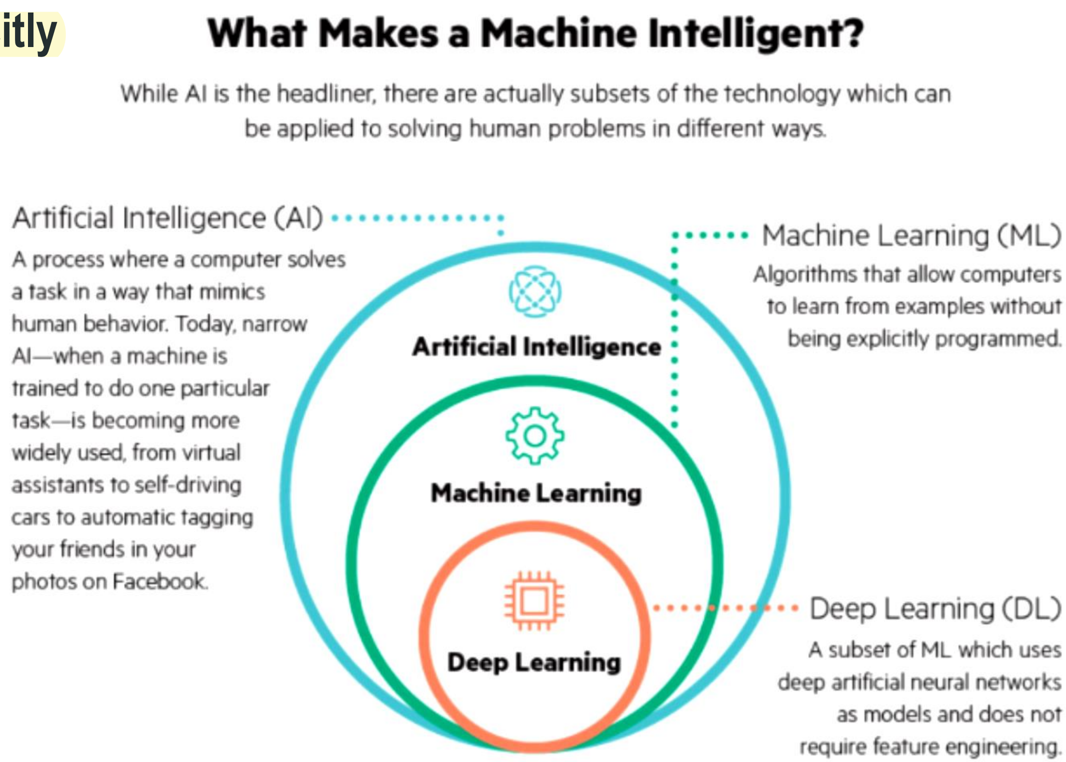
    A branch of computer science dealing with the simulation of intelligent behaviour in computers
  - AI as a **property of a machine**:

    The capability of a machine to imitate intelligent human behaviour
- **AI technologies**
  - set of rich sub-disciplines and methodologies:
    - machine learning
    - intelligent sensor data processing
      - image processing & computer vision
    - expert systems
    - robotics
    - …



Machine learning in AI

- **Machine learning**, a first definition
  - a subset of artificial intelligence in the field of computer science that makes use of a **varied set of techniques** to give computers
    - the ability to **learn from data**
      - solve a problem on the basis of “previous experience” (= collected data)
      - maybe also progressively improve performance on the specific task
    - without being **explicitly programmed**



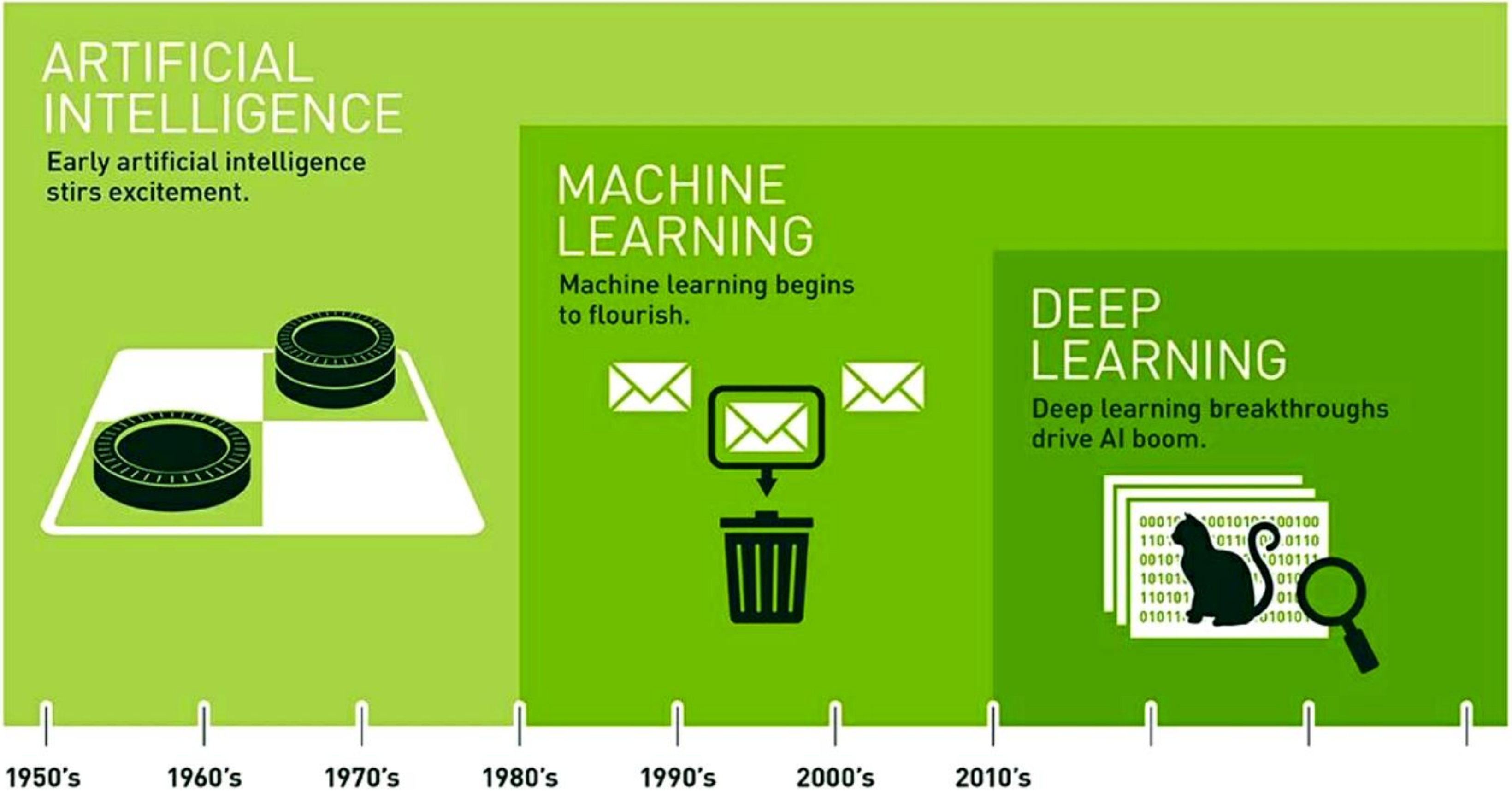
Machine learning in AI

- **Machine learning**, a formalization (Tom Mitchell, CMU 1998)

  – A computer program 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**

- **Machine learning**, a brief chronology



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

- **Machine learning**, a brief chronology

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

VOL. LIX. NO. 236.] [October, 1950

MIND

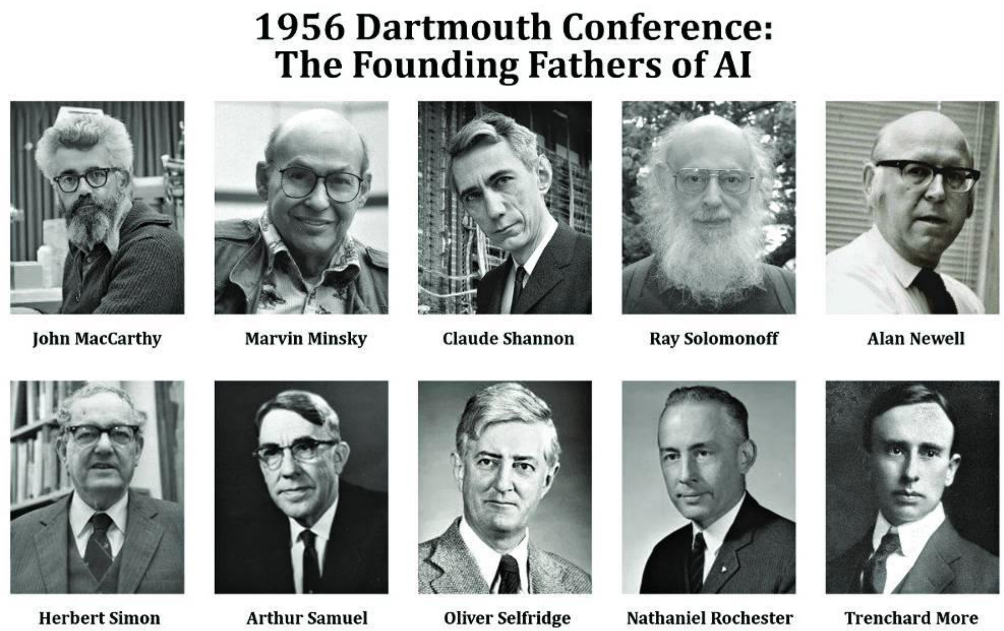
A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING



Machine learning in AI

- **Why now?**
  - Availability of computational power
    - Hardware (GPU, TPU) at reasonable cost
    - Cloud computing
    - Case of perception systems: availability of low-cost sensors, e.g. vision cameras
  - Availability of data
    - Datasets publicly available in general
    - Tons of data produced and available, “big data” (90% produced during the last years)
  - Democratization of tools
    - Open source tools and frameworks
    - Wide adoption in the research community and also companies
  - Economical value from AI
    - More funding
    - More research
    - More applications

Contents

Description of the problem

- At the **biological level** we can find multiple examples of perception systems
- In particular, over the course of their evolution, **humans** have succeeded in developing highly sophisticated systems capable of extracting information from the environment
  - We are able to recognize faces in a straightforward fashion,
  - We understand spoken language independently of the accent,
  - We can read handwritten characters, most times without noticeable effort,
  - We can identify the car keys in our pocket, among other keys, just by touch,
  - We can decide if an apple is rotten by its smell, etc.
- This ability is crucial for the **survival** of any species, e.g. recognition of friends/enemies/predators, food, ...
- However, it is **not so straightforward** for a computer

Description of the problem

- A simple example:

  classify fish arriving on a conveyor belt based on the information provided by a vision camera

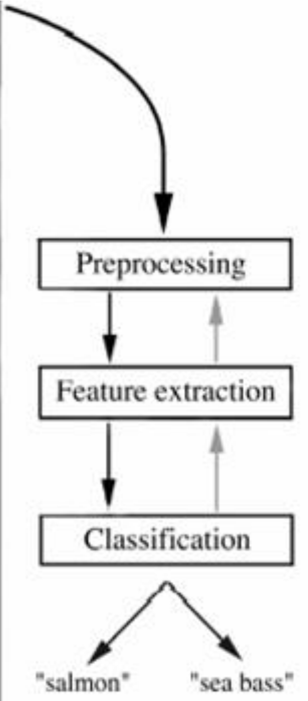
  – Discriminate between **salmon** and **sea bass**

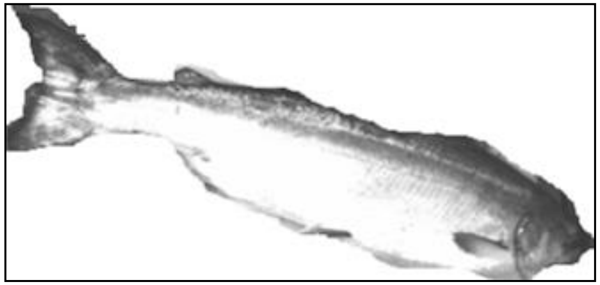
  – The image is **pre-processed** as much as needed, e.g. isolate the fish instances that appear in the image (segmentation)

  – A **feature extractor** is able to measure key properties of every piece

  – A **classifier** runs on the selected features

  (data flow can be bi-directional, stages can cooperate among them)

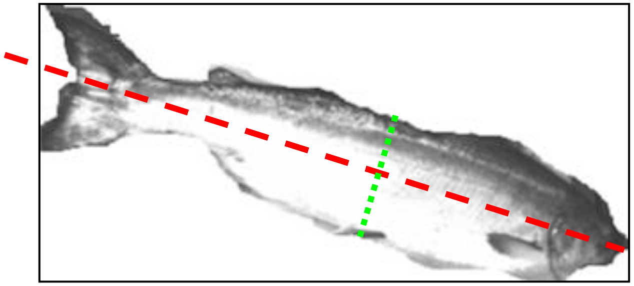




pre-procesing

Description of the problem

- A simple example (cont.):
  - We can consider several properties of every piece:
    - length
    - brightness (i.e. gray-level)
    - width
    - number and shape of the fins
    - position of the mouth, etc.



  - One has to take into account the **variability** in the chosen property, together with:
    - Variations in the gray-level at every pixel:
      - Non-uniform reflectance
      - Non-uniform illumination
      - Shadows, specularities (glossiness)

      } control the image capture conditions
    - Position of the piece over the belt,
    - Camera noise,
    - Noise from the pre-processing stage

$\Rightarrow$ One cannot expect the same value in all measurements

Description of the problem

- A simple example(cont.):
  - We have been informed that salmons tend to be shorter than sea bass pieces
  - We take a number of **samples** (= images) **for training** (200-300) and build a histogram:

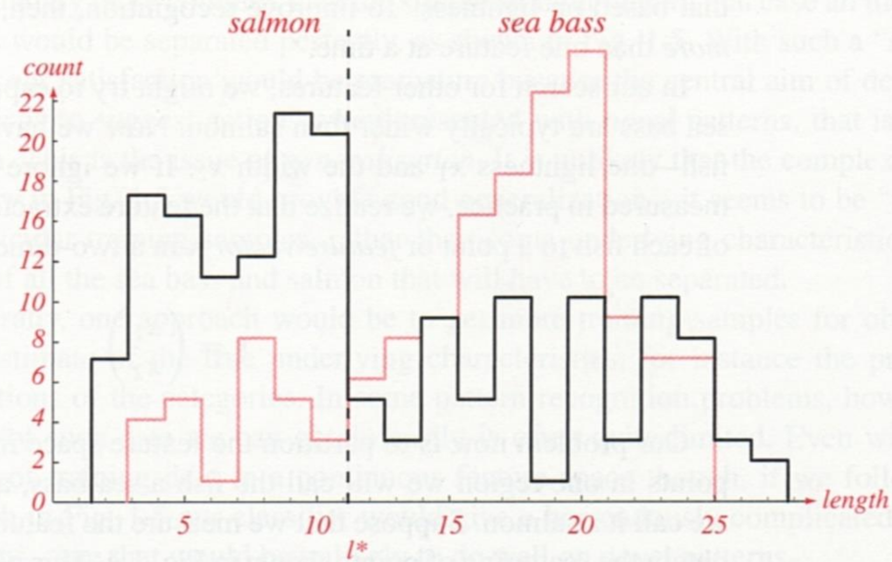


FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked *l** will lead to the smallest number of errors, on average.

  - As can be seen, the piece length by itself is a rather **poor criterion** to discriminate in a trustworthy way between the two species

Description of the problem

- A simple example (cont.):
  - We consider another feature: the average gray level of the scales

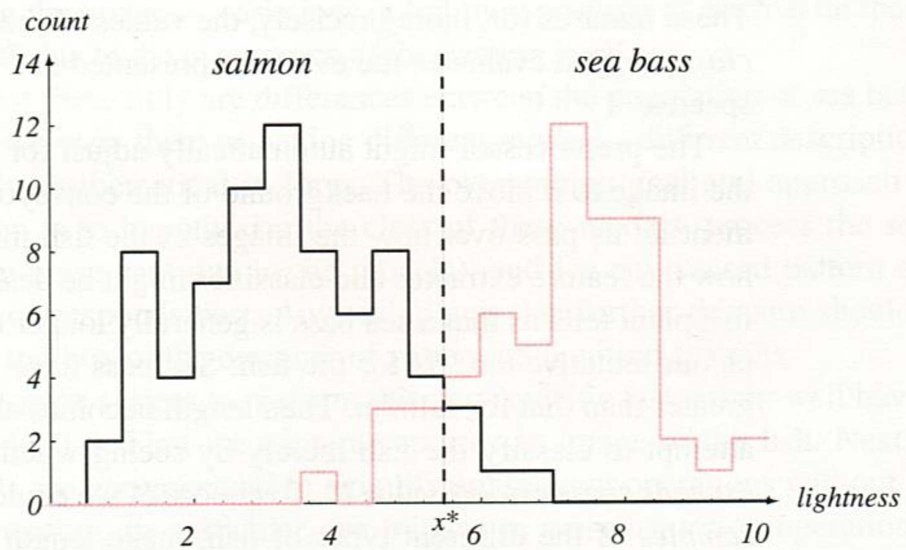


FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors, on average.

  - The resulting histograms and the **critical value x*** are much more satisfactory since **classes are better separated**

Description of the problem

- A simple example (cont.):
  - Feature selection and evaluation of the system
    - associate a cost to each misclassification and optimize the cost among the different possible features
      - Look for **x^* that minimizes total cost** to set an **optimal decision rule**
    - so far we have assumed that the cost of a misclassification is **symmetrical**: it is just as wrong to confuse sea bass with salmon as it is to do the opposite
      - However, customers are not likely to think the same …
        $\Rightarrow$ define the **types of error** and associate a different cost to each
  - Let us assume we have tested all the features separately. Now, it is turn to **try with several features simultaneously** …

Description of the problem

- A simple example (cont.):
  - We observe that the sea bass tends to be wider than salmons …
  - … and so we have a **vector of features**, so-called as a **descriptor**:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \text{gray level} \\ \text{width} \end{pmatrix}$$

¡¡ the feature extractor has reduced the image of every piece to a point in a plane !!

  - The problem has now become into partitioning the feature space into two regions and find a **decision curve (2D)**

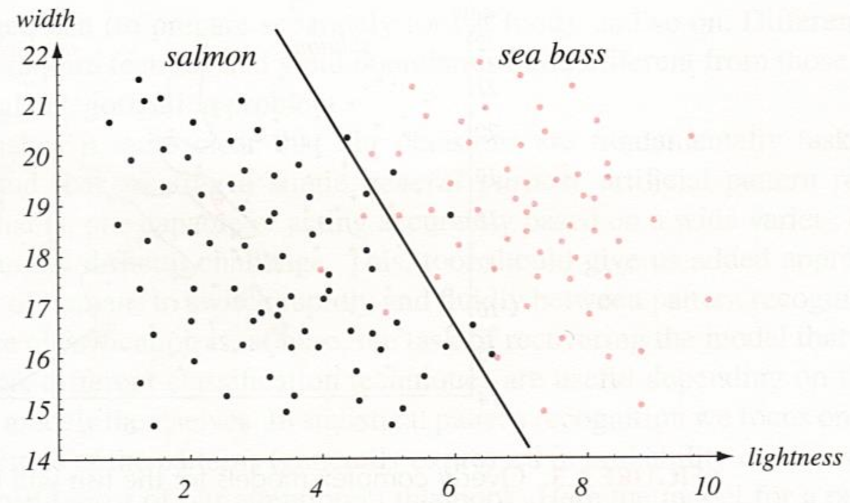


FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors.

- A simple example (cont.):
  - the best decision curve is that one that classifies **in an optimal way the samples of the training set** ?

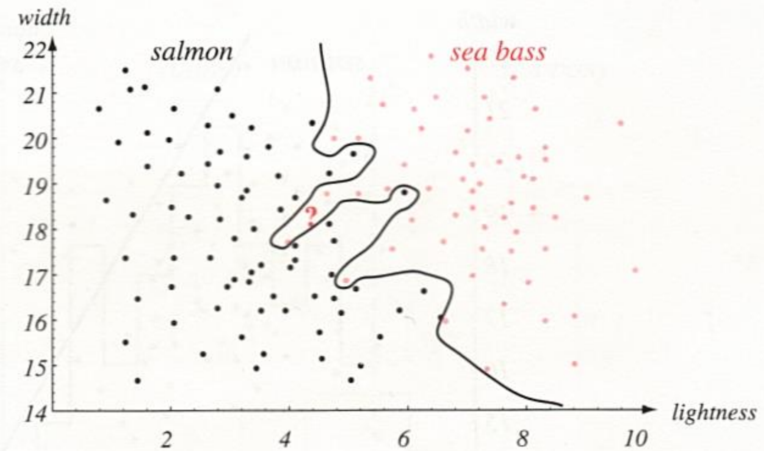


FIGURE 1.5. Overly complex models for the fish will lead to decision boundaries that are complicated. While such a decision may lead to perfect classification of our training samples, it would lead to poor performance on future patterns. The novel test point marked ? is evidently most likely a salmon, whereas the complex decision boundary shown leads it to be classified as a sea bass.

  - would this model classify samples out from the training set with the **same level of performance**?

    this decision curve is overfitted to the training set !! (**overfitting**)

- A simple example (cont.):
  - The following decision curve could be a good compromise between performance on the training set, simplicity of the classifier and behaviour with new samples

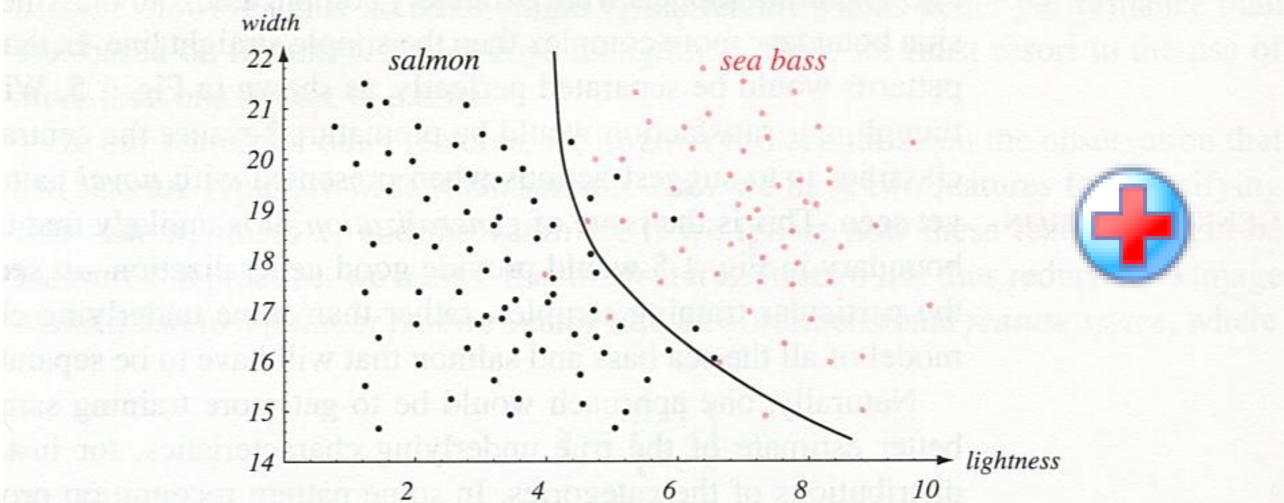


FIGURE 1.6. The decision boundary shown might represent the optimal tradeoff between performance on the training set and simplicity of classifier, thereby giving the highest accuracy on new patterns.

Description of the problem

- A simple example (cont.):
  - The key point is that the classifier is capable of dealing well with as much samples as possible with unseen samples $\rightarrow$ problem of **generalization**
    - It is not a matter of huge amounts of data, but of a **training set well representing the classification problem**
      - Notice that the classifier would be perfect only if all possible cases were available
    - It is better a classifier not so good with the training set but that **generalizes well** and is capable of classifying correctly samples not used for training
    - On the other side:
      - William of Occam, 1280-1347?

        Entia non sunt multiplicanda praeter necessitatem
        (entities should not be multiplied unless needed)

        $\equiv$ **under equal conditions, the simplest model is the likeliest to be correct**
        (Occam's razor)

Description of the problem

- Description of the problem (cont.):
  - We could add other features: e.g. the eye color

$$\vec{x} \in \overbrace{\mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_n}^{\text{feature space}} \ (\text{e.g. } \vec{x} \in \mathcal{R}^n) \ \overset{\text{nD}}{\rightarrow} \ \textbf{decision rule}$$

(surface, hypersurface)

- Features should be informative in order to **separate better** the classes (= **uncorrelated** with the ones that we already have)
- New features **should not reduce the effectivity** of the classifier $\Rightarrow$ remove noisy features
- The **computational cost** associated to calculating each new feature has to be taken into account

❶ Working in **n dimensions** is not for free, adding a new dimension improves the effectivity in a significative way?

❷ **Real-time operation** is necessary? Is it possible at the computational level?

e.g. recognize zip codes: conveyor belt for letters distribution moves at a speed of *v* cm/s to classify *t* letters per hour

Basic concepts

- So far, some **basic concepts** from machine learning have emerged:
  - Samples are represented by means of **descriptors**
    - Designed to capture the relevant information for the specific classification problem, aiming at removing any **unnecessary complexity**:
      - Ideally, the representation should disclose in a simple and natural way the **structure of the classes** in **feature space**
      - A good representation is a key aspect of any ML problem
    - Usually, descriptors are **n-dimensional vectors**:

$$\vec{x} \in \overbrace{\mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_n}^{\text{feature space}} \Rightarrow \vec{x} = (c_1, c_2, \ldots, c_n)^T$$

      - **Vectors of real numbers**: $\vec{x} \in \mathcal{R}^n, \vec{x} = (2.6, 10.4, \ldots, 65.5)^T$
      - **Categorical data**: $\vec{x} = (\text{blue}, \ \text{big}, \ \text{spherical})^T$

- Hence the ML task becomes into working out a function f as follows:

$$f : C_1 \times C_2 \times \cdots \times C_n \to L$$
$$\vec{x} = (c_1, c_2, \ldots, c_n) \to l_x$$

$$f : Z = [0, 255] \times \mathcal{R} \to \{\text{salmon}, \text{sea bass}\}$$
$$(40, 20.3) \to \text{salmon}$$
$$(80, 40.7) \to \text{sea bass}$$

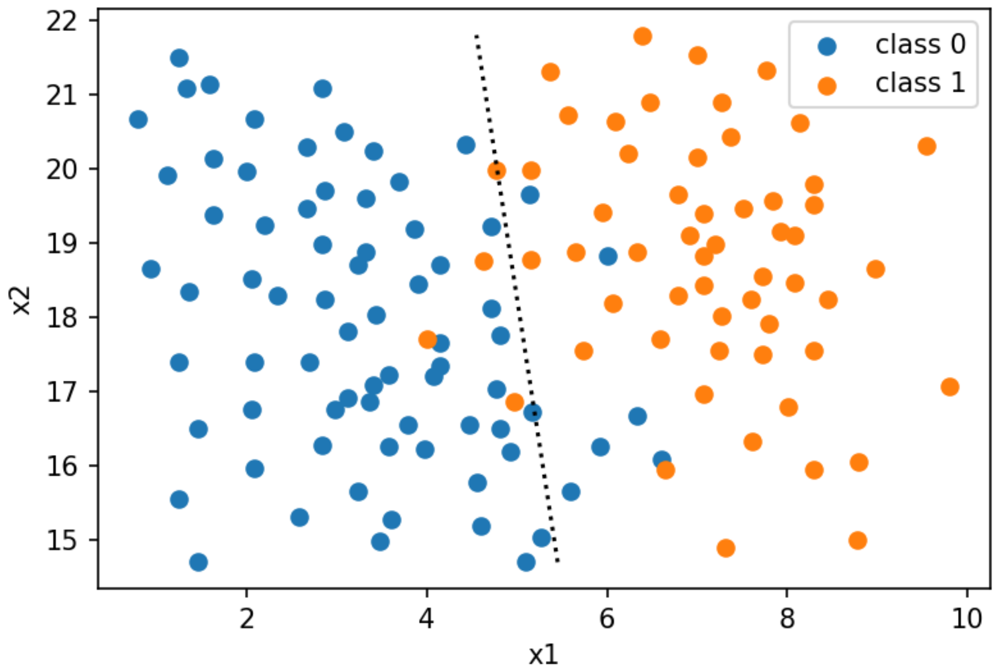
  - If we know f, we also have the separator between classes, which in turn defines the **decision rule**: single value, straight line, generic curve, surface, hyper-surface

- Solving the fish example:

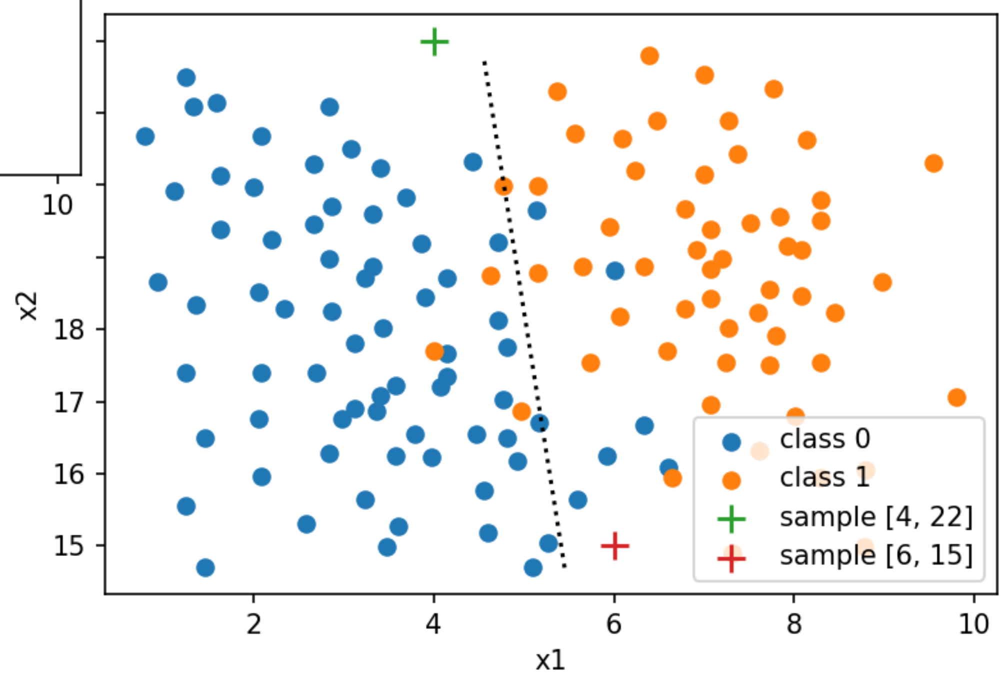
```
import numpy as np
from sklearn import svm
import matplotlib.pyplot as plt

data = np.loadtxt('fish.txt')
X = data[:,1:-1]
y = data[:,-1]

# indices of classes
i0 = np.where(y == 0)[0]
i1 = np.where(y == 1)[0]

# class samples
X0 = X[i0,:]
y0 = y[i0]
X1 = X[i1,:]
y1 = y[i1]

# number of samples for each class
print('number of samples class 0: ',
      X0.shape[0], y0.shape[0])
print('number of samples class 1: ',
      X1.shape[0], y1.shape[0])
```

```
# ML model: straight line
clf = svm.LinearSVC(fit_intercept=True, random_state=0)
clf.fit(X, y) # training

# get the model parameters
w = clf.coef_[0]
a, b = w[0], w[1]
c = clf.intercept_[0]
print('a = %.3f, b = %.3f, c = %.3f' % (a, b, c))

# plotting
plt.figure()
# plot samples
plt.scatter(X0[:,0],X0[:,1],label='class 0')
plt.scatter(X1[:,0],X1[:,1],label='class 1')
# plot model
yy = np.linspace(X[:,1].min(),X[:,1].max(),100)
plt.plot(-b/a * yy - c/a, yy, 'k:')
plt.legend()
plt.show()

# make two predictions
p = [4, 22]
l = clf.predict([p])[0]
print('predicted label for p = {} is {}'.format(p, l))

q = [6, 15]
l = clf.predict([q])[0]
print('predicted label for q = {} is {}'.format(q, l))
```

- Solving the fish example:




```
number of samples class 0:  74 74
number of samples class 1:  57 57
a = 0.455, b = 0.058, c = -3.328
predicted label for p = [4, 22] is 0.0
predicted label for q = [6, 15] is 1.0
```



Basic concepts

- Hence the ML task becomes into working out a function f as follows:

$$f : C_1 \times C_2 \times \cdots \times C_n \to L \qquad f : Z = [0, 255] \times \mathcal{R} \to \{\text{salmon}, \text{sea bass}\}$$

$$\vec{x} = (c_1, c_2, \ldots, c_n) \to l_x \qquad (40, 20.3) \to \text{salmon}$$

$$(80, 40.7) \to \text{sea bass}$$

  - If we know f, we also have the separator between classes, which in turn defines the **decision rule**: single value, straight line, generic curve, surface, hyper-surface

- Once f has been determined, the model has to be **evaluated**:

CONFUSION MATRIX	real positives	real negatives
positive predictions	TP	FP
negative predictions	FN	TN

$$A = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad \text{(accuracy)}$$

Basic concepts

- RECAPITULATION: To solve a classification problem, it is important
  - to know how to **generate features** and **choose the most appropriate** ones
  - know **as many classification techniques as posible**, as well as their **strengths** and **weaknesses**
- Some machine learning models:
  - Bayesian classifiers
  - Neural networks
  - Decision trees, etc.
- Although humans are able to move quickly, smoothly and without apparent effort from one classification task to another ...
  … designing a **universal classifier** (capable of performing accurately in a wide variety of tasks) is yet an **unsolved problem**
  - each decision task may require different features and thus result into different decision rules with different effectiveness levels
  - each technique is suitable for one type of problem

Contents

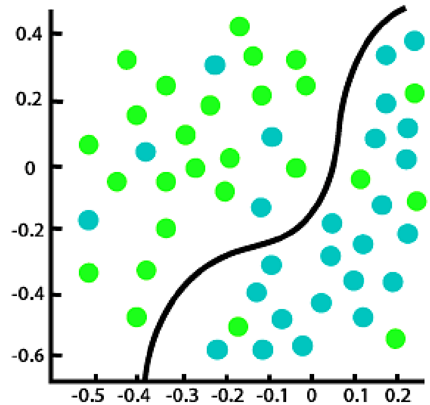
- Machine learning in the context of Artificial Intelligence
- Description of the problem and basic concepts
- Regression tasks
- ML design cycle
- Exploitation (maybe as part of a perception system)
- Flavours of machine learning
- Development framework (suggested)

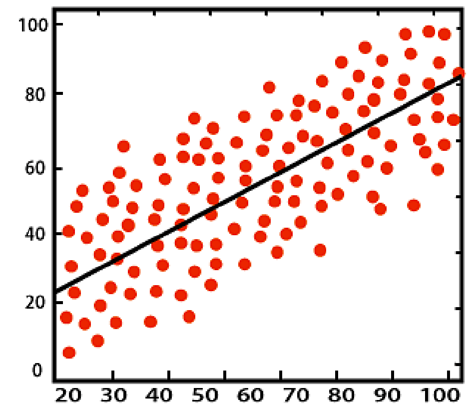
Regression tasks

- **<u>Goal</u>.** Find some **functional description of data**, often for **predicting** values for new inputs

Classification *versus* Regression

x_1	x_2	x_3	x_4	y = class
1	2	-4	3	1
3	2	2	2	1
2	5	3	2	2
2	4	2	3	2
1	1	0	2	1
6	2	4	1	2





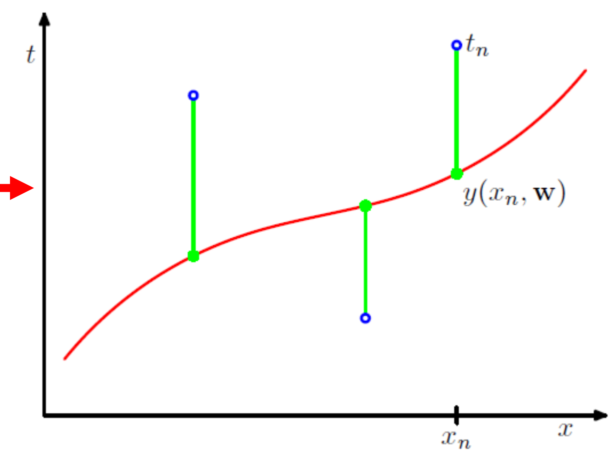
Straight line fitting: *given* (x_i, y_i) *and*

$$y = ax + b$$

$$a, b\ ?$$

(in general, **curve fitting**)

- Data involved:
  - X = **N** samples
  - y = **expected values** instead of class labels
- Learning based on the **approximation error**

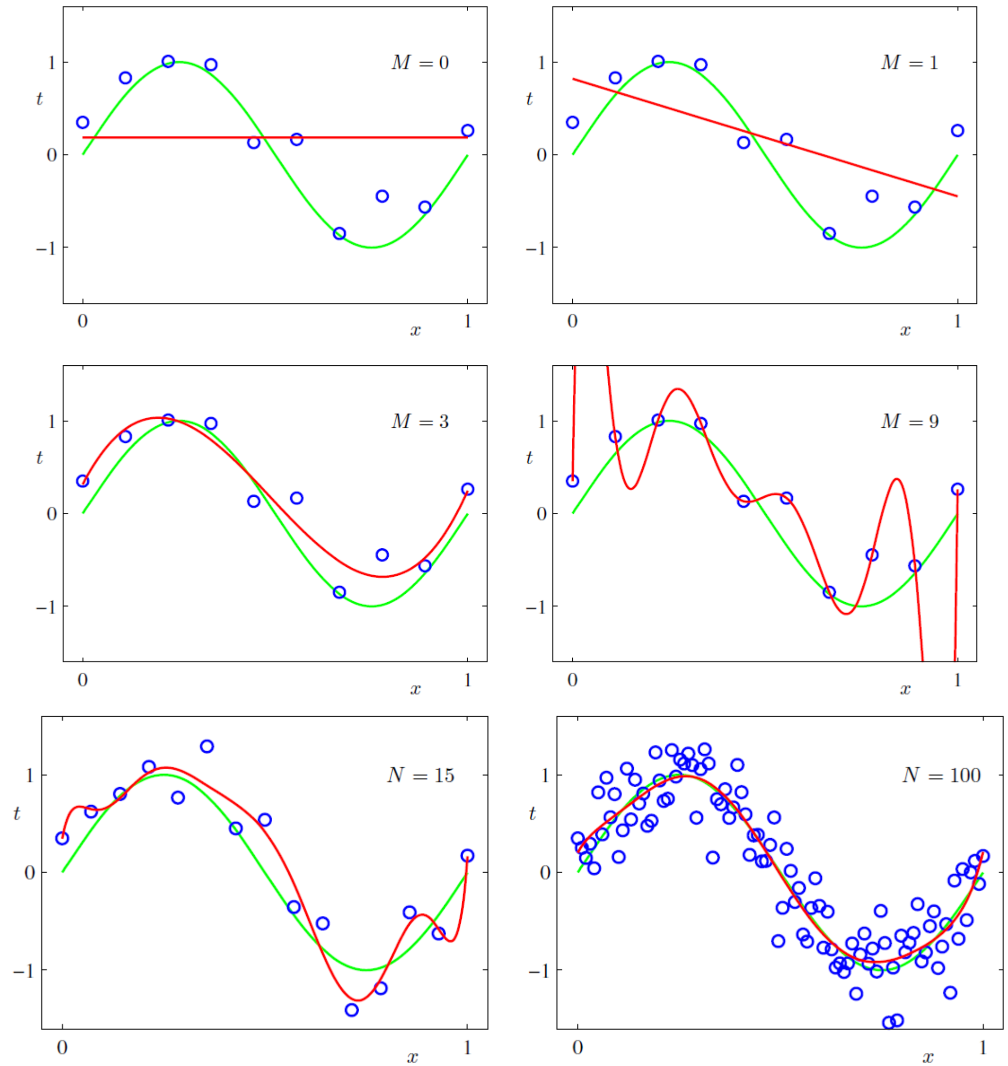


- Meaningful examples:
  - Weather prediction
  - Stock market price prediction
  - Economical/Market trends capture and forecasting
  - etc.

Regression tasks

- **<u>Example</u>. M-degree** polynomial curve fitting: $y = a_M x^M + a_{M-1} x^{M-1} + \cdots + a_1 x + \ a_0$

  – Data involved:
    - X = **N** samples
    - y = expected values (continuous variable)

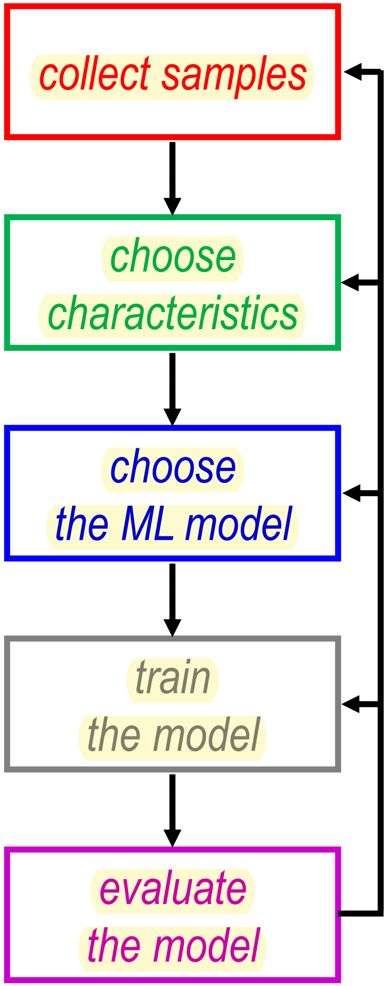


Contents

- Machine learning in the context of Artificial Intelligence
- Description of the problem and basic concepts
- Regression tasks
- ML design cycle
- Exploitation (maybe as part of a perception system)
- Flavours of machine learning
- Development framework (suggested)

The design cycle

- To design an ML / perception system one typically has to:



- can be a large part of the cost
- preliminary study with few samples but many more later
- how do you know if we have all the necessary samples?

- characteristics that separate classes well, invariant to irrelevant transformations, etc.
- useful prior knowledge: typical attributes, shape of classes, ...

- linear/no, single/ensemble, neural network / SVM / decision tree ...
- predict classifier behaviour, performance, complexity, etc.

- choose samples for training **(training set)**
- find the parameters of the classes if needed, e.g. distribution
- determine the model parameters

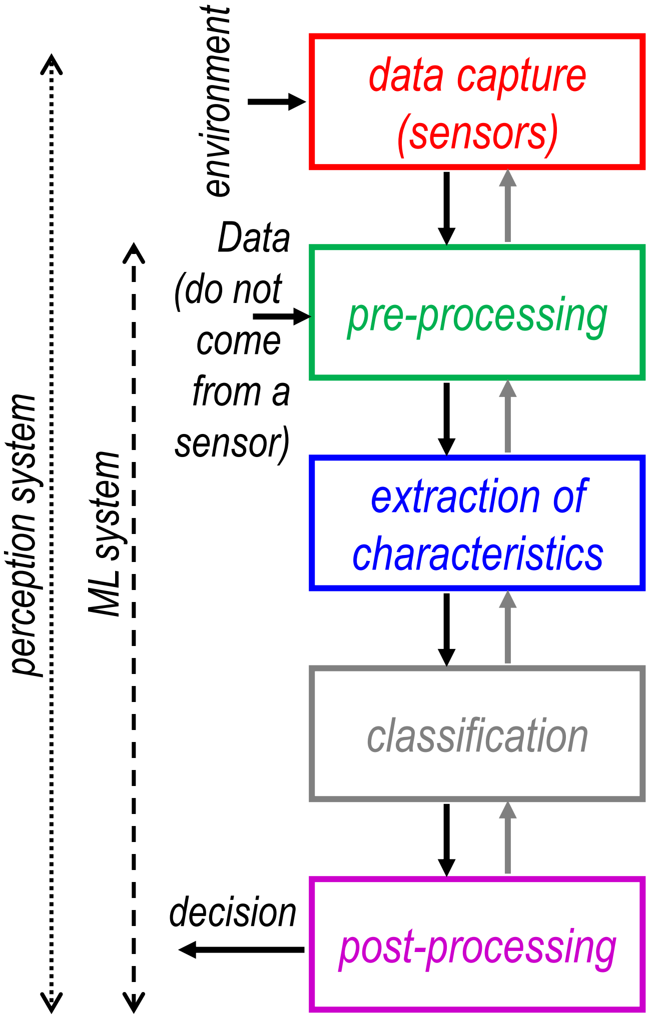
- choose the samples for testing (**test set**)
- detect **overfitting**, and other misbehaviours
- evaluate the classif. complexity and scalability (dimensions/classes)

Contents

- Machine learning in the context of Artificial Intelligence
- Description of the problem and basic concepts
- Regression tasks
- ML design cycle
- Exploitation (maybe as part of a perception system)
- Flavours of machine learning
- Development framework (suggested)

ML and perception systems

- Perception systems typically adheres to the following structure:



- difficulty = sensor characteristics and limitations: bandwidth, resolution, sensitivity, distortion, signal-to-noise ratio,...

- isolate structures in the data: e.g. isolate objects in an image, isolate phonemes/words in sound, …

- ideal extractor versus omnipotent classifier
- characteristics invariant to rotations, translations, scale, pronunciation speed/amplitude, ...
- detect anomalies in the data: missing values, outliers, …

- works with abstract entities
- typically, independent of the application domain
- difficulty = accurate predictions despite classes variability

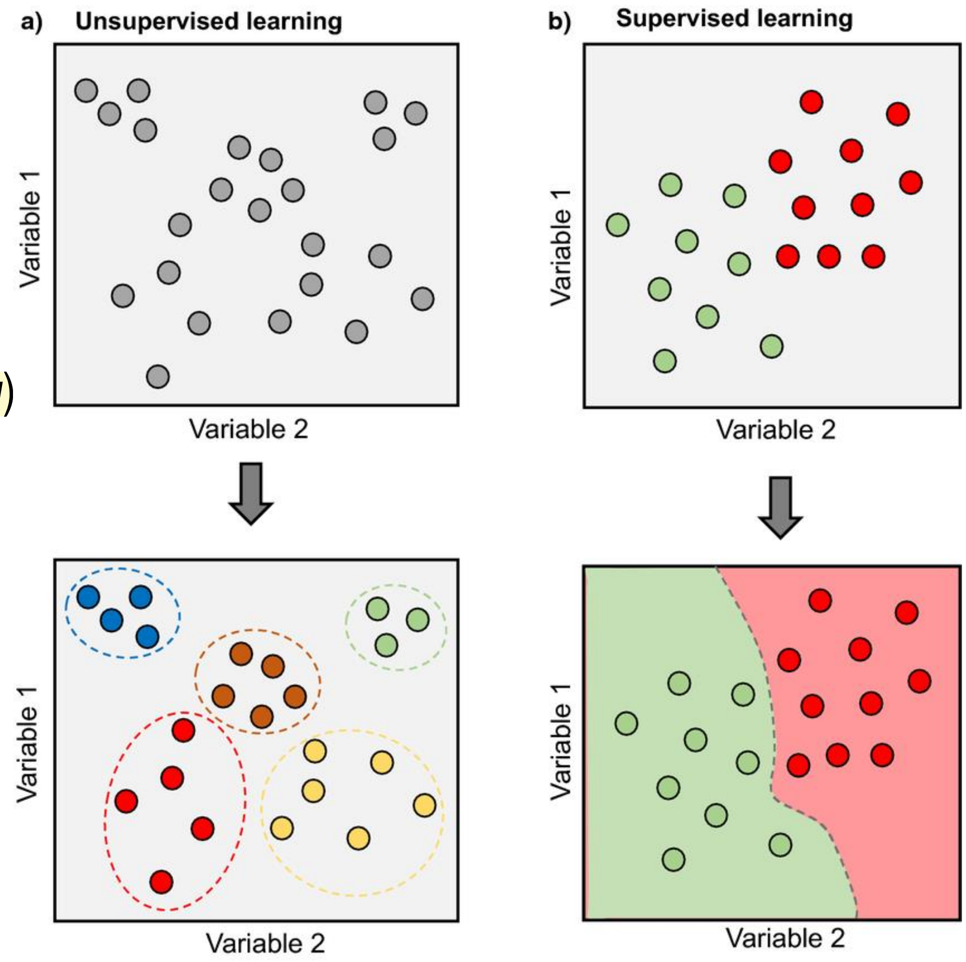
- exploit context information to improve classification
  - e.g. T-E C-T in English would be completed as THE CAT
- combine classifiers: acoustic recognition + lip reading

Contents

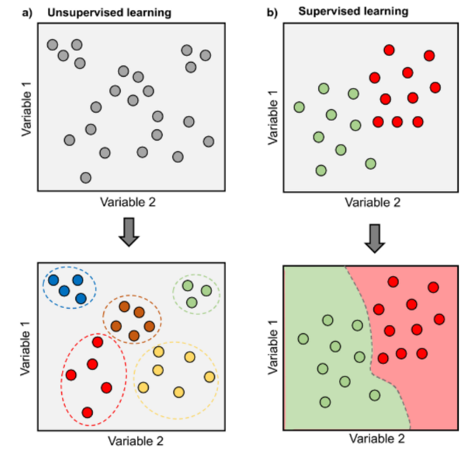
- Machine learning in the context of Artificial Intelligence
- Description of the problem and basic concepts
- Regression tasks
- ML design cycle
- Exploitation (maybe as part of a perception system)
- Flavours of machine learning
- Development framework (suggested)

Flavours of machine learning

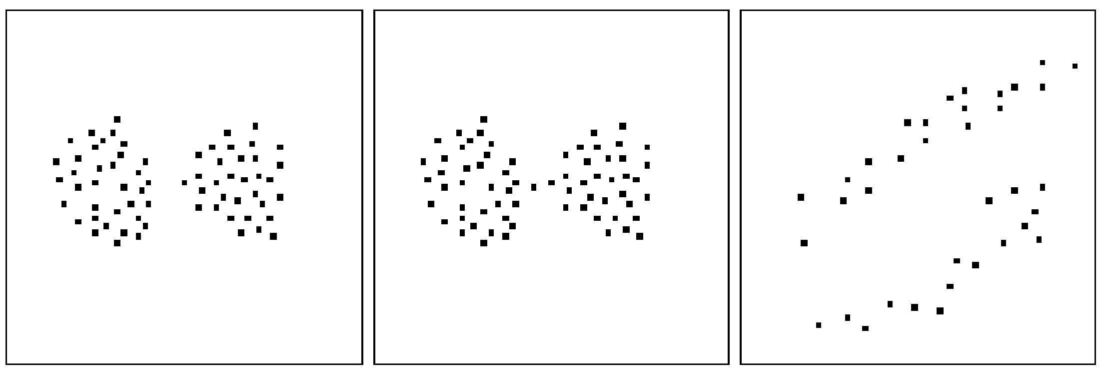
- A distinction is made between several types of learning:
  - supervised learning
    - an expert labels each sample of the dataset
  - unsupervised learning
    - there is no explicit expert
    - grouping techniques (*clustering*)



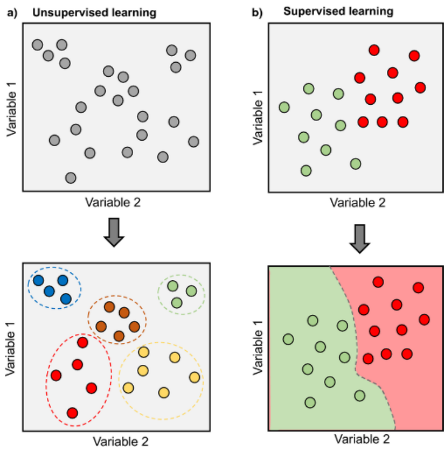
Flavours of machine learning



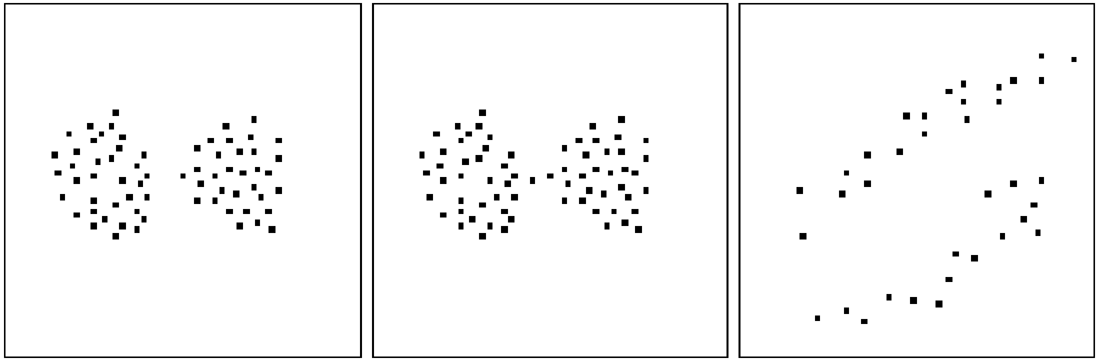
- A distinction is made between several types of learning:
  - **supervised learning**
    - an expert tags each sample of the dataset
  - **unsupervised learning**
    - there is no explicit expert, grouping techniques (*clustering*)
    - ideally, the system looks for the *natural structure* of the data



Flavours of machine learning



- A distinction is made between several types of learning:
  - **supervised learning**
    - an expert tags each sample of the dataset
  - **unsupervised learning**
    - there is no explicit expert, grouping techniques (*clustering*)
    - ideally, the system looks for the *natural structure* of the data



  - **reinforcement learning** or **learning with a critic**
    - the system learns how to make decisions by exploring the problem through a set of trials/interactions with the environment which lead to positive or negative rewards

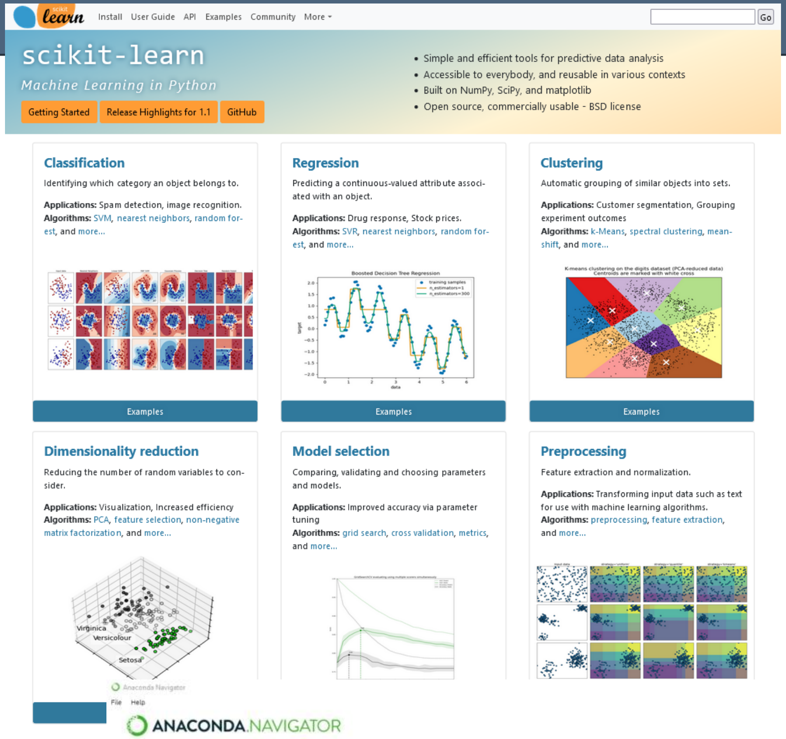
Contents

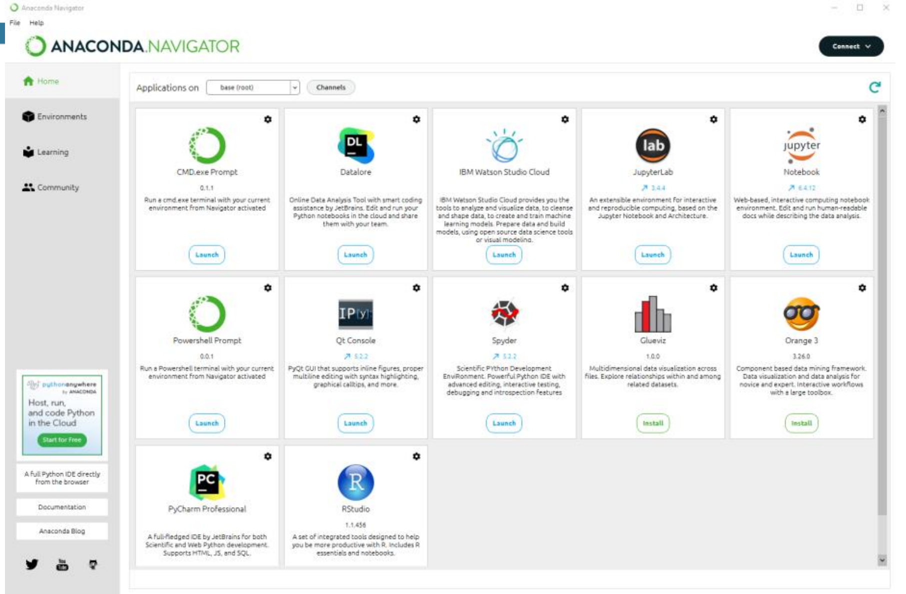
(suggested) Development framework

- Python 3.x (www.python.org)

- Numpy (numpy.org)
- Scikit-learn (scikit-learn.org)
- Pandas (pandas.pydata.org)
- Matplotlib (matplotlib.org)
- Other libraries as needed

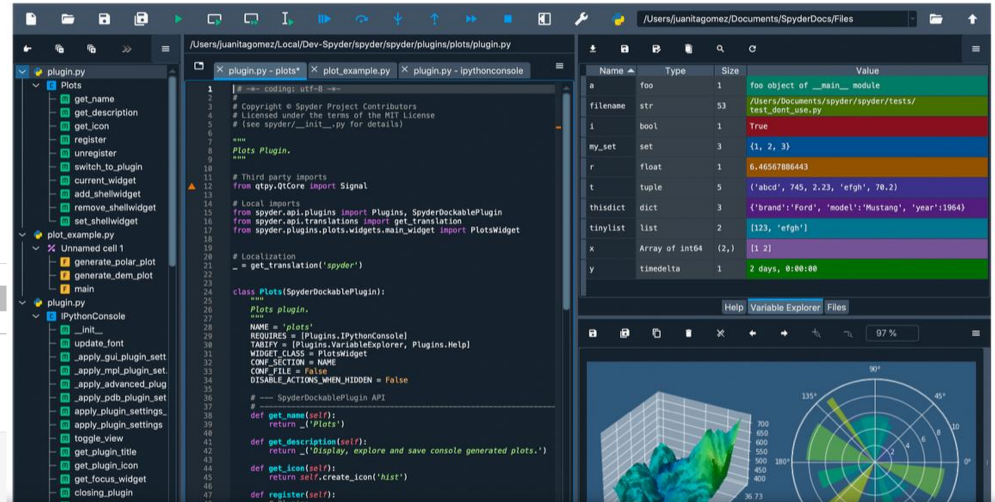
- Anaconda (www.anaconda.com)

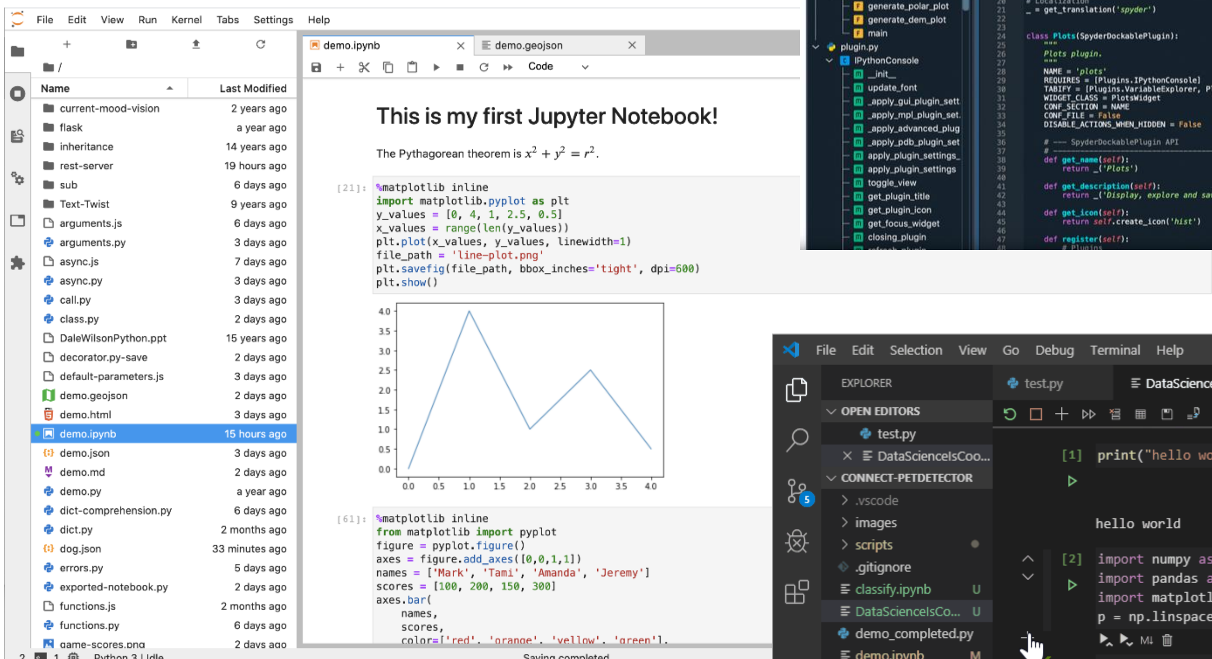




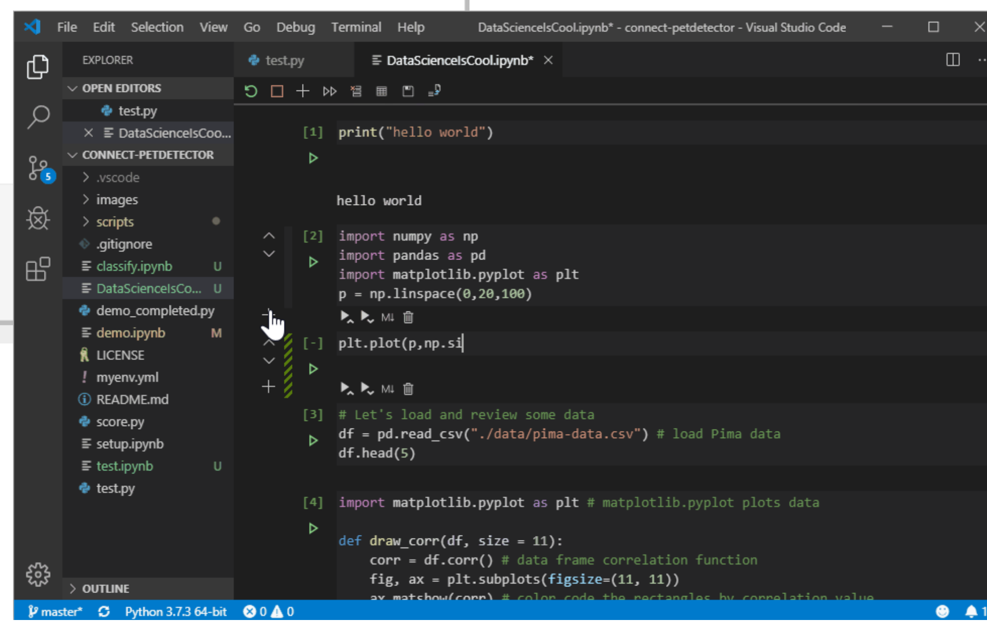
(suggested) Development framework

- Spyder IDE (www.spyder-ide.org)
- JupyterLab (from inside Anaconda)





- Visual Studio IDE (code.visualstudio.com)



Lecture 1: Introduction

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ