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

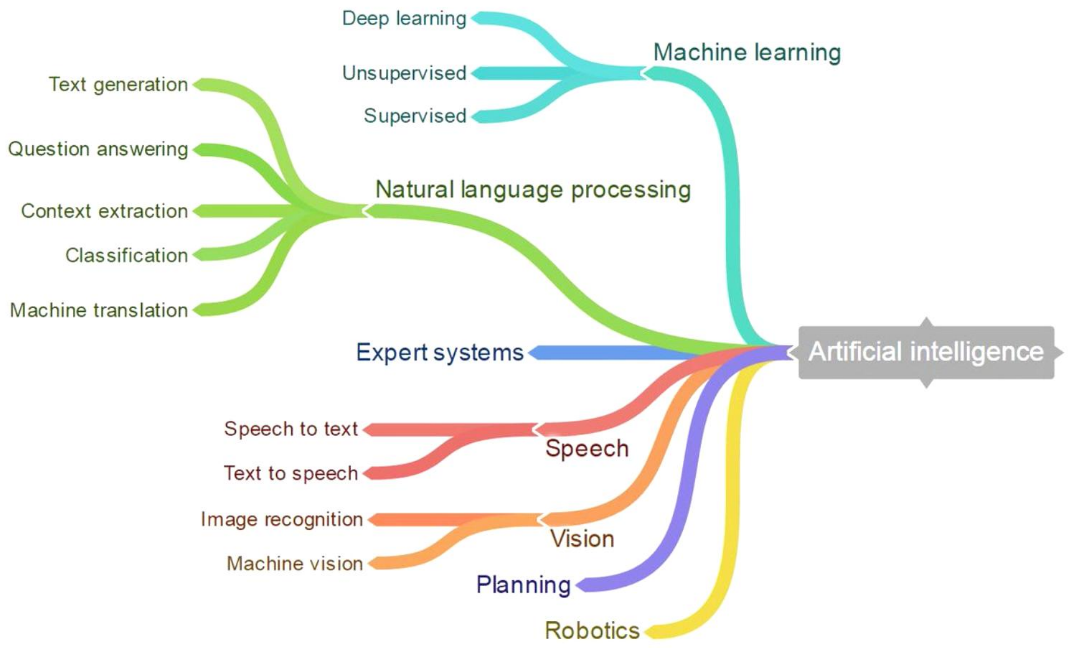
Contents

Machine learning in AI

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

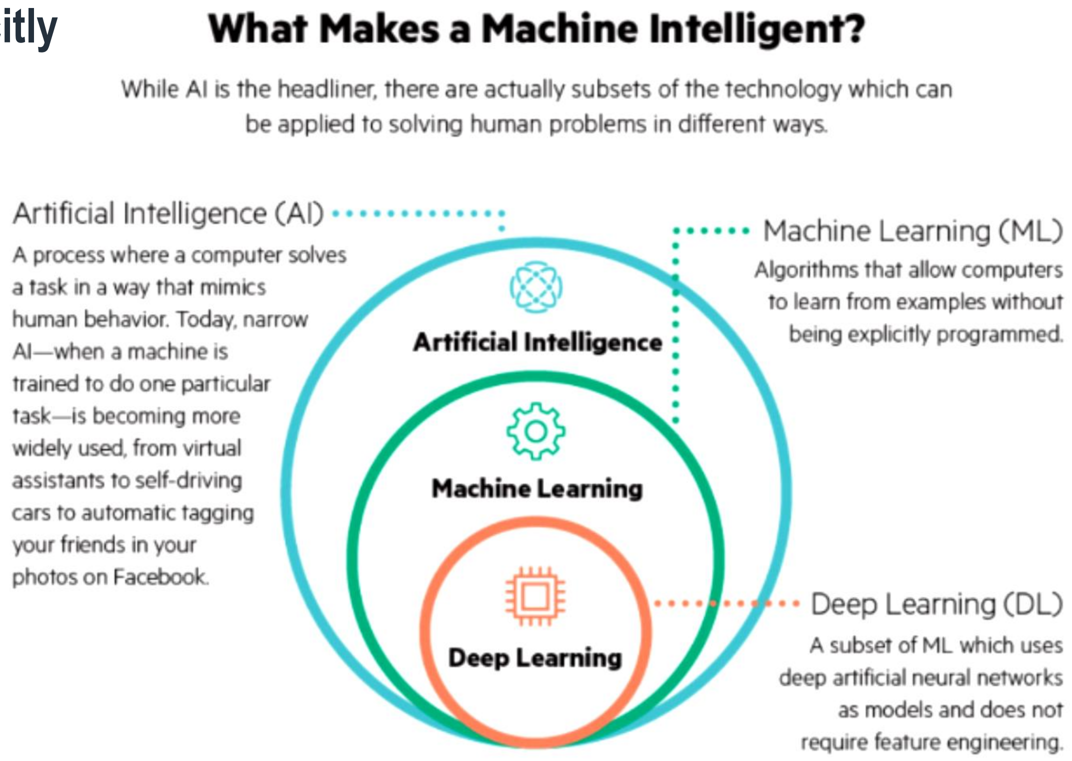
    A branch of computer science dealing with the simulation of intelligent behaviour in computers
  - AI as a **property of a machine**:

    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**


What Makes a Machine Intelligent?
While AI is the headliner, there are actually subsets of the technology which can be applied to solving human problems in different ways.
Artificial Intelligence (AI)
A process where a computer solves a task in a way that mimics human behavior. Today, narrow AI—when a machine is trained to do one particular task—is becoming more widely used, from virtual assistants to self-driving cars to automatic tagging your friends in your photos on Facebook.
Machine Learning (ML)
Algorithms that allow computers to learn from examples without being explicitly programmed.
Deep Learning (DL)
A subset of ML which uses deep artificial neural networks as models and does not require feature engineering.
Artificial Intelligence
Machine Learning
Deep Learning

Machine learning in AI

- A formalization of **Machine Learning** (from Tom Mitchell, CMU 1998)
  - Machine Learning is the study of algorithms that
    - improve their performance P
    - at some task T
    - with experience E
  - A well-defined learning task is given by $< T, E, P >$

- Some examples:

T: **Recognizing hand-written words**
E: Database of human-labeled images of handwritten words
P: Percentage of words correctly classified

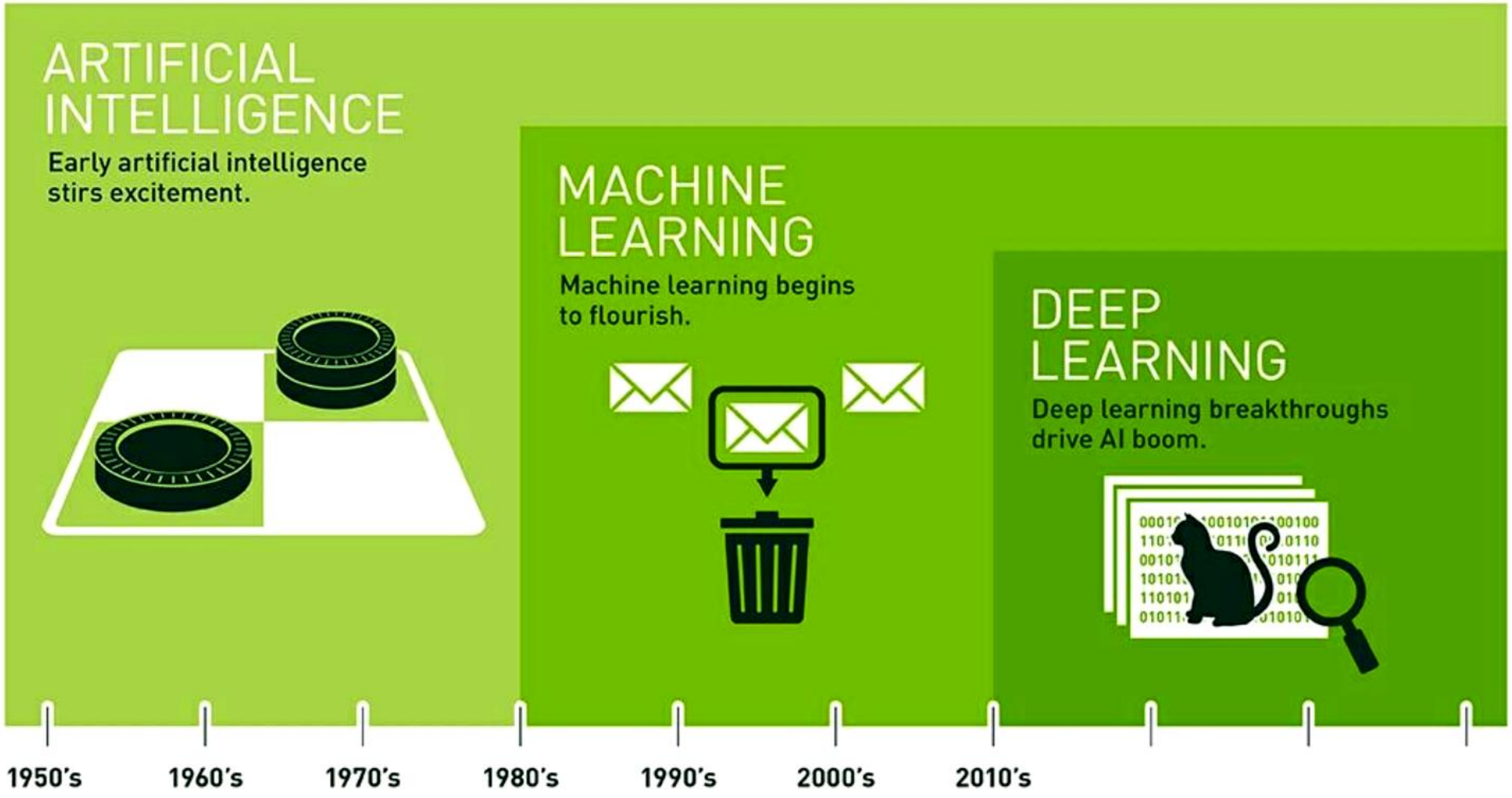
T: **Playing checkers**
E: Record of practice games
P: Percentage of games won against an arbitrary opponent

T: **Driving on four-lane highways using vision sensors**
E: A sequence of images and steering commands recorded while observing a human driver
P: Average distance traveled before a human-judged error

T: **Categorize email messages as spam or legitimate**
E: Database of emails, with human-given labels
P: Percentage of email messages correctly classified

Machine learning in AI

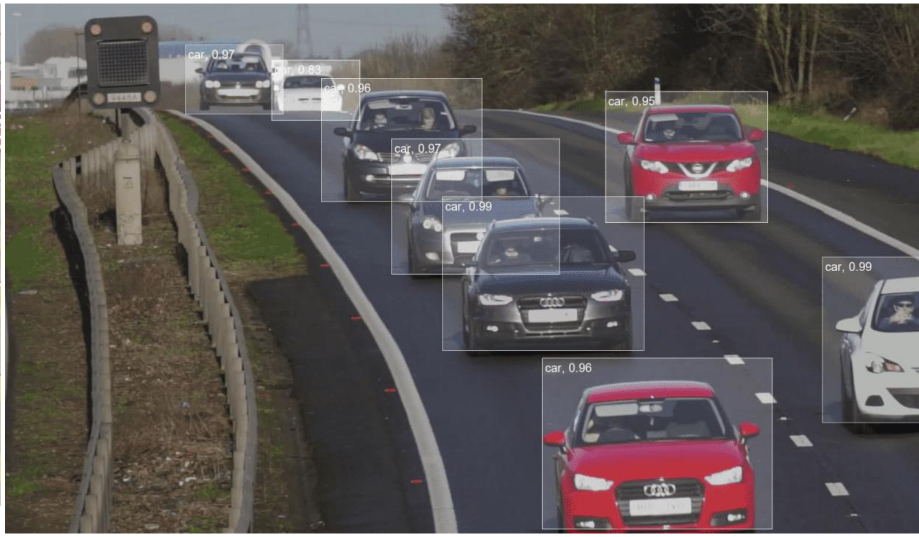
- **Machine learning**, a brief chronology

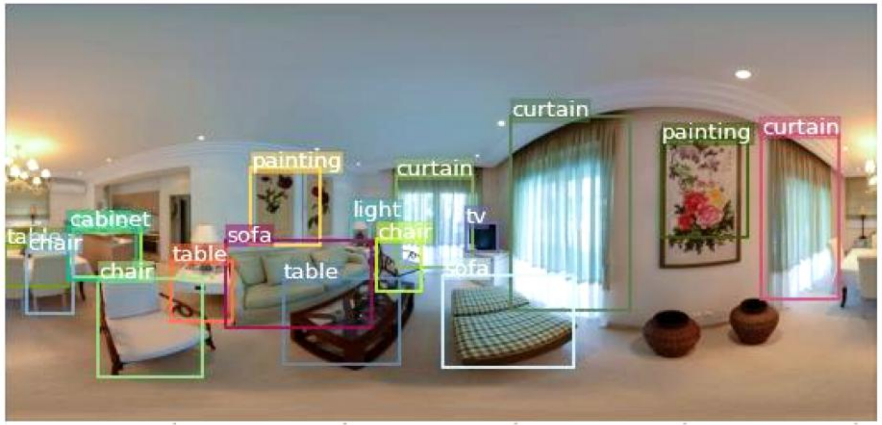

ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

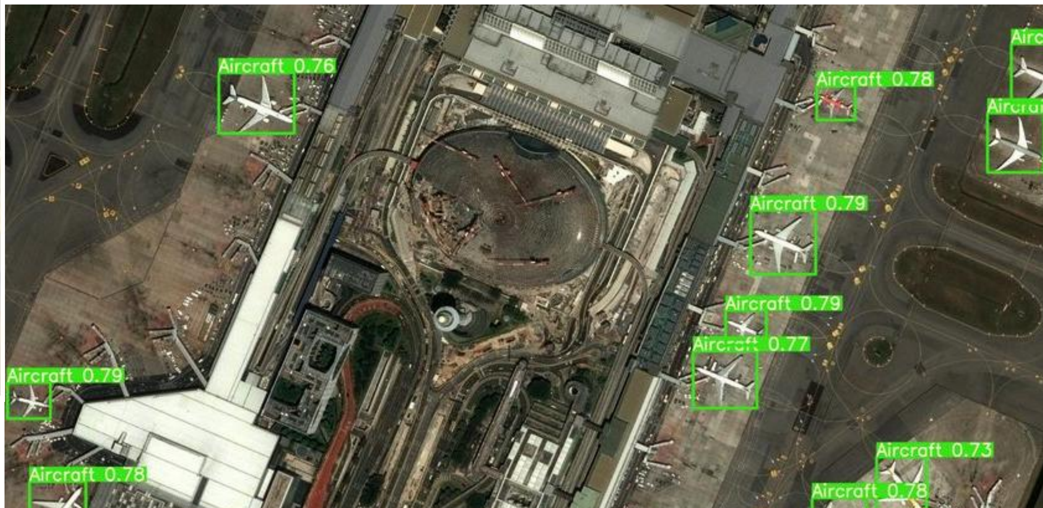
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Machine learning in AI

- Nowadays, we are experiencing the **AI boom**. Why now?
  - We are now capable of addressing and solving hard problems



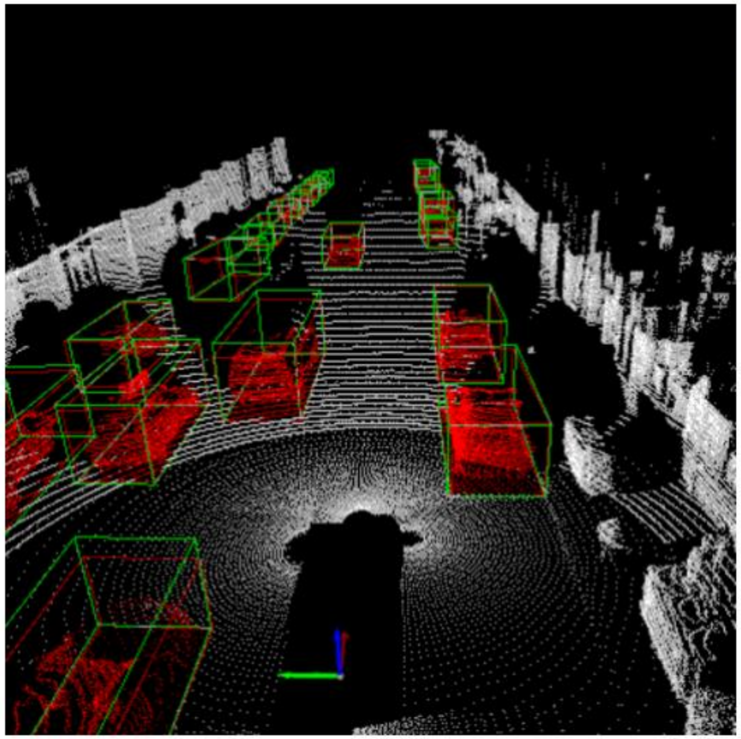


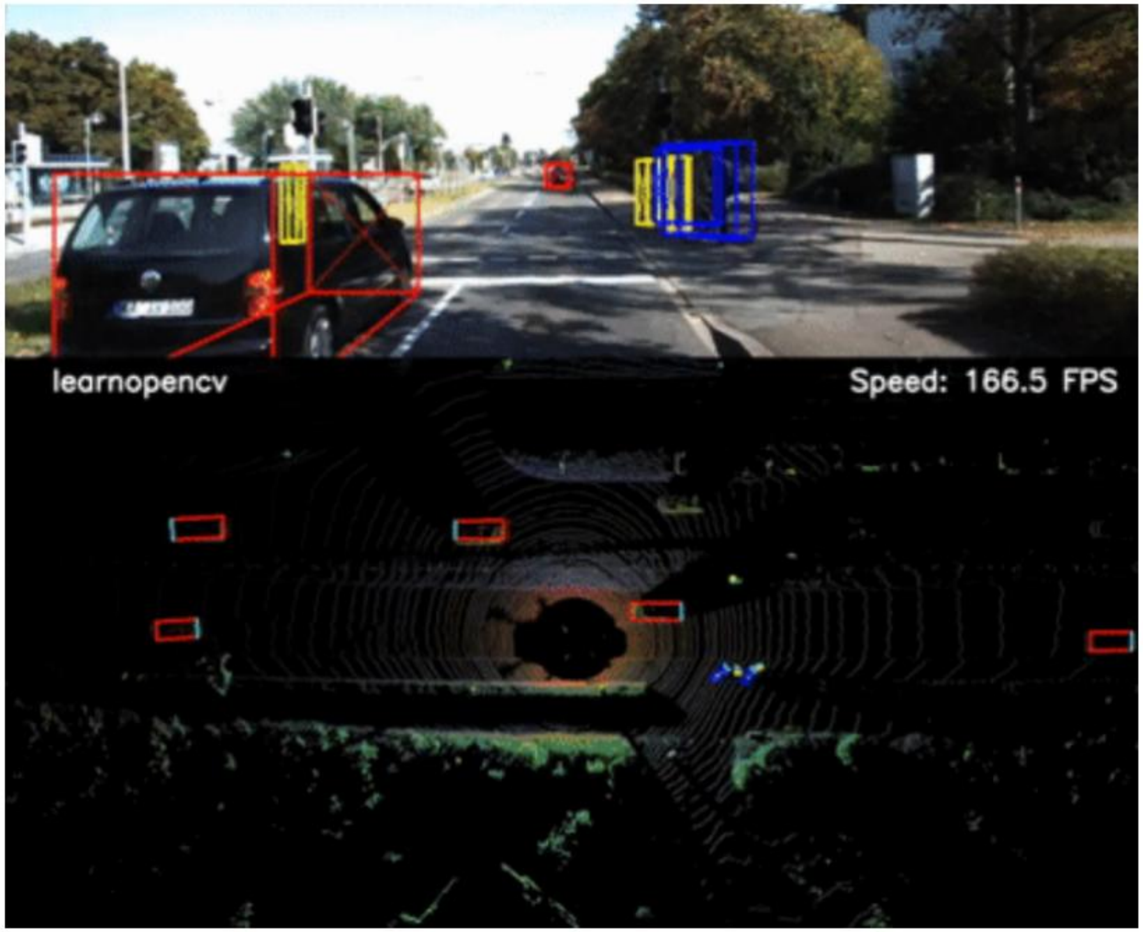


Alberto Ortiz (updated 23/09/2025)

Machine learning in AI

- Nowadays, we are experiencing the **AI boom**. Why now?
  - We are now capable of addressing and solving hard problems





Machine learning in AI

- Nowadays, we are experiencing the **AI boom**. 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

python

Keras

TensorFlow

PyTorch

Windows Azure
MACHINE LEARNING

IBM
Cognitive computing
IBM Watson
IBM.com/developerWorks

Contents

Description of the problem

- A simple example:

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

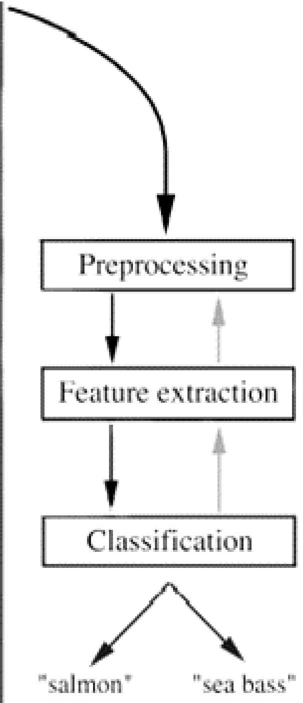
  – Discriminate between **salmon** and **sea bass**

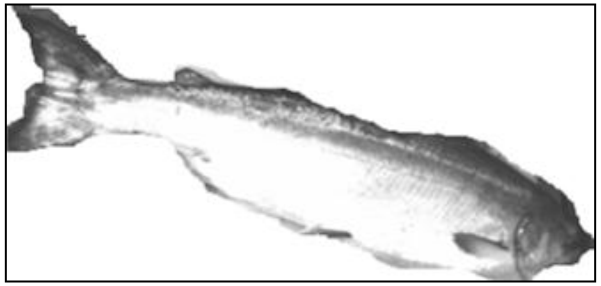
  – The image is **pre-processed** as much as needed, e.g. isolate the fish instances that appear in the image (segmentation)

  – 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)

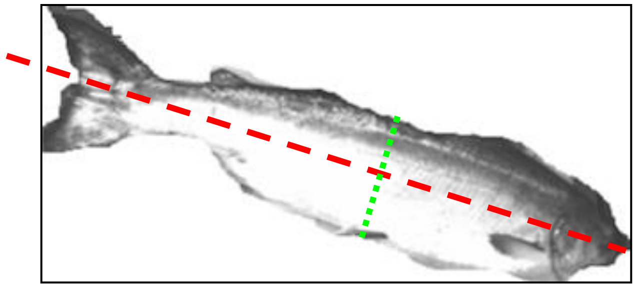




pre-procesing

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)

$\left.\begin{array}{l}\text{Non-uniform reflectance}\\ \text{Non-uniform illumination}\\ \text{Shadows, specularities (glossiness)}\end{array}\right\}$ 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:

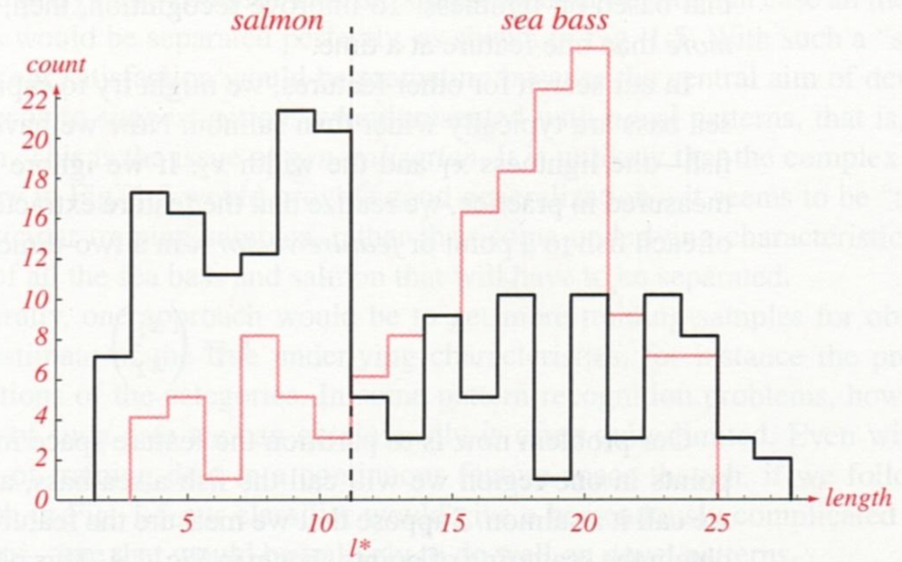


FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked *l** will lead to the smallest number of errors, on average.

  - 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

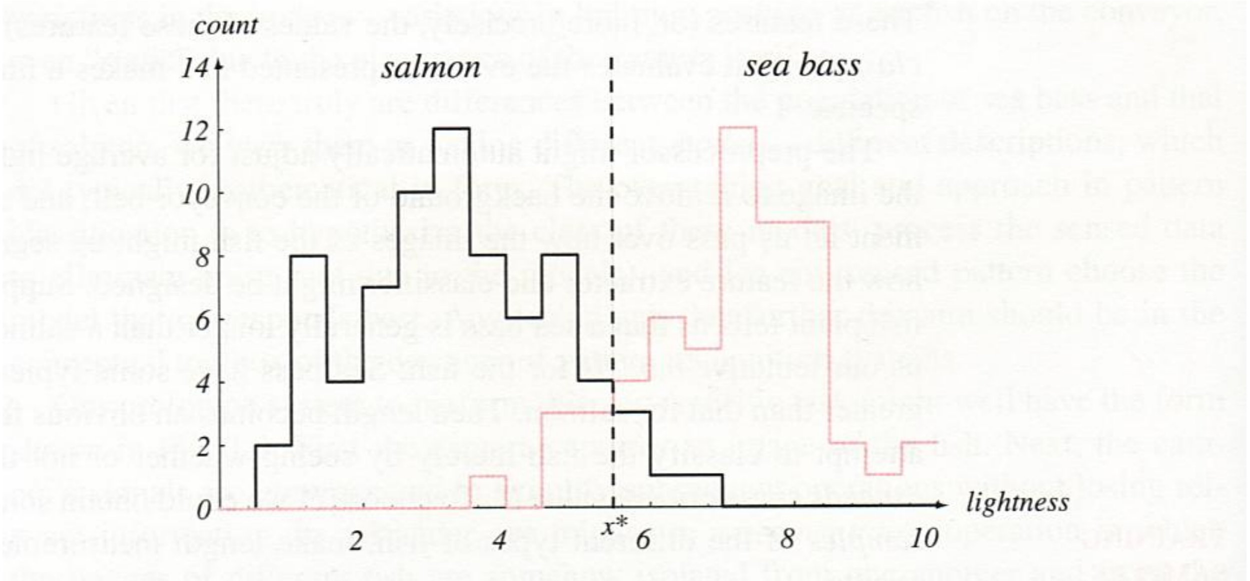


FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors,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

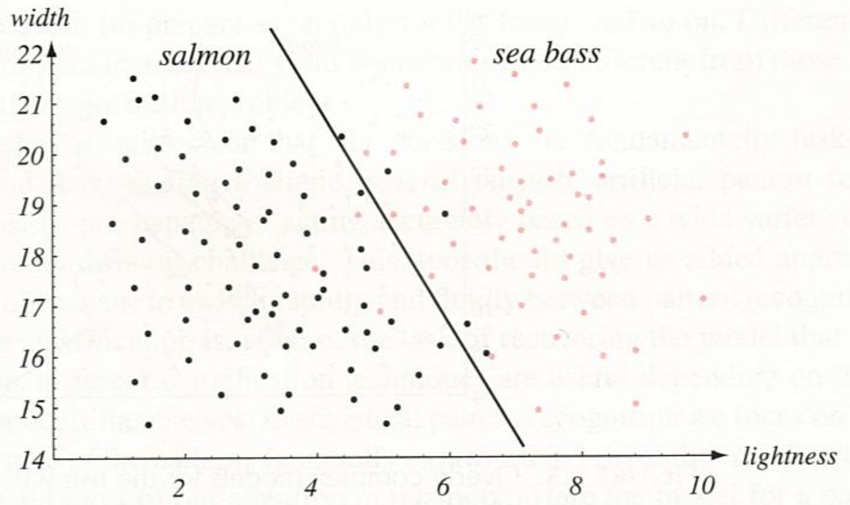


FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors.

Description of the problem

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

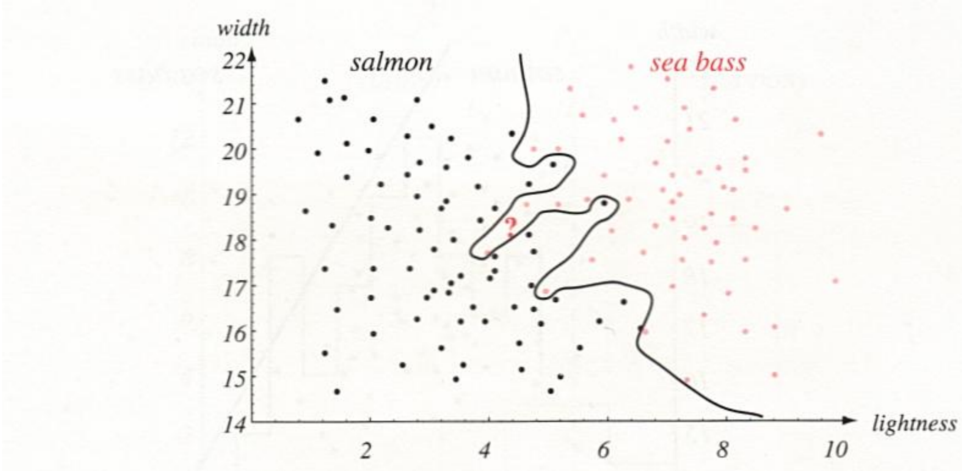


FIGURE 1.5. Overly complex models for the fish will lead to decision boundaries that are complicated. While such a decision may lead to perfect classification of our training samples, it would lead to poor performance on future patterns. The novel test point marked ? is evidently most likely a salmon,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

Description of the problem

- 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

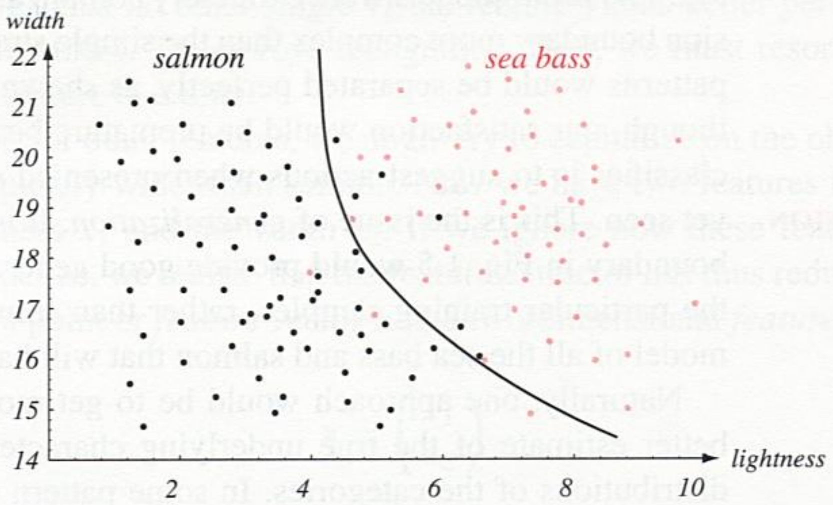


FIGURE 1.6. The decision boundary shown might represent the optimal tradeoff between performance on the training set and simplicity of classifier, thereby giving the highest accuracy on new patterns.

Description of the problem

- A simple example (cont.):
  - The key point is that the classifier is capable of dealing well with as many unseen samples as possible → 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)

        ≡ **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

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

(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

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

➋ **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**: $\vec{x} = (\text{blue, big, spherical})^T$

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 **separation curve** between classes, which in turn defines the **decision rule**: single value, straight line, generic curve, surface, hyper-surface

Basic concepts

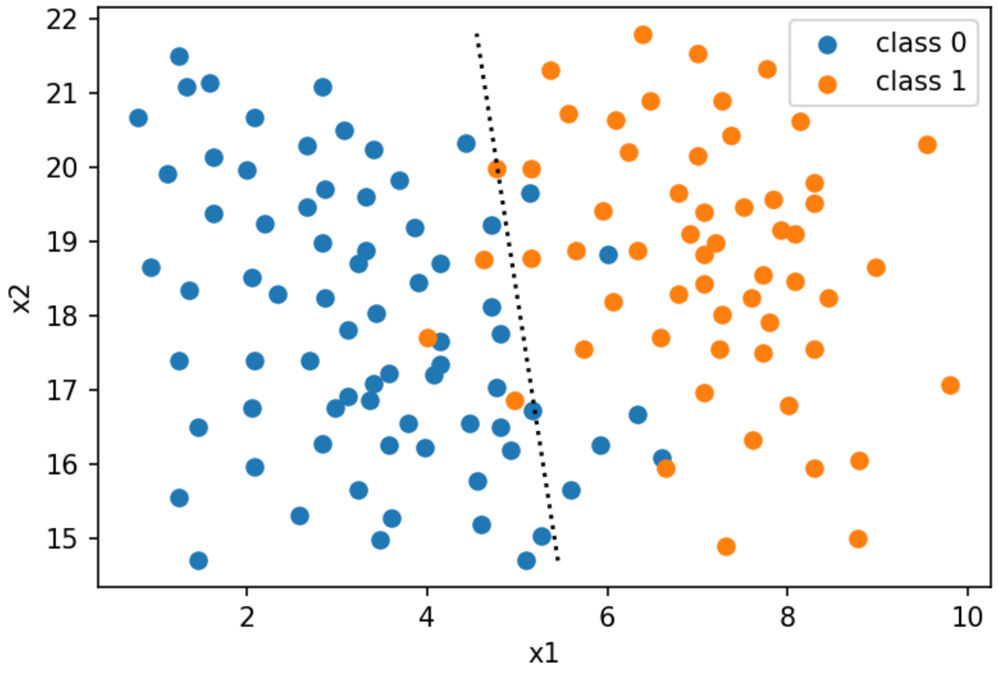
- Solving the fish example:

```
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])
```

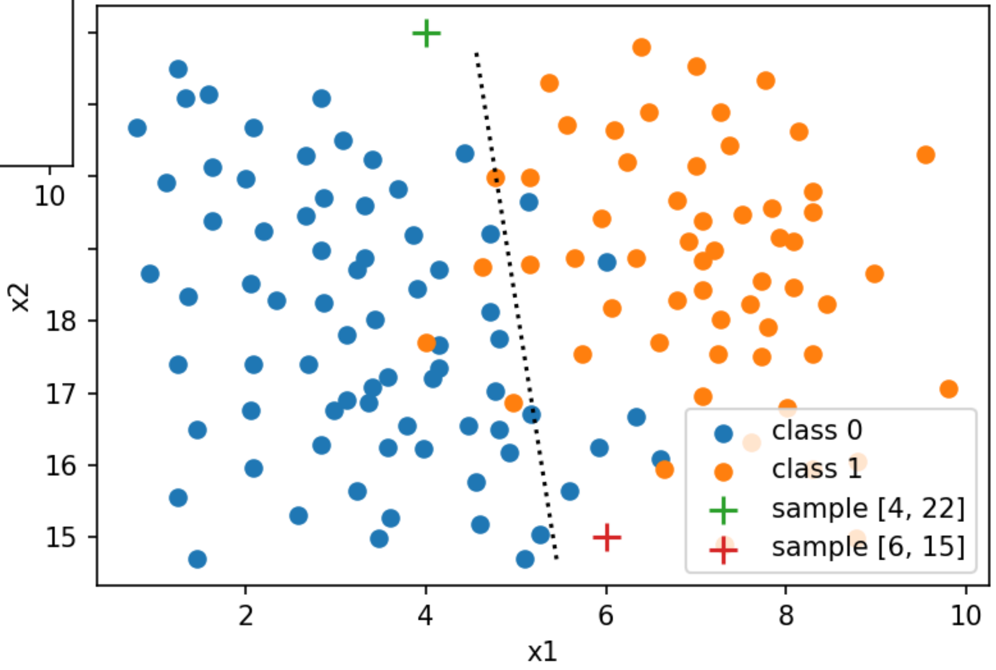
```
# ML model: straight line
clf = svm.LinearSVC(fit_intercept=True, random_state=0)
clf.fit(X, y) # training

# 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))
```

Basic concepts

- 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 **separation curve** 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

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

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

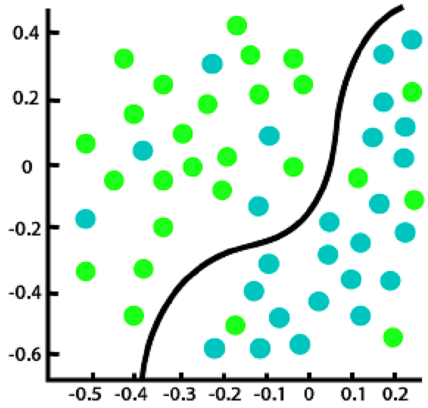
Contents

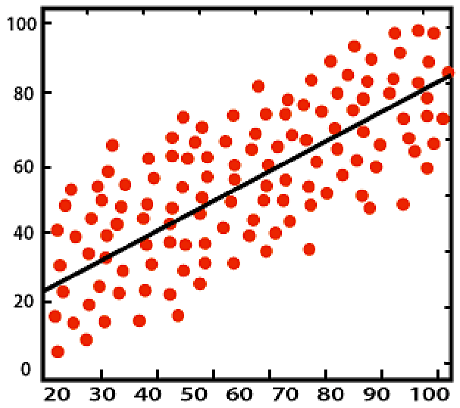
Regression tasks

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

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

Classification *versus* Regression





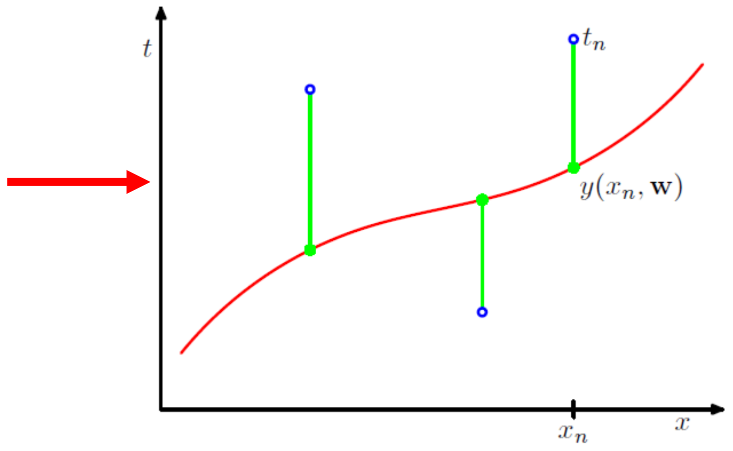
Straight line fitting: *given* (x_i, y_i) *and*

$$y = ax + b$$

$$a, b\ ?$$

(in general, **curve fitting**)

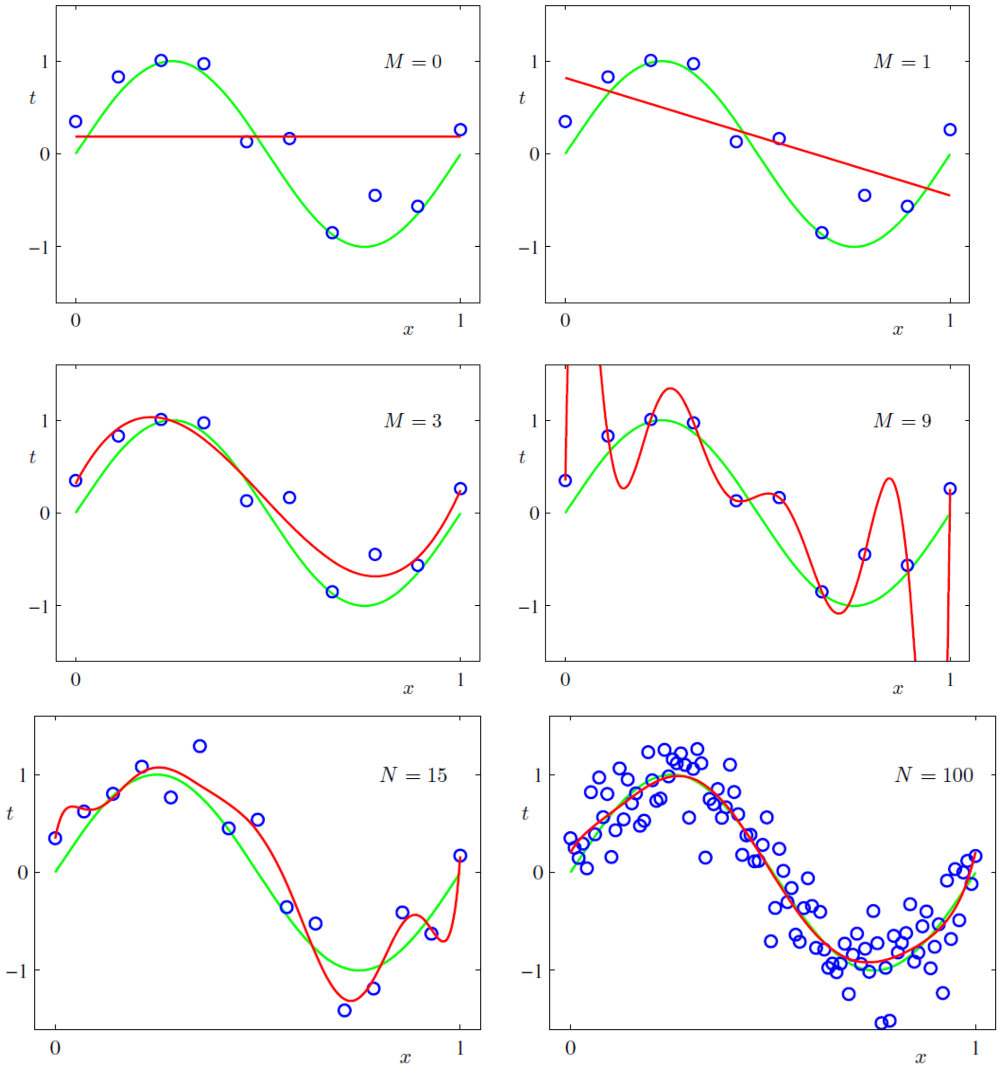
- Data involved:
  - X = $\boldsymbol{N}$ samples
  - y = **expected values** instead of class labels
- Learning based on the **approximation error**



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

Regression tasks

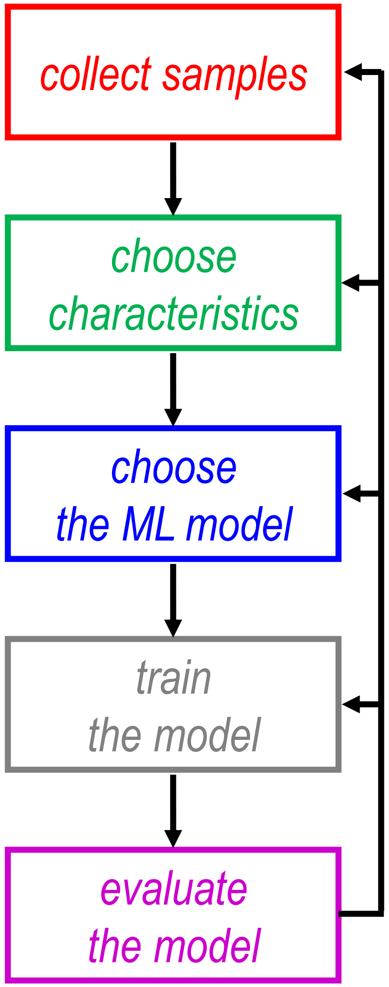
- **Example. M-degree** polynomial curve fitting: $t = a_M x^M + a_{M-1} x^{M-1} + \cdots + a_1 x + a_0$
  - Data involved:
    - X = $\boldsymbol{N}$ samples
    - y = expected values (continuous variable)



Contents

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

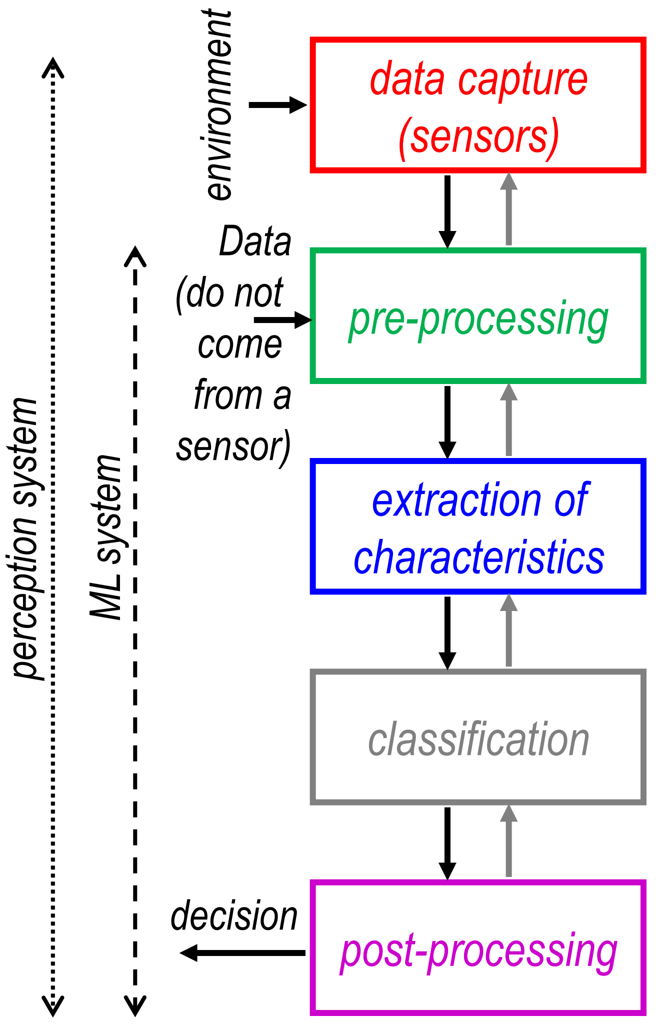
- choose the samples for testing (**test set**)
- detect **overfitting**, and other misbehaviours
- evaluate the classif. complexity and scalability (dimensions/classes)

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 and ML systems typically adhere 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, …

- calculate each feature (= properties) of the chosen descriptor
- simple or complex calculations

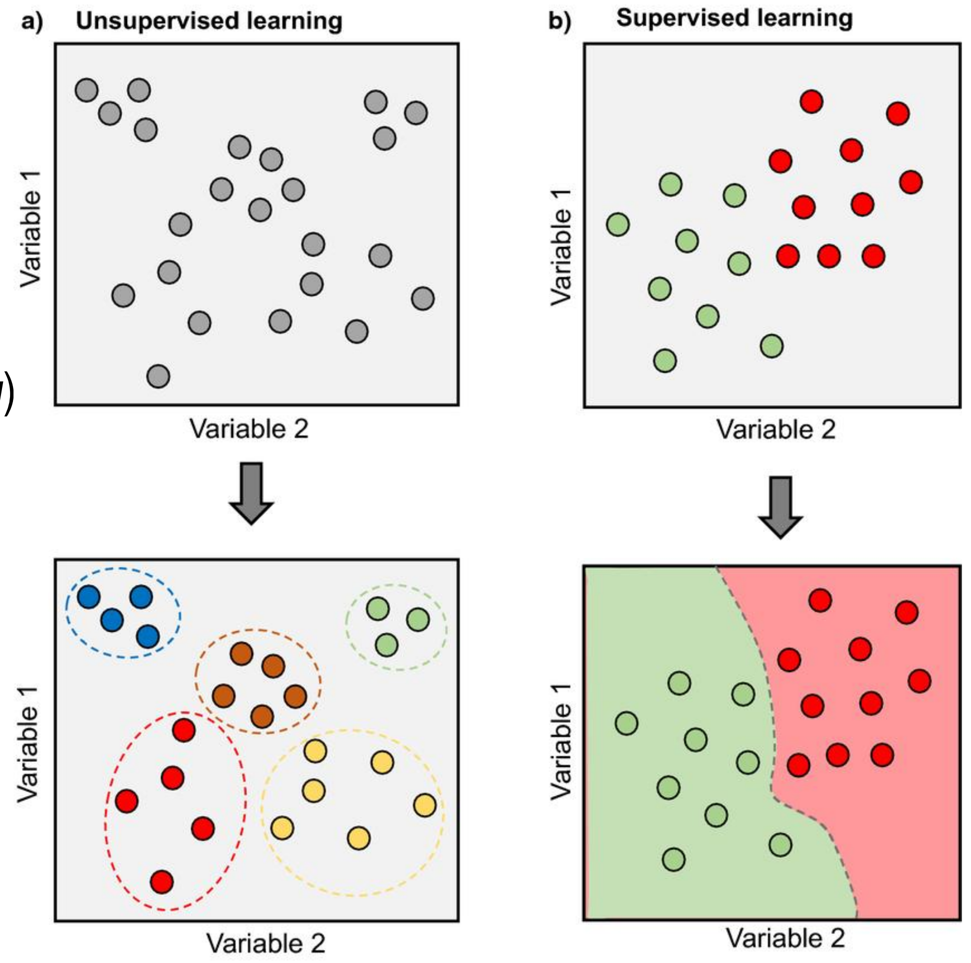
- works with abstract entities
- typically, independent of the application domain
- difficulty = accurate predictions despite classes variability

- 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

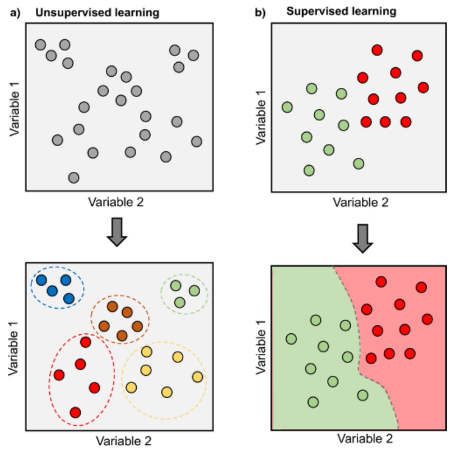
Flavours of machine learning

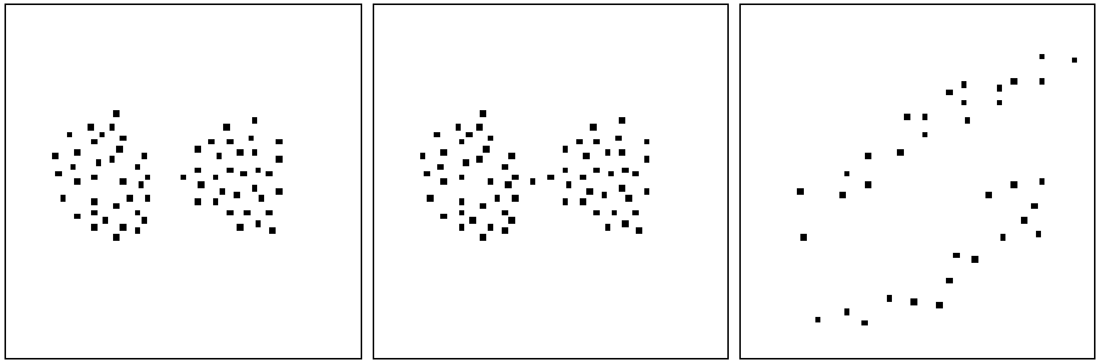
- A distinction is made between several types of learning:
  - **supervised learning**
    - an expert labels each sample of the dataset
  - **unsupervised learning**
    - there is no explicit expert
    - grouping techniques (*clustering*)



Flavours of machine learning

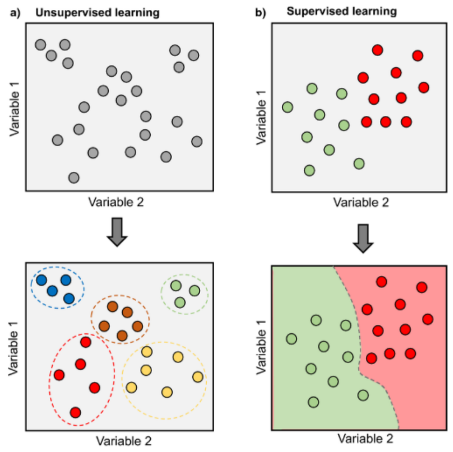
- A distinction is made between several types of learning:
  - **supervised learning**
    - an expert tags each sample of the dataset
  - **unsupervised learning**
    - there is no explicit expert, grouping techniques (*clustering*)
    - ideally, the system looks for the *natural structure* of the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

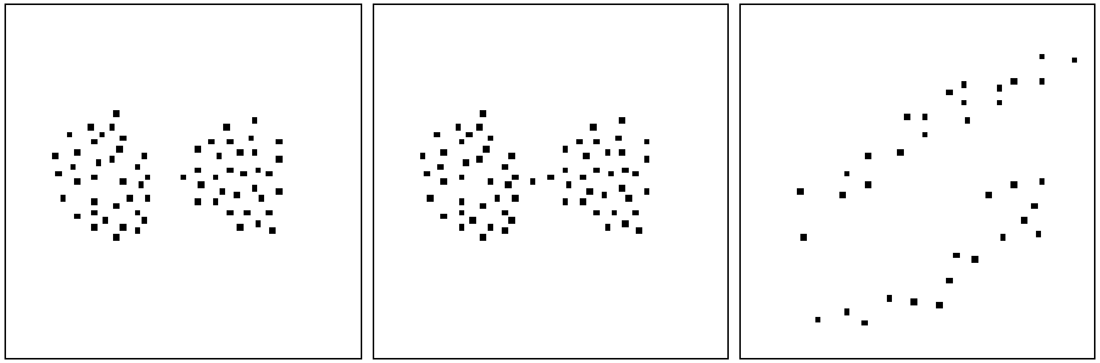




Flavours of machine learning

- 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

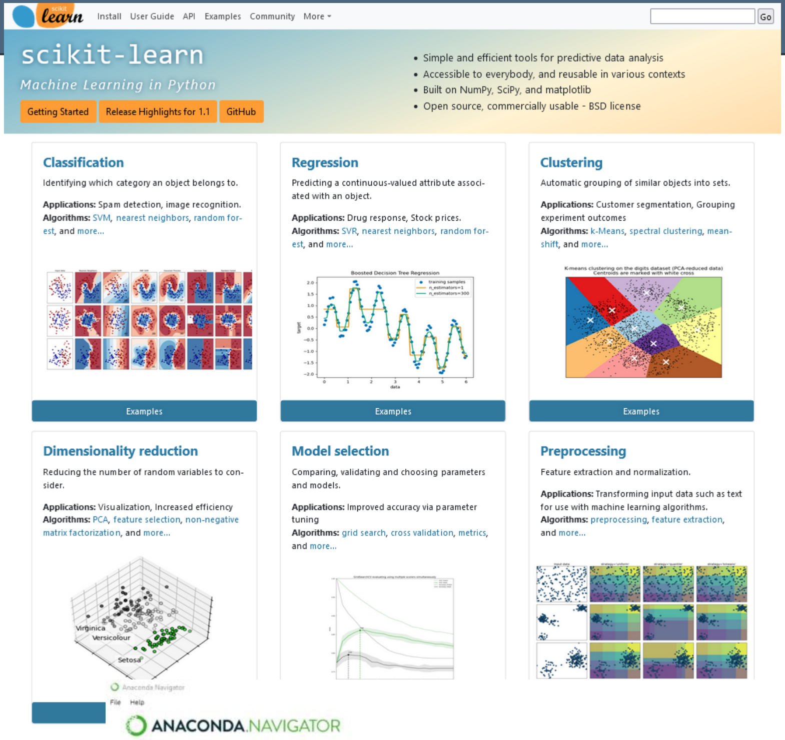
Contents

(suggested) Development framework

- Python 3.x (e.g. 3.13) (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)


Install User Guide API Examples Community More

scikit-learn

Machine Learning in Python

Getting Started | Release Highlights for 1.1 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

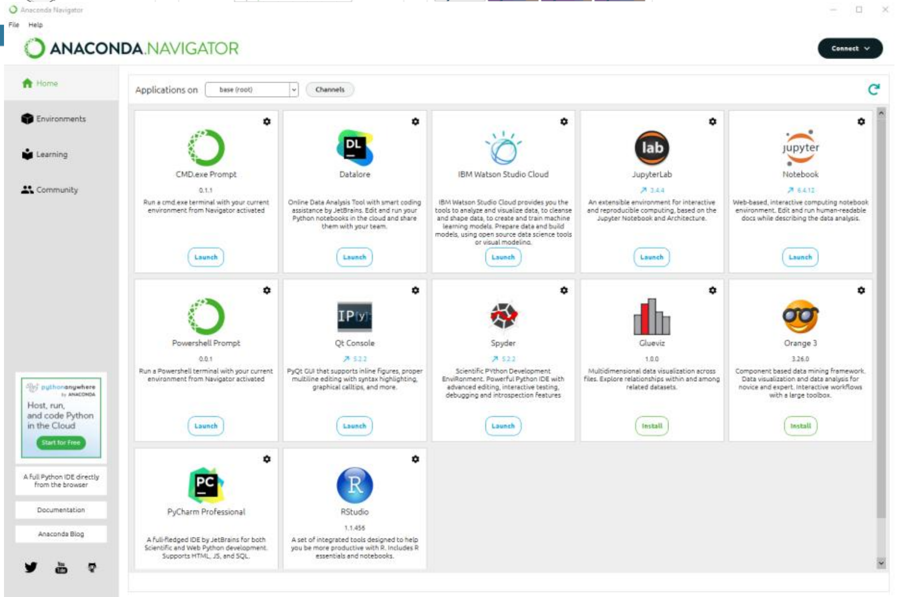
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...

Examples
Boosted Decision Tree Regression
K-means clustering on the digits dataset (PCA-reduced data) Centroids are marked with white cross
Virginica
Versicolour
Setosa


ANACONDA.NAVIGATOR

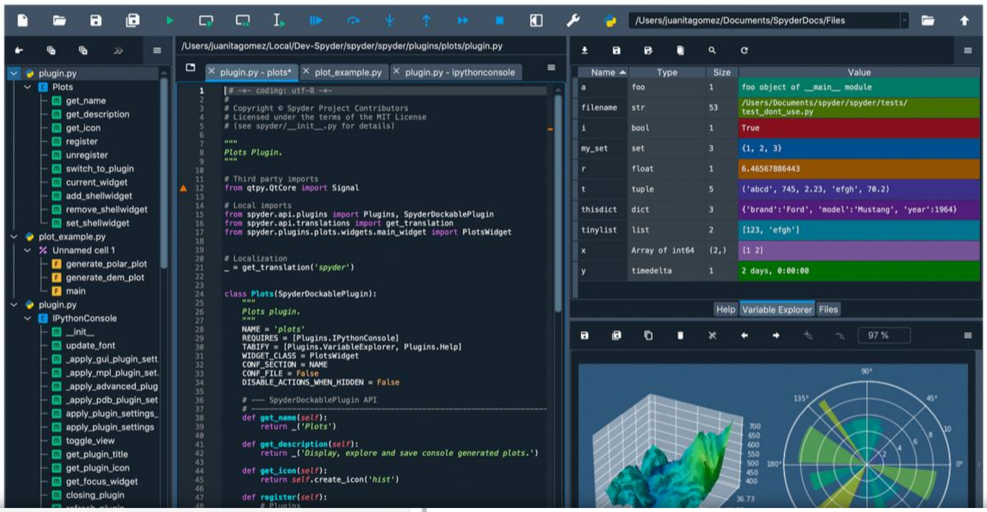
Applications on base (root) Channels

CMD.exe Prompt 0.1.1 Run a cmd.exe terminal with your current environment from Navigator activated	Datalore Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.	IBM Watson Studio Cloud IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.	JupyterLab 3.4.4 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.	Notebook 6.4.12 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Powershell Prompt 0.0.1 Run a Powershell terminal with your current environment from Navigator activated	Qt Console 5.2.2 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.	Spyder 5.2.2 Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features	Glueviz 1.0.0 Multidimensional data visualization across files. Explore relationships within and among related datasets.	Orange 3 3.26.0 Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
PyCharm Professional A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.	RStudio 1.1.456 A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.			

Alberto Ortiz (updated 23/09/2025)

(suggested) Development framework

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



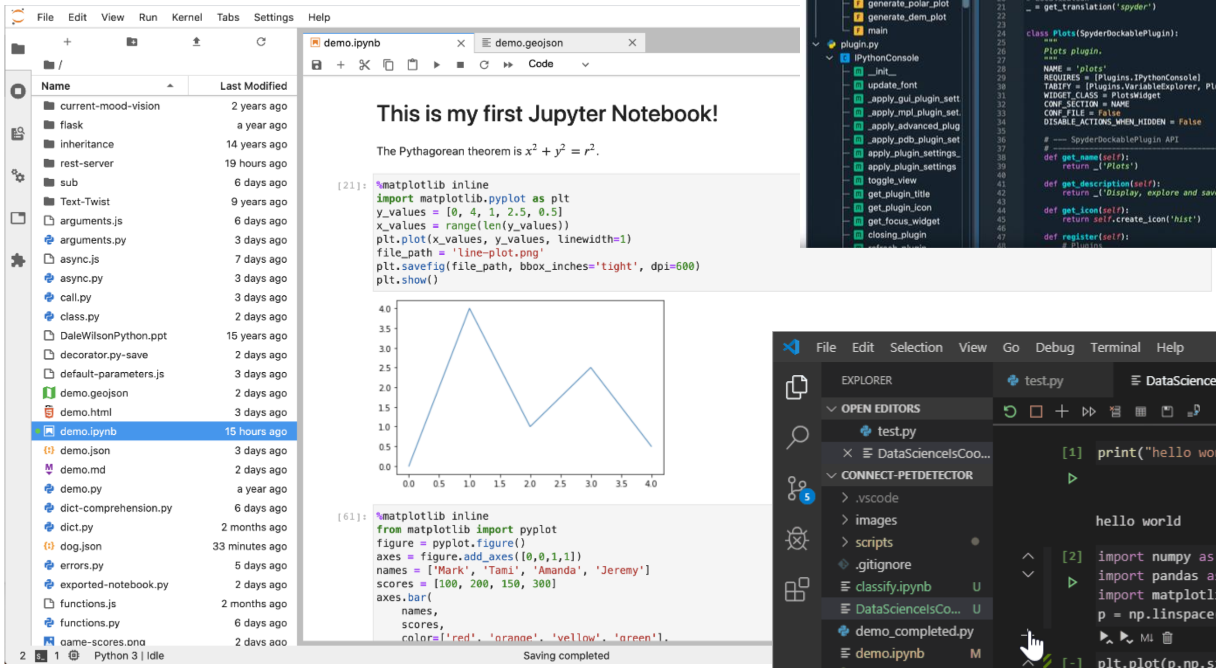
/Users/juanitagomez/Local/Dev-Spyder/spyder/spyder/plugins/plots/plugin.py

plugin.py - plots* | plot_example.py | plugin.py - ipythonconsole

```
# -*- coding: utf-8 -*-
#
# Copyright © Spyder Project Contributors
# Licensed under the terms of the MIT License
# (see spyder/__init__.py for details)

"""
Plots Plugin.
"""

# Third party imports
from qtpy.QtCore import Signal

# Local imports
from spyder.api.plugins import Plugins, SpyderDockablePlugin
from spyder.api.translations import get_translation
from spyder.plugins.plots.widgets.main_widget import PlotsWidget


# Localization
_ = get_translation('spyder')


class Plots(SpyderDockablePlugin):
    """
    Plots plugin.
    """
    NAME = 'plots'
    REQUIRES = [Plugins.IPythonConsole]
    TABIFY = [Plugins.VariableExplorer, Plugins.Help]
    WIDGET_CLASS = PlotsWidget
    CONF_SECTION = NAME
    CONF_FILE = False
    DISABLE_ACTIONS_WHEN_HIDDEN = False

    # --- SpyderDockablePlugin API
    # ------------------------------------------------------------
    def get_name(self):
        return _('Plots')

    def get_description(self):
        return _('Display, explore and save console generated plots.')

    def get_icon(self):
        return self.create_icon('hist')

    def register(self):
        # Plugins
```

/Users/juanitagomez/Documents/SpyderDocs/Files

Name	Type	Size	Value
a	foo	1	foo object of __main__ module
filename	str	53	/Users/Documents/spyder/spyder/tests/test_dont_use.py
i	bool	1	True
my_set	set	3	{1, 2, 3}
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
thisdict	dict	3	{'brand':'Ford', 'model':'Mustang', 'year':1964}
tinylist	list	2	[123, 'efgh']
x	Array of int64	(2,)	[1 2]
y	timedelta	1	2 days, 0:00:00



Name	Last Modified
current-mood-vision	2 years ago
flask	a year ago
inheritance	14 years ago
rest-server	19 hours ago
sub	6 days ago
Text-Twist	9 years ago
arguments.js	6 days ago
arguments.py	3 days ago
async.js	7 days ago
async.py	3 days ago
call.py	3 days ago
class.py	2 days ago
DaleWilsonPython.ppt	15 years ago
decorator.py-save	2 days ago
default-parameters.js	3 days ago
demo.geojson	2 days ago
demo.html	3 days ago
demo.ipynb	15 hours ago
demo.json	3 days ago
demo.md	2 days ago
demo.py	a year ago
dict-comprehension.py	6 days ago
dict.py	2 months ago
dog.json	33 minutes ago
errors.py	5 days ago
exported-notebook.py	2 days ago
functions.js	2 months ago
functions.py	6 days ago
game-scores.png	2 days ago

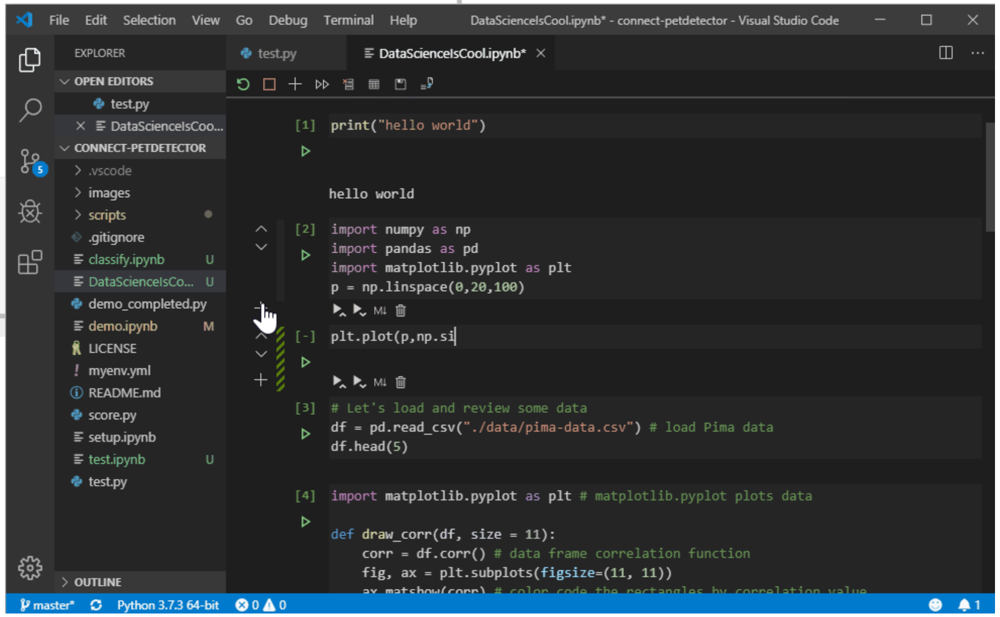
This is my first Jupyter Notebook!

The Pythagorean theorem is $x^2 + y^2 = r^2$.

```
[21]: %matplotlib inline
import matplotlib.pyplot as plt
y_values = [0, 4, 1, 2.5, 0.5]
x_values = range(len(y_values))
plt.plot(x_values, y_values, linewidth=1)
file_path = 'line-plot.png'
plt.savefig(file_path, bbox_inches='tight', dpi=600)
plt.show()
```

```
[61]: %matplotlib inline
from matplotlib import pyplot
figure = pyplot.figure()
axes = figure.add_axes([0,0,1,1])
names = ['Mark', 'Tami', 'Amanda', 'Jeremy']
scores = [100, 200, 150, 300]
axes.bar(
    names,
    scores,
    color=['red', 'orange', 'yellow', 'green'],
```

- Visual Studio IDE (code.visualstudio.com)



```
[1]  print("hello world")

hello world

[2]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     p = np.linspace(0,20,100)

[-]  plt.plot(p,np.si

[3]  # Let's load and review some data
     df = pd.read_csv("./data/pima-data.csv") # load Pima data
     df.head(5)

[4]  import matplotlib.pyplot as plt # matplotlib.pyplot plots data

     def draw_corr(df, size = 11):
         corr = df.corr() # data frame correlation function
         fig, ax = plt.subplots(figsize=(11, 11))
         ax.matshow(corr) # color code the rectangles by correlation value
```

Alberto Ortiz (updated 23/09/2025)

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