Machine Learning and Artificial Intelligence

Lab 01 – Introduction

14/03/2022

An initial FAQ

- What will we learn in this lab?
  - Implementations of various algorithms seen during the lectures in *Python*.

- What can we do if we have doubts regarding the arguments treated during the labs?
  - You can contact me anytime via mail at geri.skenderi@univr.it

An initial FAQ

- How are the lab lectures structured?
  1. Revision of the code from the previous labs.
  2. Brief theory recap
  3. Coding exercises

- You will work on the code during the lab by yourself or with your colleagues and we will then see possible solutions together.

Recap: Machine Learning

- **Machine learning:** The study of computer algorithms get better (automatically) at some task though experience.

- Starting from a set of data, we can try to understand underlying tendencies in order to then make predictions or decisions without being explicitly programmed to do so.

- Simple example: Given a set of images, we need to try and understand if a new never seen before image contains a cat or not.

Recap: Machine Learning

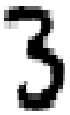
- **The above task is very intuitive and easy for humans**: partially because of mechanisms we don't fully understand yet.

- On a computing level, these tasks are not simple.

- I like to think of ML as a shift from a procedural and propositional paradigm to one related with uncertainty, which trying to emulate human learning.

Example

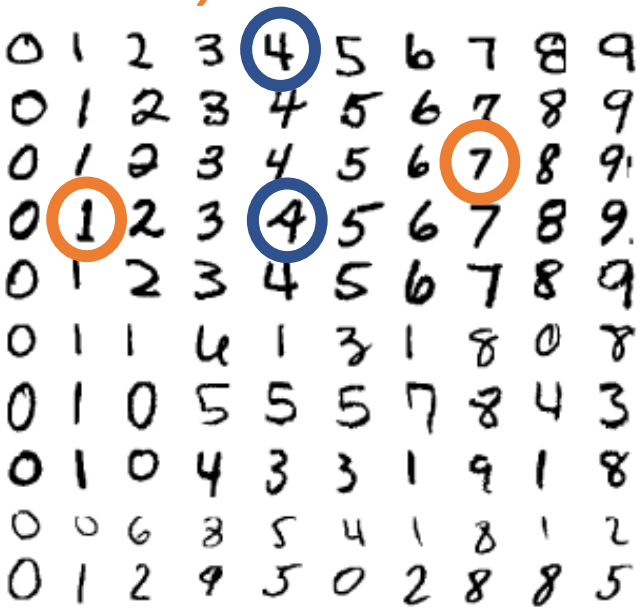
What we see



What the computer sees

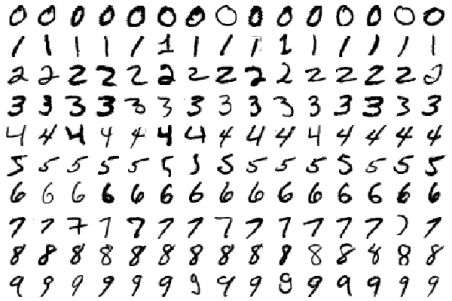
```
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.4 0.4 0.4 0.4 0.4
0.1 0.6 0.6 0.6 0.6 0.6
0.0 0.3 0.6 0.6 0.6 0.6
0.0 0.0 0.0 0.2 0.2 0.1
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
```

Variability



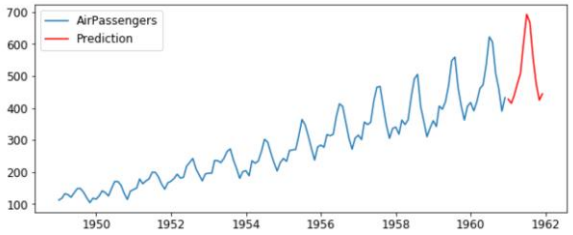
Some more examples

Understand that this is an apple



Distinguish different characters written by hand

Identify the most important object in a scene

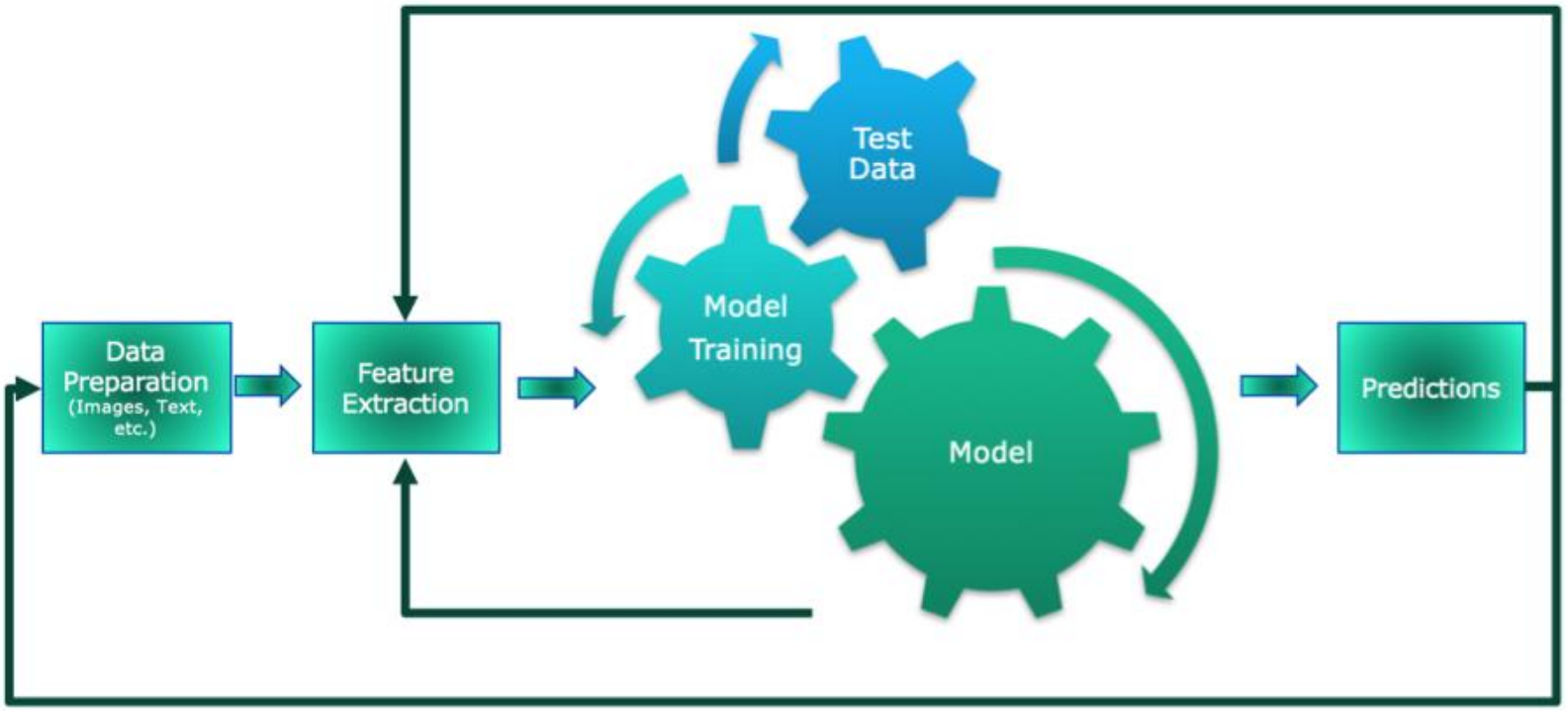


Predict the future values of a time series

More recent examples:

- Live scene understanding:
  - http://places2.csail.mit.edu/demo.html
- Understanding images through natural language:
  - https://openai.com/blog/clip/
- Image generation from text queries:
  - https://openai.com/blog/dall-e/

A Standard Machine Learning Pipeline



Learning from experience

ML and Python

- Python is a high-level, multi-purpose, cross-platform programming language.
- Can be used in an object oriented or functional manner
- In recent years, it has been widely used for scientific computing, especially for ML.

Python

Weird stuff:

- Dynamic typing
- Indentation is *very* important
- Multiple differences between version 2.7 and 3+

IDEs:

- PyCharm (specific IDE)
- VS Code (my personal favourite)
- Bash command *python* opens an interactive shell

https://docs.python.org/3/tutorial/

Python package manager

Python has an internal system for downloading, storing, and resolving packages. The most common packages (open source, third-party) are available in a repository called the Python Package Index (PyPI).

```
pip install numpy
pip install scipy==1.2.1 # install specific version
pip install --user Pillow # install only for current user
```

A requirements.txt file is handy for saving all the packages on which the project depends:

```
pip install –r requirements.txt
```

https://docs.python.org/3/tutorial/

Importing packages

Python is so popular because of its packages and libraries. Here's an import of numpy in the shell:

```
>> import numpy as np
```

N.B.: pip installs packages system-wide (globally). It may not always be the best solution (compatibility issues between versions, need to be superuser etc.)

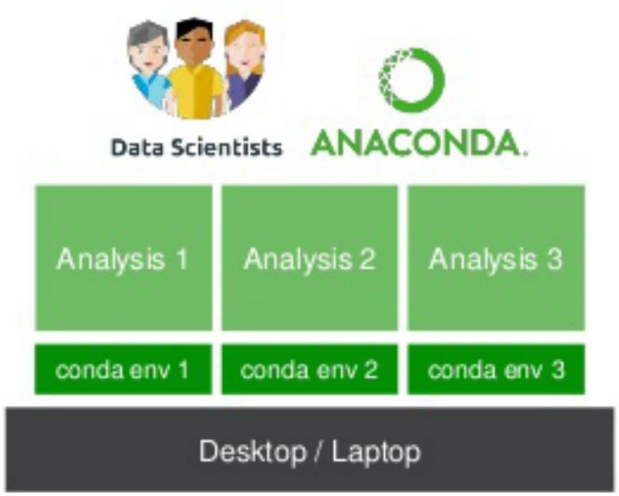
https://www.dabapps.com/blog/introduction-to-pip-and-virtualenv-python/?utm_source=feedly

Conda

Create an isolated *virtual* environment

To install → https://www.anaconda.com/products/individual#Downloads

Legacy → https://repo.anaconda.com/archive/



Creating a virtual env

```
conda create -n <name> python=3.8
```

Activating and deactivating the env

```
conda activate <name>
conda deactivate
```

Python: Lists, vectors and matrices

```
>> a = 5               # scalar
>> b = [2, 3]          # list, len(b): 2
>> c = [[1,2],[3,4]]  # list of lists. len(c): 2
```

```
>> import numpy as np
>> d = np.array([0])           # scalar. len(d): 1, d.shape: (1,)
>> e = np.array([2, 3])        # vector (array). len(e): 2, e.shape: (2,)
>> F = np.array([[1,2],[3,4]] # matrix. len(F): 2, F.shape: (2,2)
```

Extracting data from matrices

```
>> a = [10,20,30,40]
>> b = a[0:2] # b = [10,20]
>> A = np.array([[10,20,30],[40,50,60]])
>> B = A[:2,:2]
```

Matrix operations

Crucial for scientific computing and also Machine Learning.

```
>> B = A + A
>> C = A ** 2 # power of 2
>> C = A * A # element-wise
>> C = np.dot(A,A)
>> D = np.divide(A,A) # element-wise
>> D = np.dot(A, np.linalg.inv(A))
>> np.sum(A)
>> np.sum(A, axis=0) # over rows
>> np.mean(A, axis=1) # over cols
```

Matrix product: $c_{i,j} = \sum_k A_{i,k} B_{k,j}$

Matrix division: $D = AA^{-1}$

Running Python code

- Python scripts are simple text files with a .py extension
- To execute a script: python <name>.py
- Debugging: depends on the IDE
- A lot of useful information can be found in the python cheatsheet in the material.

Visualization

The most popular libraries for visualization are:

1. Matplotlib: for the visualization and plotting of https://matplotlib.org/contents.html

2. Pillow: for the visualization of images https://pillow.readthedocs.io/en/latest/

3. Seaborn: More elegant and statistical plots: https://seaborn.pydata.org/

Jupyter Notebook

- Unique set-up for programming: open-source web-based application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. https://jupyter.org/

- Very useful for sharing and interactive output functionalities.

- Files end with a .ipynb extension.

Google Colab

- Free platform for developing Python code on cloud based on Jupyter Notebooks provided by Google.

- PROs:
  - Most scientific Python libraries already installed
  - Access to hardware accelerated GPUs and TPUs
  - Ability to share code
  - *I will run code and share it with you via Colab*

https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c

Exercises

exercises.txt