MLRF Lecture 01

J. Chazalon, LRDE/EPITA, 2021

Course outline

Lecture 01 part 02

What is it about?

Actual goals of this course

Teach you that you can (and should whenever possible) **optimize the parameters** of your CV/PR product.

Show you some **simple tools** to try to do it, on simple and less simple examples.

Make you realize the importance (and complexity) of **evaluation**, just like testing.

This will require to have data to work on, and describe such data in a ML-friendly way ⇒ a good deal of **feature extraction** in contained in this course.

Actual content of this course

Address practical problems: describe a pattern, look for a pattern, match a pattern, classify a pattern, describe a set of patterns (an object / an image), retrieve an object given a query, segment objects…

...and face the unavoidable work surrounding them: problem definition and modeling, evaluation, data pre-processing and post-processing…

How?

Course agenda

6 “weeks” (Friday to Friday).
See the web page for complete agenda.
Weekly tests + assignments (practice sessions). No final exam.

Weekly workflow should be:

- **Friday, 09:30-10:00:** Answer the weekly quiz on Moodle *(starting next Friday)*
- **Friday, 10:00-12:00:** Attend the lecture using Teams
- **Friday, 14:00-17:00:** Work on the practice session and join the discussion using Teams
- **Before next Friday:** Complete the assignment and submit your results using Moodle *(for sessions 4, 5 and 6 only)*

No deep learning (DL)!

We need a course about basic techniques.

Feature extraction, clustering, classification…

There are cases where setting up a deep learning pipeline is not relevant.

Not enough examples, “simple” problem…

This should be an introduction to a more advanced DL course.

Deep learning techniques “blend” some of the techniques we’ll see.

Deep learning techniques still require pre- and post-processings.

Evaluation is of paramount importance in deep learning.

Pedagogical approach

Case-based

We will focus on practical problems and try to solve them.

Developer-friendly

We will see code examples and implement problem-solving techniques.

A balanced amount of theory and practice

We will introduce theoretical aspect when we need them to solve problems.

Don’t get bored

We will follow a smooth learning curve,
alternating between lectures and practice sessions.

Practice sessions: setup your dev. env.

Follow the instructions on the webpage.

Basically: Python with:

- Jupyter
- Numpy
- Matplotlib
- Scikit-image
- Scikit-learn
- OpenCV


WOULD YOU LIKE TO KNOW MORE?

Going further: Suggested books

- **R. Szeliski, Computer Vision Algorithms and Applications, Springer, 2011**
- **R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley & Sons, 2001**
- **C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006**
- **A. Cornuéjols, L. Miclet, L'apprentissage artificiel - Concepts et algorithmes, Editions Eyrolles, 2010**
- **R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004**
- J. Leskovec, A. Rajaraman, J. Ullman, Mining of Massive Datasets, Cambridge University Press, 2014
- F. Cao, J.L. Lisani, J.M. Morel, P. Musé, F. Sur, A Theory of Shape Identification, Springer, 2008
- G. Bradski, A. Kaehler, Learning OpenCV, O'Reilly, 2008
- R. Laganière, OpenCV 2 Computer Vision Application Programming Cookbook, Packt Publishing, 2011
- J.E. Solem, Programming Computer Vision with Python, O'Reilly, 2012

Going further: Courses

http://work.caltech.edu/telecourse.html: Learning from Data, Caltech

http://mmds.org/: Mining of Massive Datasets, Stanford

https://www.coursera.org/instructor/andrewng: Courses from Andrew Ng

…

Going further: Research work

Scientific societies

- IAPR
- The Computer Vision foundation

Journals

- Pattern Analysis and Machine Intelligence
- Transactions on Image Processing
- Pattern Recognition
- Pattern Recognition Letters
- …

Conferences

- CVPR
- ECCV
- ICCV
- ICDAR
- ICPR

Major software tools

We will use:

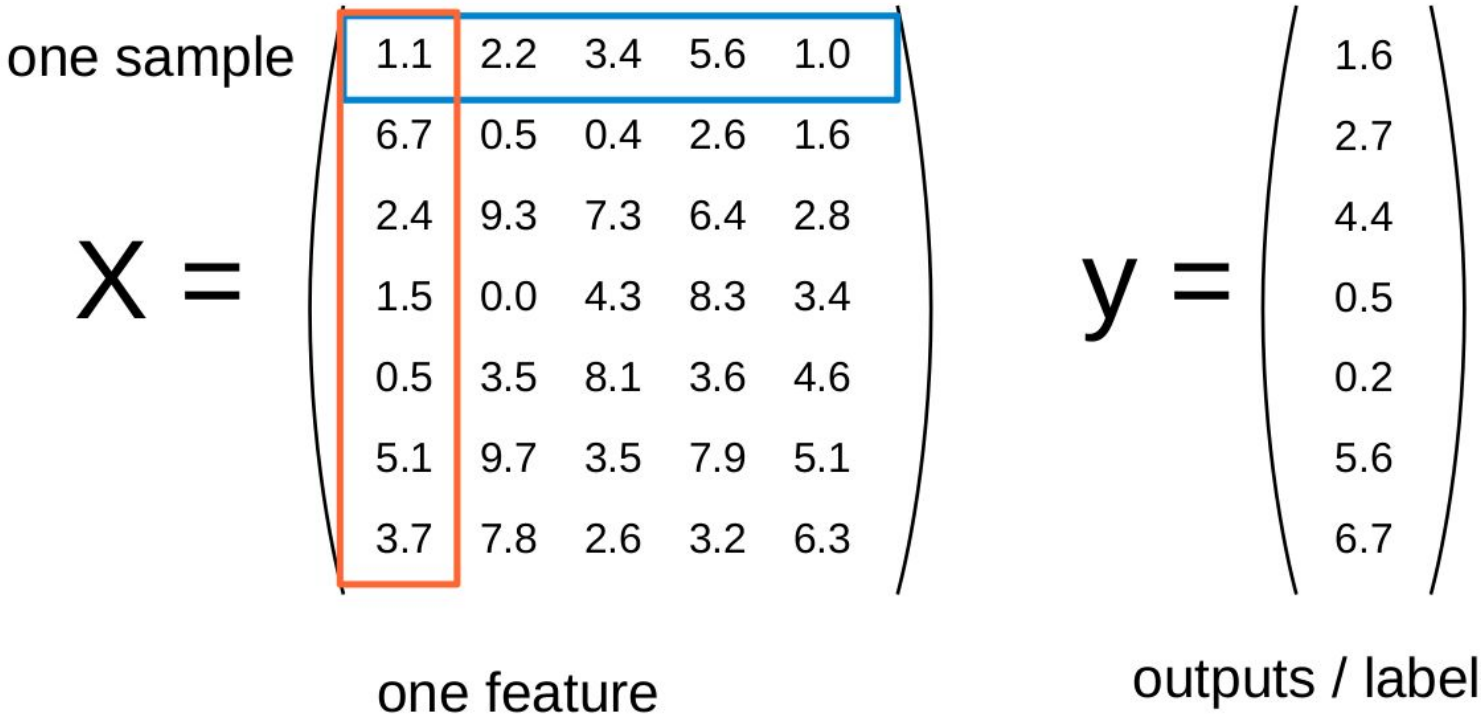
- Python, SciPy, Numpy, Matplotlib…
- OpenCV
- scikit-image
- Scikit-learn
- ~~VLFeat~~
- ~~Dlib~~
- …

We will not use:

- Keras
- Caffe2
- Tensorflow
- Pytorch
- Theano
- mxnet

Why I 🖤 scikit-learn

Numpy-friendly



3 way documentation: User guide, API ref, Examples

1.9. Ensemble methods

The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- In **averaging methods**, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

  Examples: *Bagging methods*, *Forests of randomized trees*, ...

- By contrast, in **boosting methods**, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

  Examples: *AdaBoost*, *Gradient Tree Boosting*, ...

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

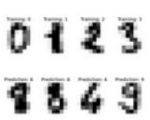
Parameters: **n_estimators** : integer, optional (default=10)

The number of trees in the forest.

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

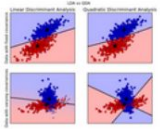
Examples



Recognizing hand-written digits

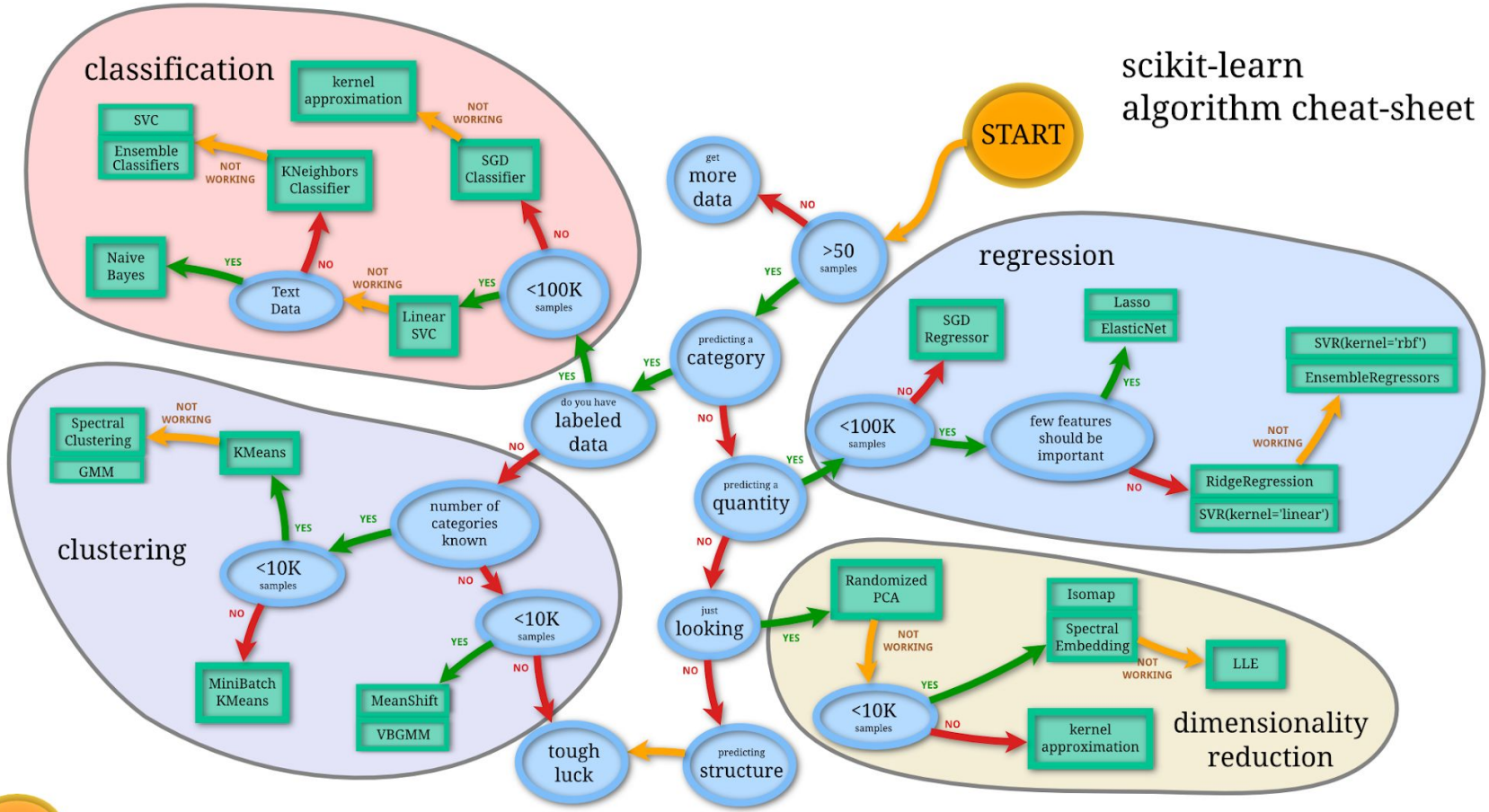
Plot classification probability

Classifier comparison



Linear and Quadratic Discriminant Analysis with confidence ellipsoid

Super smart API: *decomposition, level of detail, default values, consistency, etc.*



Back

scikit learn

With evaluation tools for each predictor and transformer!