MLRF Lecture 01

J. Chazalon, LRDE/EPITA, 2020

Course outline

Lecture 01 part 02

Why, oh, why?

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 + 1 extra test
See the official web page for complete agenda.
Weekly tests + assignments (practice sessions 4, 5 and 6). No final exam.

Weekly workflow should be:

- **Before Monday:** watch the lecture (see below for links) and prepare questions.
- **On Monday morning:** attend the Q/A session using Teams.
- **On Monday afternoon:** work on the practice session and join the discussion using Teams.
- **Before Sunday evening:** (for sessions 4, 5 and 6 only) complete the assignment and submit your results using Moodle.
- On the **next Monday morning** (before the Q/A session): Complete the test about previous lecture and practice using Moodle.

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 at
https://www.lrde.epita.fr/~jchazalo/teaching/MLRF/202004_IMAGE_S8_SCIA_S8/

Download and run the first practice session to test.

Feel free to start working on practice sessions before Monday afternoon.

Do you want 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
- ~~VLFeat~~
- scikit-image
- scikit-learn
- …

We will not use:

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

Why I 🖤 scikit-learn

from A. Müller

Numpy-friendly

one sample

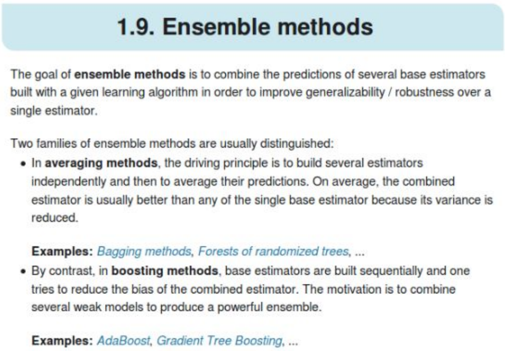
$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

one feature

outputs / labels

from A. Müller

3 way documentation: User guide, API ref, Examples



1.9. Ensemble methods

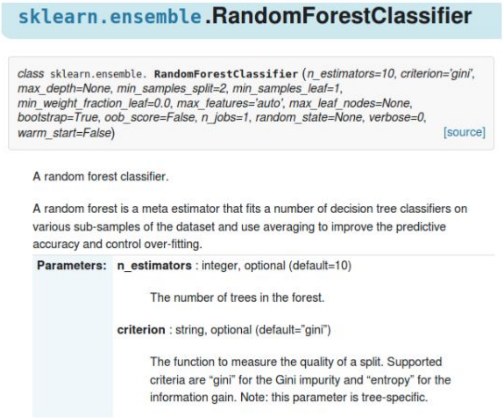
The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

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.

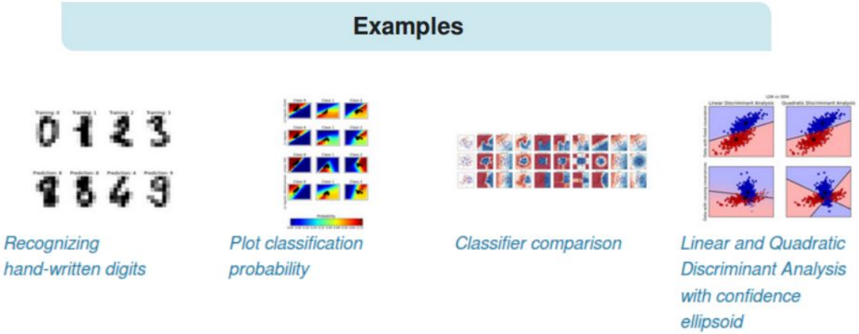
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

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

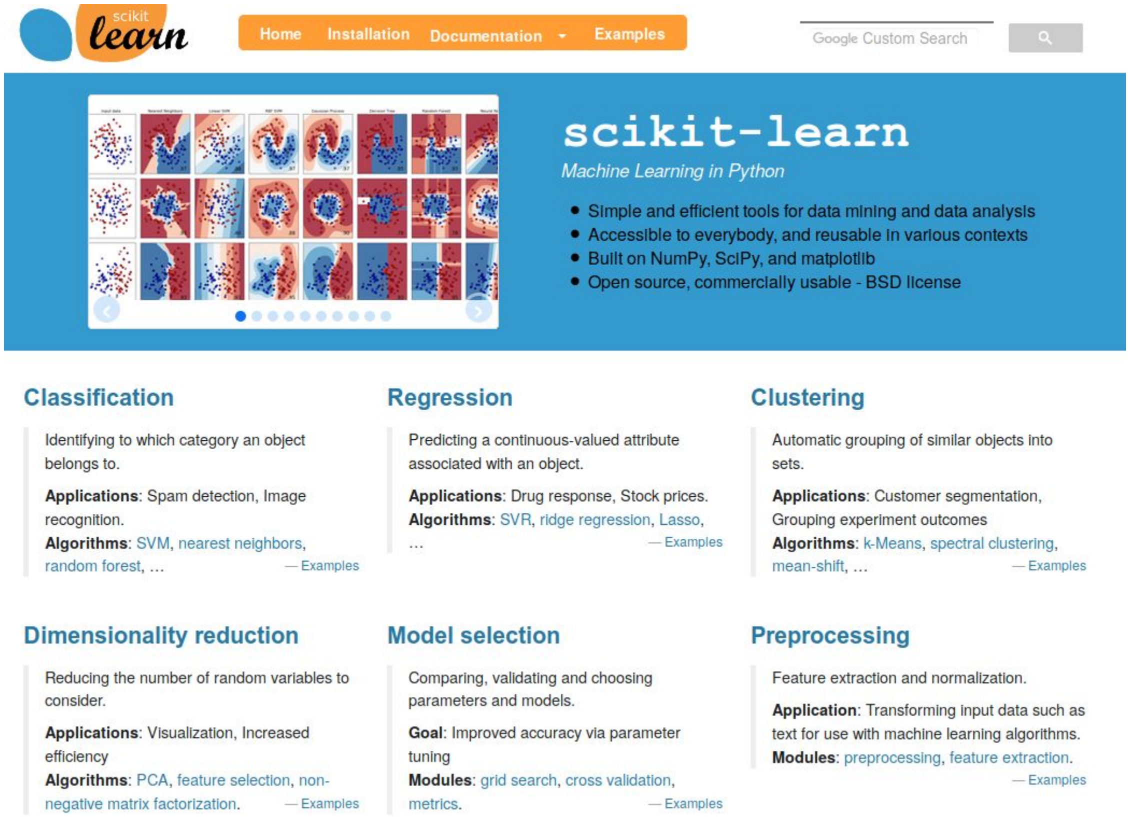
Recognizing hand-written digits

Plot classification probability

Classifier comparison

Linear and Quadratic Discriminant Analysis with confidence ellipsoid

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



scikit-learn

Machine Learning in Python

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

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

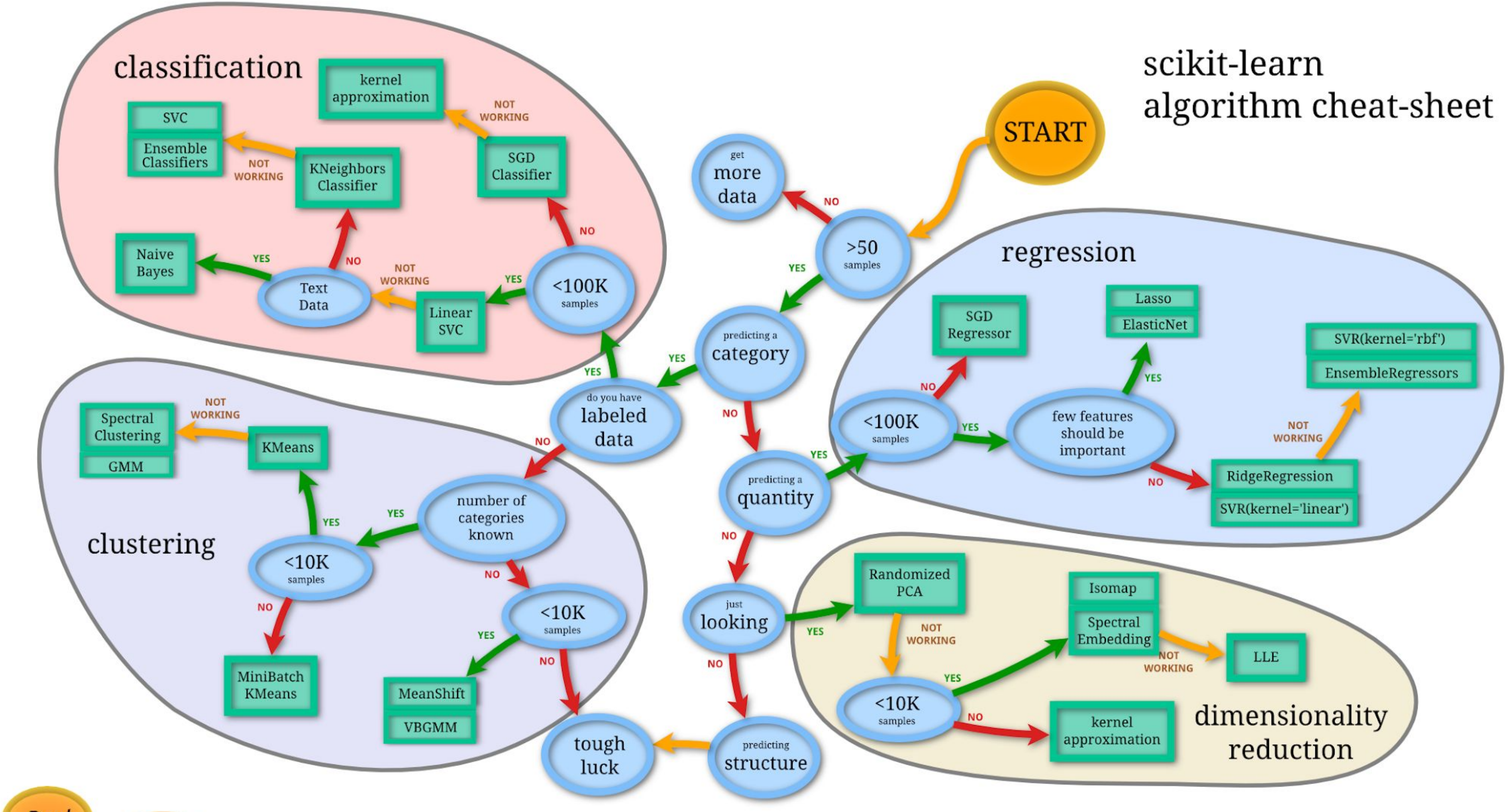
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction. — Examples



With evaluation tools for each predictor and transformer!