Practical Random Forest

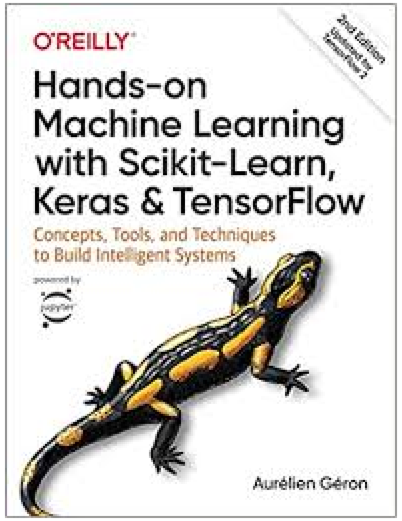
Overview

- one same project of object oriented design and implementation in python
- project is a machine learning method, random forests
- applied to 3 important tasks in machine learning : classification, regression, anomaly detection, plus
  - different impurity measures
  - optimized version "*extra trees*"
  - feature importance
  - Python coding style and logging
- more importantly, **all this needs to apply object oriented principles and several design patterns**

Why object-oriented

- It *is* possible to build a random forest classifier without classes, just a bunch of functions, like in this messy implementation.
- But then *extending* it to regression, anomaly detection, compute feature importance etc. becomes difficult.
- By applying OO principles and patterns we can build a design and implementation where such extensions, and probably others, can be added gracefully, easily.
- At the same time following a code style will make the code much more easy to understand.

Reference

 O'REILLY 2nd Edition Updated for TensorFlow 2 Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems powered by jupyter Aurélien Géron	*Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Aurélien Géron, O'Reilly (2019) Electronic version at UAB library

What we need is in pages 1-31 (Ch. 1), 175-186 (Ch. 6) and 189-199 (Ch. 7). This selected pages are also in campus virtual.

You can read also the Wikipedia entry.

Marks

- the project is divided into 3 steps or milestones
- **what to do exactly for each milestone is in a set of slides I'll show you along the course**
- **then the goals of the next slides will become clear**
- each milestone has its own grading scheme, see next slides
- the possible marks of a milestone are A=10, B=7.5, C=5, D=2.5, E=0.
- you get E when no show, the work is a copy or does not achive the minimum set for D

First milestone

Goals

- design and implement a random forest for classification
- design in UML notation
- test it on two small datasets: Sonar and Iris
- start with the Gini Index as the only split criterion and then extend to others
- optimize the algorithm in two ways: "*extra-trees*" and parallelize it
- test the optimized versions on the larger MNIST dataset
- practice logging in Python
- start choosing good names for classes, methods etc. and write good comments

Grading

- C :
    1. the design follows OO principles
    2. only the Gini index
    3. code works well on the two small datasets
    4. the UML design has enough detail and is right and correspond to the code
    5. logging sentences to debug/trace the execution instead of `print`
    6. right names for classes, methods, attributes, variables, constants + comment for each class and key methods

- B : C plus
  1. design easily extendable with new impurity measures by applying a certain design pattern
  2. added entropy
- A : B plus
  1. optimization by parallelism, small number of values and extra-trees
  2. comparison of elapsed times, both for training and prediction
  3. good accuracy on MNIST for each optimization
- D : the classifier works fairly well and code corresponds to an existing UML design, some but not all of the necessary OO principles have been applied

Deliverables

Upload into campus virtual a file named `first_milestone.zip` with

1. a text file `autors.txt` with the NIU and names of authors
2. a file `source_code.zip` with Python source code, the plantUML class diagram `design.puml` and a `design.png` image of the UML design (exported)
3. a file `results.pdf` (export a presentation) with the parameters of the experiments for Sonar and Iris and the accuracies obtained
4. if you have done the optimization part, a file `times.pdf` (export a presentation) with the tables comparing exection times

Do not include the datasets.

Second milestone

Goals

- modify and extend the design to also do regression, avoiding duplicate code by employing inheritance and a certain desing pattern
- print the learned decision trees and compute feature importance by applying another design pattern
- keep the good practices of logging, comments and naming

Grading

- C :

    1. regression part is done and works
    2. the test with the minimum daily temperature dataset achieves low RMSE error
    3. the UML class design is updated with regression
    4. more logging sentences to debug/trace the execution
    5. right names for classes, methods etc. plus comments

- B : C plus

    1. feature importance and print trees done by applying a certain design pattern
    2. the UML class design is updated with feature importance
    3. tree printing and feature importance work well on small datasets Sonar and Iris

- A : B plus
    - feature importance on MNIST using extra-trees produces a reasonable result
- D : C.i is done but one or more of the other items in C are wrong or missing

Deliverables

Within a file `second_milestone.zip` put items 1 and 2 like in the first milestone, plus a file `results.pdf` with :

- figures of regression and RMSE achieved for the prediction of the last 1 and 2 years of the dataset min. daily temperature
- two printed trees for Iris
- computed importances of features of datasets Sonar and Iris
- figure of the importance of each pixel for the dataset MNIST, shown as an image, adding as scale a `plt.colorbar()`

Third milestone

Goals

- extend the design to accomodate a new machine learning task, anomaly (outlier) detection
- keep doing logging
- edit the code to conform to Python coding style PEP-8
- write good comments and documenting
- use assertions to check some methods contract

Grading

- C :
  1. anomaly detection done with the isolation forest algorithm works well on the synthetic and credit card fraud datasets
  2. the UML design has been updated accordingly
  3. comments are at the right places and with the right content
- B : C plus
  1. the code has no PEP-8 problems or warnings, or very few, and when they exists there is a reason explained in a comment
  2. the classes intended to the users have been documented with doc strings (avoiding copy-paste), and their documentation generated with Sphinx

- A : B plus
  - isolation forest, which requires the extra-trees optimization, finds reasonable anomalies in the MNIST dataset, and precision and recall are similar to those reported in the slides
  - type hints for every function, class method, argument and class attribute
- D : isolation forest design and implementation is not perfect but still detect some ouliers, or the UML design is wrong or missing

Deliverables

Within a file `third_milestone.zip` put items 1 and 2 like in the first milestone, plus

- a file `results.pdf` with :
    - result figure for synthetic dataset
    - what are your outliers within the first 10K samples of the credit card fraud dataset and top 0.5% scores, the precision and recall
    - outliers detected for the MNIST dataset for 3 digits at least, for top 0.2% scores
- a file `style.txt` with the remaining PEP-8 problems and why
- the documentation generated by Sphinx