Machine Learning B

2024-2025

Home Assignment 7

Yevgeny Seldin and Amartya Sanyal
Department of Computer Science
University of Copenhagen

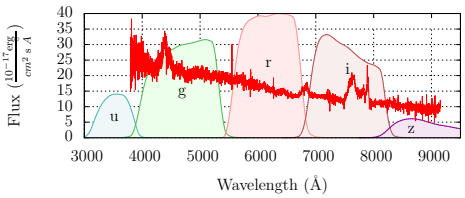
The deadline for this assignment is **12 June 2025, 17:00**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your full source code in the PDF file, only selected lines if you are asked to do so.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF. The programming language of the course is Python.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in *SpeedGrader*. Zipped PDF submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Handwritten solutions will not be accepted. Please use the provided latex template to write your report.

(a) Observatory

(b) Image Data



(c) Spectrum (Quasar)

Figure 1: The Apache Point Observatory shown in Figure (a) gathers both images and spectra. The image data are given in terms of grayscale images that are based on five filters covering different wavelength ranges, called the `u`, `g`, `r`, `i`, and `z` bands. In Figure (b) an RGB image is shown that is based on such images of a particular region. For a small subset of detected objects (white squares), detailed follow-up observations in terms of spectra are available, see Figure (c). Image sources for Figures (a) and (b): `https://www.sdss.org`

1 XGBoost (30 points) [Amartya]

An important problem in astrophysics is to estimate the distance of objects to our Earth. Since one cannot directly measure these distances, one resorts to the light emitted by the objects and that reaches our Earth. In particular, one considers the so-called *redshift* z of an object: the larger the redshift, the larger is the distance an object to our Earth. This redshift can be measured very accurately in case one has access to the detailed spectrum of the light for an object at hand. Unfortunately, obtaining such spectra is very time-consuming and are, hence, only available for a small subset of the detected objects, see Figure 1. Instead, one has access to image data, which is used to estimate the redshift of the detected objects.

Note that you do not need to understand the physical details given above to address this exercise (they are just provided for the sake of completeness). All you have to know is that this task can be formalized as regression problem and that you have access to a dataset $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^{10} \times \mathbb{R}$, which is stored in the file `quasars.csv`. Each line contains, for each object (row), a target value $y_i \in \mathbb{R}$ (last column) and a vector $\mathbf{x}_i \in \mathbb{R}^{10}$ of ten attributes (first 10 columns).

Your task is to build XGBoost regression models that can be used to predict the target (redshift) for new, unseen objects.[1] The corresponding library is available at `https://xgboost.readthedocs.io/en/stable/`.

[1]The dataset contains data of so-called "quasars", which belong to the most distant objects that can be observed from our Earth!

1. Start by loading the dataset (`quasars.csv`)[2] and split it up into a training set containing about 80% and a test set containing about 20% of the instances.

2. Fit a XGBoost regression model (square loss) using the following parameter assignments (stick to the default assignments for the remaining parameters): `colsample_bytree=0.5`, `learning_rate=0.1`, `max_depth=4`, `lambda=1`, `n_estimators=500`. Train the model on about 90% of the training data and make use of an hold-out validation set containing about 10% of the training set to monitor the training process by plotting both the training and the validation RMSE (y-axis) vs. the boosting iterations (x-axis).

   Finally, make predictions for the test set via the fitted model and compute the induced RMSE and the R2 score on the test set.

3. Next, conduct a grid search to find good parameter assignments for your XGBoost model. Include the parameters used above as a starting point and extend the grid. Select the parameter combination with the lowest 3-fold cross validation RMSE (on the training set).

   Refit your model on all the training instances using the best parameter configuration. What is induced RMSE/R2 score on the test set? Can you beat a simple nearest neighbors regression model ($k = 5$ neighbors) that is fitted on the training set (note that beating this baseline is not a requirement for getting all the points :-))?

Hints: You can find many XGBoost tutorials on the internet. Have a look at some of them! Also, check out the parameters `eval_metric` *and* `eval_set` *of the fit function for monitoring the training process. For the grid search, note that you can combine XGBoost with Scikit-Learn; the whole grid-search can be conducted in a few lines of code.*

2 A simple version of Empirical Bernstein's inequality (30 points) [Yevgeny]

Solve Exercise 2.7 in Yevgeny's lecture notes (the updated version).

[2] Available at `https://sid.erda.dk/cgi-sid/ls.py?share_id=c9SgMJSGik`.

3 PAC-Bayes-Unexpected-Bernstein (40 points) [Yevgeny]

Solve Exercise 3.10 in Yevgeny's lecture notes (the updated version).