Projects
Machine Learning Course
Fall 2017

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
mlo.epfl.ch/page-146520.html
epfmlcourse@gmail.com

Class Project 2
Submission Deadline: Dec 21st, 2017

Introduction

In this project, you will learn to use the concepts we have seen in the lectures and practiced in the labs on a real-world dataset, start to finish. You will be doing exploratory data analysis to understand your dataset and your features, do feature processing and engineering to clean your dataset and extract more meaningful information, implement and use machine learning methods on real data, analyze your model and generate predictions using those methods and report your findings.

Grading. Project 2 counts 30% to your final grade in the course. Within project 2, we will again grade your code, report and competition score (each counting one third).

Logistics

Group formation. For Project 2, you again form a team of 3 members on your own. It can be different from your group in Project 1. If you are still searching for teammates, please use the discussion forum on Moodle. In exceptional cases, if you didn't manage to form a team of 3 by Dec 1st, contact us.

Deliverables at a glance. (More details and grading criteria further down)

- **Code.** In Python. External libraries are allowed, if properly cited.
- **Written Report.** You will write a maximum 4 page PDF report on your findings, using LaTeX.
- **Competitive Part.** To give you immediate feedback and a fair ranking, we use the competition platform kaggle.com to score your predictions. You can submit whenever and almost as many times as you like, up until the final submission deadline.

Pick Your Favorite Among Three Challenges. Pick your favorite competition among the following three online competitions. Don't be influenced by their seemingly different difficulty level, since your contribution as compared to standard approaches will be taken into account in the grading.

Step 1 - Getting started

Pick your favorite competition among the following three: To join and download the dataset, you must first click the corresponding signup link

`https://www.kaggle.com/c/epfml17-rec-sys` : signup link!

`https://www.kaggle.com/c/epfml17-text` : signup link!

`https://www.kaggle.com/c/epfml17-segmentation` : signup link!

For all three possible tasks, we provide some additional description and sample code on the course github:

`https://github.com/epfml/ML_course/tree/master/projects/project2`

Step 2 - Implement ML Methods

You are allowed to use any external library and ML techniques, as long as you properly cite any external code used.

Step 3 - Submitting your Predictions

Once you have a working model (using the above methods or a modified one), you can send your predictions to the Kaggle competition to see how your model is doing against the other teams. You can submit whenever and as many times as you like, until the deadline.

Your predictions must be in `.csv` format, see `sample-submission.csv`. You must use the same datapoint ids as in the test set `test.csv`. To generate `.csv` output from Python, use our provided helper functions in `helpers.py` (see Project 1 folder on github).

After a submission, Kaggle will compute your score on the test set, and will show you your score and ranking in the leaderboard.

Do not use the Kaggle score as the only estimate of your error. Instead, always estimate your test error by using a local validation set, or local cross-validation! This is important to avoid overfitting the test set online. Also, it allows you to make experiments faster, and save uploading bandwidth :). You are only allowed a maximum of 5 submissions to Kaggle per day.

Step 4 - Final Submission of Your Project

Your final submission to the CMT system (the standard system as used for scientific conferences) must consist of the following:

- **Report:** Your 4 page report as `.pdf`
- **Code:** The complete executable and documented Python code, as one `.zip` file.
  Rules for the code part:
  - *Reproducibility:* In your submission, you must provide a script `run.py` which produces *exactly* the same `.csv` predictions which you used in your best submission to the competition on Kaggle. This includes a clear `ReadMe` file explaining how to reproduce your setup, including precise training, prediction and installation instructions if additional libraries are used - the same way as if you would ideally instruct a new teammate to start working with your code.
  - *Documentation:* Your ML system must be clearly described in your PDF report and also well-documented in the code itself. A clear `ReadMe` file must be provided. The documentation must also include all data preparation, feature generation as well as cross-validation steps that you have used.
  - *External ML libraries* are allowed, as long as accurately cited and documented.
  - *External datasets* are allowed, as long as accurately cited and documented.

Submission URL: `http://epfml17.hotcrp.com`
(Don't forget to provide all author names with their EPFL email, your Kaggle team name, and select the right one of the 3 tasks. You can update your submission anytime until the deadline.)

Submission deadline: Dec 21st, 2017 (midnight 23:59.59)

Appendix

Grading Criteria and Some Advice

- **Competitive Part (counts one third).** The final rank of your team in the (private) leaderboard will be translated linearly to a scale from 4 to 6.
- **Code (counts one third).** In Python. You are allowed to use external libraries, as long as properly cited, documented and referenced. Similarly as in Project 1, your project submission will be graded by two assistants independently. As some additional advice for the code part,
  - Double-check that the archive you submit actually contains everything needed to run your solution, and indeed exactly reproduces your best kaggle submission (and select the submission on kaggle accordingly).
  - Make sure any additional installation needed to run your solution is fully described in the accompanying `ReadMe` file, including version numbers if necessary.
  - Try to follow your own instructions to get your code running at least once.
  - Try to make your code as readable as possible. Reduce copy-pasted code to a minimum, use helper functions and clearly name your files, functions and variables.
- **Written Report (counts one third).** You will write a maximum 4 page PDF report on your findings, using LaTeX. We will grade you on the scientific contribution you made, that is on the improvement achieved over the standard baseline methods, as well as the rigorous and fair measuring of the claimed improvements. The criteria are

  - **Solid comparison baselines supporting your claims**
    Quantify the benefits of your method by providing clear quality measurements of the most important aspects and additions you chose for your model. Start with a very basic baseline, and demonstrate what improvements your contributions yield.
  - **Reproducibility**
    Your classmates should be able to reproduce your results based on your report only. Describe what preprocessing is required, what hyper-parameter values you selected and why, and clearly describe the overall pipeline you used.
  - **Scientific novelty and creativity**
    You will likely be using more than the standard methods we saw in the first half of the course. To communicate that your methods work and that you understand them, you should make sure that your report makes clear the following points.
    - What *specific* problem your method is intended to solve.
      By specific, we do not mean "image classification" but what specific issue with your current model are you trying to improve with this method.
    - Why is this an important problem? Why are you solving this one instead of something else?
    - How is your method helping?
    - What are the results of your method? Compare the error before and after.
  - **Writeup quality**
    Some advice:
    - Try to convey a clear story giving the most relevant aspects of your approach, in a reproducible way. Learning what has not worked can additionally help the reader (and help them better understand *why* you have made the many choices you did), but focus on what is most relevant for your final solution.
    - Before the submission, have an external person proofread your report. It is easy to write a sentence that makes perfect sense to you since you wrote it but is actually hard to parse. Use a spell-checker.
    - Plots are great way to share information that might be hard to convey by writing. Make sure that your plots are understandable, have labels for axes, a title, correct axes limits, add a description of what your plot is about and what can be learned from it.

As usual, your code and report will be automatically checked for plagiarism.

Guidelines for Machine Learning Projects

Now that you have implemented few basic methods, you should use this toolbox on the dataset. Here are a few things that you might want to try.

Exploratory data analysis You should learn about your dataset - figure out which features are continuous, which ones are categorical, check if there are obvious relationships between the features, take a look at the distribution of each feature, and so on. Check `https://en.wikipedia.org/wiki/Exploratory_data_analysis`.

Feature processing Cleaning your dataset by removing useless features and values, combining others, finding better representations of the features to feed your model, scaling the features, and so on. Check this article on feature engineering: `http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/`.

Determining whether a method overfits or underfits You should be able to diagnose the whether your model is over- or underfitting the data and take actions to fix the problems with your model. Recommended reading: *Advice on applying machine learning methods* by Andrew Ng: `http://cs229.stanford.edu/materials/ML-advice.pdf`.

Applying methods and visualizing Beyond simply applying the models we have seen, it helps to try to understand what the ML model is doing. Try to find out which datapoints are wrongly classified and, if possible, why this is the case. Then use this information to improve your model. Check Peter Domingo's *Useful things to know about machine learning*: `http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf`

Accurately estimate how well your method is doing By applying cross-validation and estimating the test error.

Report Guidelines

In addition to finding a good model for the data, you will need to explain your methodology in a report.

Clearly describe your used methods, state your conclusions and argue that the results you obtained make (or do not make) sense, and the reasons behind it. Keep the report short and to the point, with a strict limit of 4 pages. References are allowed to be put on a extra page.

To get started more easily with writing the report, we provide you a LaTeX template here

github.com/epfml/ML_course/tree/master/projects/project1/latex-example-paper

The file also contains some more helpful information on how to write a scientific report or paper. We will also help you learn it during the exercise session and office hours if you ask us.

For more guidelines on what makes a good report, see the grading criteria above. In particular, don't forget to take care about

- *Reproducibility:* Not only in the code, but also in the report, do include complete details about each algorithm you tried, e.g. what lambda values you used for ridge regression? How exactly did you do that feature transformation? how many folds did you use for cross-validation? etc...
- *Baselines:* Give clear experimental evidence: When you added this new combined feature, or changed the regularization, by how much did that increase or decrease the test error? It is crucial to always report such obtained differences in the evaluation metrics, and to include several properly implemented baseline algorithms as a comparison to your approach.

Some additional resources on LaTeX:

- https://github.com/VoLuong/Begin-Latex-in-minutes - getting started with LaTeX
- http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/ - tutorial on LaTeX
- http://www.stdout.org/~winston/latex/latexsheet-a4.pdf - cheat sheet collecting most of all useful commands in LaTeX
- http://mirror.switch.ch/ftp/mirror/tex/info/first-latex-doc/first-latex-doc.pdf - example how to create a document with Latex
- http://en.wikibooks.org/wiki/LaTeX - detailed tutorial on LaTeX

Producing figures for LaTeX in Python

There are some good visualization tools in Python. "matplotlib" is probably the single most used Python package for 2D-graphics. The relevant tutorials are as follow:

- Matplotlib tutorial: http://www.labri.fr/perso/nrougier/teaching/matplotlib/
- Matplotlib tutorial: https://sites.google.com/site/scigraphs/tutorial
- Matplotlib Tutorial: http://jakevdp.github.io/mpl_tutorial/

Regarding other useful Python data visualization libraries, please refer to this blog for more information.