Projects
Machine Learning Course
Fall 2024

EPFL
School of Computer and Communication Sciences
Nicolas Flammarion & Martin Jaggi
www.epfl.ch/labs/mlo/machine-learning-cs-433

Class Project 1
Submission Deadline: Nov 1st, 2024

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 do 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 1 counts 10% to your final grade in the course. In Project 1, we will grade your code and report and you also obtain a competition score for feedback. (Project 2 will count 30%).

Logistics

Group formation. For Project 1, you will work in a team of 3 students, by your choice. If you are still searching for teammates, please use the discussion forum on Moodle. A good data science team combines a diverse set of skills, and greatly benefits from inter-disciplinary backgrounds.

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

- **Code.** In Python. For this first project, we want you to implement and use the methods we have seen in class. You need to put all code in a github classroom repository of your team, here using this github classroom invitation link. The Python libraries allowed in this project are
  - The Python standard libraries
  - `NumPy`
  - Visualization libraries (e.g. matplotlib, seaborn) but only for visualization purposes.

  No external libraries allowed! (e.g. Pandas, Scikit-Learn, PyTorch, TensorFlow, ...). External libraries will be allowed in Project 2.
- **Written Report.** You will write a maximum 2 page PDF report on your findings, using LaTeX. References are allowed to be put on a extra third page.
- **Competitive Part.** To give you immediate feedback and a fair ranking, you can use the competition platform aicrowd.com to score your predictions. You can submit whenever and almost as many times as you like, up until the final submission deadline (not graded).

The Dataset. For this course, we are providing you with our own online competition based on an interesting real dataset from the health domain - prediction of heart attacks using data of more than 300'000 individuals.

Step 1 - Getting Started

Create an account using your `epfl.ch` email and head over to the competition arena

`https://www.aicrowd.com/challenges/epfl-machine-learning-project-1`

Then, download the training dataset, available in `.csv` format. To load the data in your python script, you can use the function `load_csv_data` in `helpers.py`; remember to keep the three files `x_train.csv`, `y_train.csv` and `x_test.csv` in the same folder and do not change their names.

Step 2 - Implement ML Methods

We want you to implement and use the methods we have seen in class and in the labs. You will need to provide working implementations of the functions in Table 1. If you have not finished them during the labs, you should start by implementing the first ones to have a working toolbox before diving in the dataset.

Function	Details
`mean_squared_error_gd(y, tx, initial_w, max_iters, gamma)`	Linear regression using gradient descent
`mean_squared_error_sgd(y, tx, initial_w, max_iters, gamma)`	Linear regression using stochastic gradient descent
`least_squares(y, tx)`	Least squares regression using normal equations
`ridge_regression(y, tx, lambda_)`	Ridge regression using normal equations
`logistic_regression(y, tx, initial_w, max_iters, gamma)`	Logistic regression using gradient descent ($y \in \{0, 1\}$)
`reg_logistic_regression(y, tx, lambda_, initial_w, max_iters, gamma)`	Regularized logistic regression using gradient descent ($y \in \{0, 1\}$, with regularization term $\lambda\|w\|^2$)

Table 1: **List of functions to implement.** In the above method signatures, for iterative methods, `initial_w` is the initial weight vector, `gamma` is the step-size, and `max_iters` is the number of steps to run. `lambda_` is always the regularization parameter. (Note that here we have used the trailing underscore because `lambda` is a reserved word in Python with a different meaning). For SGD, you must use the standard mini-batch-size 1 (sample just one datapoint). The mean squared error formula has a factor 0.5 to be consistent with the lecture notes. For least squares, you are allowed to use everything from `numpy.linalg` but not `numpy.linalg.lstsq`. All vectors should be implemented as 1D arrays with shape (X,) instead of (X, 1).

You should take care of the following:

- **Return type:** Note that all functions should return: (`w`, `loss`), which is the last weight vector of the method, and the corresponding loss value (cost function). Note that while in previous labs you might have kept track of all encountered `w` for iterative methods, here we only want the last one. Moreover, the `loss` returned by the regularized methods (`ridge_regression` and `reg_logistic_regression`) should **not** include the penalty term.
- **File names:** Please provide all function implementations in a single python file, called `implementations.py`.
- All code should be easily readable and commented.
- Note that we will call your provided methods and evaluate for correct implementation. We provide some basic tests to check your implementation in `https://github.com/epfml/ML_course/tree/main/projects/project1/grading_tests`.

Step 3 - Generating Good Predictions on the Medical Dataset

The second part of this project allows you to experiment more freely with any machine learning methods, on a real medical dataset. You will experience the importance of all steps of the data science pipeline, from data cleaning, preparation, modeling choices, training to the evaluation of results, and the documentation of all you have done.

You can use any machine learning techniques of your choice, including the ones seen in the lecture so far, but you are not allowed to use any external libraries (as mentioned above).

Coding and experimenting is only one part of this project. It is at least equally important to write a convincing scientific report about your approach (the PDF deliverable). As space is limited, focus on clarity and describe the most impactful insights you found. More detailed instructions and criteria what consists in a good scientific report is provided below.

Once you have some initial model working on the data, you can send your predictions to the competition platform, to see how your approach is doing compared the other teams. You can submit anytime for a maximum of 5 times per day, until the deadline. The competition rank does *not* count to the grading of the project.

Your predictions must be in `.csv` format, see `sample-submission.csv`. You must use the same datapoint indexes as in the test set `x_test.csv`. Both, indexes and datapoints, are returned by the function `load_csv_data`. To generate a `.csv` output file from Python, use our provided helper function `create_csv_submission` in `helpers.py` (see project 1 folder on github). After a submission, aicrowd.com will compute your score on the test set, and will show you your score and ranking in the leaderboard.

This is useful to see how you compare against other teams, but you should not consider this score as the only evaluation of your model. 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 the submission system per day.

Improving your predictions. While the above described method implementations must be part of your code submission, you can now implement additional modifications of these basic methods above. You can construct better features for the task, or perform better data preprocessing for this particular dataset, or even implement an additional modification of one of the above mentioned ML methods. Note that it is not allowed to use external libraries, code or data in this project. (It will be allowed in Project 2).

Step 4 - Final Submission of Your Project

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

- **Report:** Your 2 page report as `.pdf`
  - Target audience: assume the readers are beginners in ML, but know the most basic concepts. Try to make your report as concise and self-contained as possible. Your text should aim to highlight and convey the most interesting findings you got, and at the same time allow the readers to reproduce them and put them in context as much as possible.
- **Code:** The complete executable and documented Python code, as a link to your github classroom repository. Rules for the code part:
  - *Reproducibility:* In your submission, you must provide a script `run.py` (or `run.ipynb`) which produces *exactly* the same `.csv` predictions which you used in your best submission to the competition system. If training takes a long time, this `run.py` can include only inference (loading pretrained weights and generating predictions), but then please also explain where to find the code for training these weights. If the performance in your submission is worse than the best one on competition system (due to fixing bug, randomness, etc.), then you need to mention the test performance of your submission in the report.
  - *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.
  - In addition to your customized system, don't forget that your code submission must still also include the 6 *basic method implementations* as described above in step 2.
  - *No use of external ML libraries* is allowed in Project 1. (It will be allowed in Project 2).
  - No external datasets allowed.

Submission URL: `http://mlcourse.epfl.ch`

Submission deadline: Friday Nov 1st, 2024 (16:00 afternoon)

Medical dataset background

According to World Health Organization, Cardiovascular Diseases (CVD), such as for example heart attacks, are becoming one of the leading causes of death globally. Adults are living longer, and diseases of the heart and circulatory vessels are prevalent in this growing population of older adults. The rise of new technologies such as machine learning algorithms can help with the early detection and prevention of developing CVDs. In this project, you will apply machine learning techniques to determine the risk of a person in developing CVD based on features of their personal lifestyle factors.

The data that you will use comes from the Behavioral Risk Factor Surveillance System (BRFSS), a system of health-related telephone surveys that collects state data about U.S. residents regarding their health-related risk behaviors, chronic health conditions, and use of preventive services. In particular, respondents were classified as having coronary heart disease (MICHD) if they reported having been told by a provider they had MICHD. Respondents were also classified as having MICHD if they reported having been told they had a heart attack (i.e., myocardial infarction) or angina.

In terms of early detection and prevention of these disease, it is our job to use this data to build a model able to estimate the likelihood of developing MICHD given a certain clinical and lifestyle situation. In practice, this means that you will be given a vector of features collecting the health-related data of a person, and asked to predict whether this situation leads to a MICHD or not. To do this, you will use the binary classification techniques we have discussed in the lectures.

If you're interested in more background on this dataset, we point you to the longer description here:
`https://www.cdc.gov/brfss/annual_data/annual_2015.html`.

Note that understanding the medical background is not necessary to perform well in this machine learning challenge as part of the course.

Appendix

Grading Criteria

- **Code (counts 40%).** In Python. *No external libraries allowed!* For this first project, we want you to implement and use the methods we have seen in class. The code will be graded by two assistants independently, according to the criteria described above in Step 4. Note that assistants will setup auto-grading scripts to import 6 functions in Table 1 from your `implementations.py` and verify their correctness by checking the outputs for some given inputs.
- **Written Report (counts 60%).** You will write a maximum 2 page PDF report on your findings, using LaTeX. The code will be graded by two assistants independently, and we will provide you feedback. The main criteria will be if you were able to correctly implement the methods seen in class and explain your approach. This counts half for the written report. In addition, we will grade you on the scientific contribution you made additionally, to improve your predictions. For this part, the criteria are
  - scientific novelty
  - creativity
  - reproducibility
  - solid comparison baselines supporting your claims (including e.g. an ablation study which of a set of modifications did affect performance the most)
  - writeup quality and clarity
- **Competitive Part (counts only for fun).** The final rank of your team in the (private) leaderboard is a strong indication how well you dealt with this practical ML problem. It allows you to experience the practical importance of each parts, the data cleaning, preprocessing, the ML model, and for example hyper-parameter optimization.

  The competitive part will count in Project 2, but only for the standard tasks we provide. Here in this first toy project, as long as you're not very far worse than all other teams, don't spend too much time optimizing the rank yet. But make sure you can use the submission system and see the impact of your changes. Most important is that you understand the correct implementation of the algorithms and how their fair practical evaluation works, and thereby to prepare you for Project 2.

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 generalization error. Among the (potentially many) modifications you did, show an ablation study of the most important ones: Which changes had the largest impact on your final performance? (For choices of models, training algorithms, data-preprocessing, hyperparameters etc).

Report Guidelines

In addition to finding a good model for the data, you will need to explain your methodology in a report. For the first project, this will help you getting used to writing, and prepare you for the more extensive Project 2.

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 2 pages (Project 2 will allow 4 pages, plus an appendix). No appendix allowed. References are allowed to be put on a extra third page.

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

github.com/epfml/ML_course/tree/main/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.

Longer article on what are good practices in writing a scientific report in a data science, computing or ML context:

- `http://arxiv.org/pdf/1609.00037`
- or an older article `http://arxiv.org/pdf/1210.0530`.

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://en.wikibooks.org/wiki/LaTeX` - detailed tutorial on LaTeX

Producing figures for LaTeX in Python

When making figures and plots, make sure the reader understands what the axes mean, what units and data is being visualized (see also the reproducibility criterion). 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: `http://jakevdp.github.io/mpl_tutorial/`

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