Machine Learning (HWS24)
Assignment 2: Logistic Regression

The archive provided to you contains this assignment description, a datasets in binary format, as well as Python code fragments for you to complete. Comments and documentation in the code provide further information. Please note the following:

- It **suffices to fill out the "holes" that are marked in the code fragments** provided to you, but feel free to modify the code to your liking. The aim of the assignments is to familiarize you with NumPy, so use NumPy over pure Python and try to keep your implementation efficient.
- Please **adhere to the following guidelines** in all the assignments. If you do not follow those guidelines, we may grade your solution as a FAIL. Provide a single ZIP archive with name `ml24-a0<assignment number>-<your ILIAS login>.zip`. The archive needs to contain:
  - A **single PDF report** that contains answers to the tasks specified in the assignment, including helpful figures and a high-level description of your approach. **Do not simply convert your Jupyter notebook to a PDF!** Write a separate document, stay focused and brief. You report **must not exceed 8 pages, excluding references and appendix**. You must use the seminar template from this link for your report (including the final signature page).
  - All **the code that you created** and used in its original format.
  - A **PDF document that renders your Jupyter notebook with all figures.** (If you don't use Jupyter, then you obviously do not need to provide this.)
- **A high quality report is required to achieve an EXCELLENT grade.** Such a report is self-explanatory (i.e. do not refer to your code except for implementation-only tasks), follows good scientific practice (e.g. when using images, tables or citations), does not include hand-written notes, and does not exceed 8 pages. In addition, label all figures (and refer to figure labels in your write-up), include references if you used additional sources or material, and use the tasks numbers of the assignments as your section and subsection numbers.
- You **may work on this assignment in pairs**—i.e, with one (and only one) additional student—and then hand in a pair submission. To do so:
  - The PDF report and notebook must clearly report then **name and ILIAS login** of **both students** right at the beginning.
  - **Both students must submit** the same assignment on ILIAS separately.
  - You may change whether or not you submit in a pair and with whom from assignment to assignment.

  To be clear, if only one student of the pair submits the assignment, then only this student will receive the grade.
- Hand-in your solution via ILIAS until the date specified there. **This is a hard deadline**.

Report. The following list of tasks are pure implementation tasks:

Task 1b, Task 2b–d, Task 4a.

You do *not* need to discuss these tasks in your report.

Preliminaries: Spambase Dataset

In this assignment, we are using a dataset on email spam detection, provided in the file `data/spamData.mat`. You can find more information about this dataset at https://archive.ics.uci.edu/ml/datasets/spambase.

The dataset is derived from 4601 emails, each being labeled as no-spam (0) or spam (1). Each example has 57 features:

- 48 word features, each indicating the frequency percentage of a word in the email (e.g., "business", "free", "george").
- 6 character features, each indicating the frequency percentage of a character in the email (for `{[}!$#`).
- 3 features on length statistics of consecutive upper case letters (e.g., "HELLO" gives 5), one for minimum length, one for maximum length, and one for the sum of the lengths.

The dataset is split into a training set (3065 examples) and a test set (1536 examples). We provide code to load the data and provide all feature names.

1 Dataset Statistics

Explore and preprocess the dataset.

a) Look at the kernel density plot (code provided) of all features and discuss what you see (or don't see).

b) Normalize the data using z-scores, i.e., normalize each feature to mean 0 and variance 1. Normalize both training and test data. In particular, think about how test data should be normalized.

Make sure to stick to the variable names provided in the code fragements. From now on, **we will exclusively work with the normalized data**.

c) Redo the kernel density plot on the normalized data. What changed? Is there anything that "sticks out"?

2 Maximum Likelihood Estimation

a) Show analytically that if we use a bias term, rescaling (multiply by constant) and shifting (add a constant) features leads to ML estimates with the same likelihood. Why do you think we computed z-scores then?

b) Complete the methods for computing the log-likelihood and gradient of the log-likelihood for logistic regression. We **do not use a bias term** throughout.

c) Implement gradient descent using the framework provided to you.

Optional. Can you implement each gradient descent epoch using only vectorized operations (no loops)?

d) Implement stochastic gradient descent.

e) Explore the behavior of both methods for the parameters provided to you. Discuss!

3 Prediction

Complete the `predict` and `classify` methods for the predicted spam probability and predicted class label, respectively. Explore the models that you fit in the previous task and discuss. Study the composition of the weight vector: which features are important, which are not? Is this intuitive?

4 Maximum Aposteriori Estimation

a) Implement gradient descent for MAP estimation of logistic regression with a Gaussian prior / L2 regularization with hyperparameter λ. You can reuse the methods of your solution for MLE.

b) Study the effect of the prior on the result by varying the value of λ. Consider at least the training data log-likelihood, the test data log-likelihood, and the prediction accuracy. Are these results surprising to you?

c) Study the composition of the weight vector for varying choices of λ (try very large values). Try to explain what you saw in the task above.

5 Optional: Exploration

Explore variants of preprocessing and logistic regression further. Suggestions include:

- Try gradient descent on the original data without using z-scores.
- Add a bias feature (make sure that you do not scale it).
- Try to reduce the training set size and compare MLE and MAP estimation (ideally using cross-validation).
- Run a logistic regression method from some existing library. Do you get the same results?
- Experiment with different gradient-based optimizers. To do so, we provide a PyTorch implementation, which allows you to quickly change the optimizer. For a list of optimizers, see here.