LAOML 2023–2024 PROJECT 1

The goal of this project is to practice implementing your own machine learning optimization routines and play around with different approaches/parameters to see what works well in practice for certain datasets. For this reason, you should not use functions from machine learning packages such as scikit-learn. For your implementations, please use Python with basic functions from packages such as NumPy and SciPy, or Julia with basic packages such as LinearAlgebra. Please use the same programming language for the entire project. If you prefer, you can use another open-source language, but please consult with a lecturer first.

The project is designed for groups of three. Please collaborate on each question (as opposed to dividing the work). Write an informal report where you discuss the questions below (please use a section for each question). Of course, the algorithms mentioned in this project can be found online or can be generated using large language models. Please write your report in such a way that it becomes clear to the reader that you wrote your own version of the code and that you experimented with it yourself. For instance, when answering a question you can give different versions of the same function, where you write things such as "We found that this version of the function was very slow due to the dense matrix operations; each iteration took 12 seconds. The following version of the function uses sparse linear algebra, where the iterations only take 0.8 seconds". Also, whenever there are hyperparameters, discuss your experiences with tuning these. How did this affect the performance on the training and test sets, or how did it affect the runtime? In your answers discuss whether or not the results you see agree with your expectations.

Please hand in your report as a PDF file, including the code in easy-to-read listings, or as a Jupyter notebook. In any case, please also submit an additional text file containing a copy of all the code in your report. Use Brightspace to hand in the files. The hand-in date for this project is 12 November 2024.

1. (3 points) Download the data.csv file. Each data point corresponds to an Android application which is malicious or not. The features are various properties of the applications (such as which permissions it requests, etc). The last column corresponds to the vector y and the remainder of the columns to the data matrix X. Write a function that loads this file and returns the matrix X and vector y. You can use an existing function/library to load CSV files if you want. Note that the entries of y should be ± 1-valued. How many malicious data points are there? What can you say about the sparsity of the data? Do you think it makes sense to use one-hot-coding for some of the columns?

2. (2 points) Write a function that splits the data into a training and test set according to some fraction $0 < r < 1$. Make sure to use randomization; that is, it should not be the case that the training set consists of the first data points and the test set of the remaining data points. Your function should return matrices X_{train} and X_{test} and vectors y_{train} and y_{test}.

3. (2 points) Write a function that, given the matrix X, the vector y, and a weight vector w defining a hyperplane, returns the number of correctly classified points. Verify that the output makes sense for random weight vectors.

4. (2 points) Consider the cost function for logistic regression as defined in the lectures. Write down a symbolic formula for the gradient of this function.

5. (10 points) Write a straightforward implementation for logistic regression using gradient descent with a fixed step size α. Your function should take as arguments the data matrix X and data vector y, the step size α, the regularization constant λ, and an integer K indicating the number of gradient descent steps. The function should return a weight vector w.

   Experiment with the hyperparameters, using dense and sparse linear algebra, on random splits of training and test data sets. (If you know about writing allocation-free code, you can also experiment with this.)

   Given for instance a 50/50 split between test and training data, what is the best classification performance you can obtain on the test set?

   Note that with a good implementation, this should be fast even on an old laptop.

6. (7 points) Download the file data2.csv. This is the same as the previous data file, except that 2000 fake data points have been appended to the data set. Use the singular value transform as explained in the lecture to detect and remove most of these outliers without removing too many other data points. (For the singular value transform you can use a library function, you do not have to implement this yourself.)