WI4630 Statistical Learning – Assignment 2

March 18, 2025

For this assignment create a small report (in PDF format) consisting of the required code output and discussion of results and design choices. Also hand in your code (as a Python file) with the assignment and also provide all code in the appendix of your report.

Question 1

First, we will estimate the logistic regression model for simulated data in the low-dimensional setting. In the python script `A2_logistic_regression_simulation.py` we simulate both features and labels such that we obtain $\boldsymbol{x}_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$ for $i = 1, \cdots, 1000$. Even though the features are simulated, you may assume that they are fixed, i.e., we are in the standard setting of the logistic regression model.

(a) Write a Python function that computes the maximum likelihood estimator for $\boldsymbol{\beta} = \left[\beta_1, \beta_2\right]^T$ using the Newton-Raphson algorithm (see Chapter 4.4 of Hastie et al. for a detailed description). You may use the skeleton of this function that is given in `A2_logistic_regression_simulation.py`. All required matrix algebra can be done with the `NumPy` library.

In order to gain more insight regarding the performance of the maximum likelihood estimator in this setting we can set up a Monte Carlo experiment. A Monte Carlo experiment consists of simulating data S times (a large amount) according to our model and computing the maximum likelihood estimator for each of these simulated samples.

Given our features $(\boldsymbol{x}_i)_{i=1}^n$ we simulate the targets, also called labels, $(y_i)_{i=1}^n$ according to the logistic model with parameter $\boldsymbol{\beta}_* = \left[0.2, -0.8\right]^T$; this is done in the function `logistic_simulation` in the python script. For each simulation of the labels, we compute the maximum likelihood estimator (MLE).

(b) Compute the mean of the MLE for both parameters over $S = 1000$ simulations. Also plot a histogram for the MLE of both parameters. Repeat the Monte Carlo experiment with a sample size of $n = 100$ instead of $n = 1000$. Discuss your findings.

In the remainder of this exercise we will consider the famous MNIST dataset. This dataset contains a large number of 28×28 pixel, grayscale images of handwritten digits. The dataset is available in the file `mnist.csv`. In the python script `A2_logistic_regression_mnist.py` it is shown how to read the data from the csv file into Python. Since we are only studying binary classification we will only be concerned with the zeros and ones appearing in the dataset. The images are labeled, i.e., the number they depict is present in the dataset as well. Hence our targets $y_i \in \{0, 1\}$ denote the given labels and the features $\boldsymbol{x}_i \in \mathbb{R}^{784}$ consist of $28 \times 28 = 784$ values between 0 and 255, which are the gray-scale values of the pixels in the image. The value 0 corresponds to black and the value 255 corresponds to white.

In practice we cannot simulate data multiple times to evaluate the performance of our estimator. Therefore, machine learning practitioners split the dataset in a so-called training and test set. The model is estimated using only the training set and the predictive performance of the model is evaluated using the test set. We will consider a training size of $n = 100$. The selecting of the zeros and ones of the full MNIST dataset and the splitting of the data into a training and test set has already been done in the Python script. Moreover, also the functions `logistic_forecast` and `prediction_accuracy` are given and can be used.

(c) Run the Newton-Raphson algorithm that you have written in part (a). Python will display an error message due to a matrix singularity. Compute the rank of the matrix of features X, which is called `x_train` in the python script. Explain what is going wrong.

(d) In order to solve the issue arising in (c), we add a ridge penalty to the log-likelihood criterion:

$$J(\boldsymbol{\beta}) = -\ell(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{1}$$

where we take $\lambda = 1$ and $\ell(\boldsymbol{\beta})$ denotes the log-likelihood function of the logistic regression model. Adapt the Newton-Raphson algorithm such that we obtain the regularised estimator of $\boldsymbol{\beta}$ that minimizes $J(\boldsymbol{\beta})$. Evaluate the predictive performance of this estimator using the function `prediction_accuracy`. Discuss your findings.

Question 2

In this exercise we will work with the `CIFAR-10` data set to carry out classification of color images. The data set consists of 50,000 colour images with an additional 10,000 test images which are each classified in one of ten possible categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). For background information on the `CIFAR-10` data set, and a link to download the data, see `https://www.cs.toronto.edu/~kriz/cifar.html`.

For this classification problem we will make use of the Python package Scikit-Learn, so make sure to install this package on your computer and consult the online documentation when necessary.

(a) Write code to load your data set in a matrix `X` of size $n \times p$ and a response vector `y` of length n. Separate the data in a training set and a test set of appropriate sizes. You may use the file `A2_cifar10.py` (requiring Python version ≥ 3) to make this easy. If it is working correctly then you should be able to plot images from the data set.

(b) Use the function `linear_model.LogisticRegression` of the Python package Scikit-Learn to fit a regularized logistic regression problem to your data. You will need to experiment with the regularization parameter to ensure that the optimization algorithm converges. *Note:* In Scikit-Learn, the amount of regularization is indicated by `C`, which should interpreted as the reciprocal of λ as used in the course.

The package Scikit-Learn allows for automatically fitting a logistic regression model in combination with cross-validation. In its basic version, it relies on the accuracy score.

(c) Use the function `linear_model.LogisticRegressionCV` to find a good choice of the regularization parameter using cross-validation with $K = 4$ folds. One of your jobs is to choose a suitable range of regularization parameters to try out. Make a log-log plot of the cross-validated accuracy score as a function of the regularization parameter.

Unfortunately, the accuracy score that is used by default in Scikit-Learn only takes into account the actual prediction, which for logistic regression correspond to the class with maximum estimated probability. However we would like to take into account the quality of our probabilistic prediction. For this we would like to use a logarithmic scoring rule.

(d) Write some code to apply cross validation to the logistic regression problem with a logarithmic scoring rule. Again make a log-log plot of the cross-validation score as a function of the value of the regularization parameter. Do the answers to (c) and (d) agree in terms of the optimal regularization parameter? *Hint:* Some useful functions may be `model_selection.cross_val_predict` and `metrics.log_loss`.

(e) Based on parts (c) and (d), choose one 'optimal' value of the regularization parameter; argue why you believe this is a good choice. For this value of `C`, fit the logistic regression model using the full training data set. Report the training error and error on a test set, both for accuracy score and logistic score, and compare to the results of cross-validation. (If you are curious, you can also determine train and test error for the same range of values as you used for parts (c) and (d).)