

Practical work 06 – 23rd of October 2018

Logistic Regression

Summary for the organisation :

- Submit the solutions of the practical work before Monday 12h00 next week in Moodle.
- Preferred modality : **one**¹ pdf generated from your iPython notebook(s) and an archive with your notebook(s).
- Alternative modality : one pdf report with annotated code and outputs.
- The file name must contain the number of the practical work, followed by the names of the team members by alphabetical order, for example 02_dupont_muller_smith.zip.
- Put also the name of the team members in the body of the notebook (or report).
- Only one submission per team.

Exercise 1 Classification to predict student admission

This is a continuation of the exercise of previous week, where the objective is to build a classification system to predict whether a student gets admitted into a university or not based on their results on two exams². For each training example n , you have the applicant's scores on two exams $(x_{n,1}, x_{n,2})$ and the admissions decision y_n . The dataset is illustrated on Figure 1.

a. Logistic regression classifier with linear decision boundary

The objective is to implement a classifier based on a logistic regression approach :

$$h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- a) In a similar way as for the exercise of the previous week, read the training data from file `student-dataset-train.csv`. The first two columns are x_1 and x_2 . The last column holds the class label y . Build the design matrix X as follow :

1. And only one so use a tool to merge your pdfs if more than one.
2. Data source : Andrew Ng - Machine Learning class Stanford

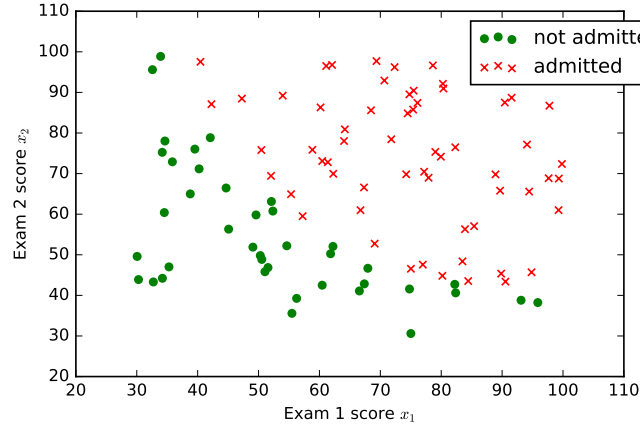


FIGURE 1 – Training data

$$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} \\ 1 & \vdots & \vdots \\ 1 & x_{N,1} & x_{N,2} \end{pmatrix} \quad (1)$$

Check that the shape of X is (100,3) and that the shape of y is (100,).

- b) Implement a z-norm normalization of the training set. You need to store the normalization values (μ, σ) for later as they will be needed to normalize the test set.
- c) Implement a sigmoid function $g(z) = \frac{1}{1+e^{-z}}$. Use numpy to compute the exp so that your function can take numpy arrays as input. Check your implementation by plotting the sigmoid function.
- d) Implement the hypothesis function $h_{\theta}(\mathbf{x})$. Hint : implement it so that the computation can take the full array X with $h(\mathbf{x})$ broadcasted to all training samples.
- e) Implement the objective function $J(\theta)$:

$$J(\theta) = \frac{1}{N} \sum_{n=1}^N y_n \log h_{\theta}(\mathbf{x}_n) + (1 - y_n) \log(1 - h_{\theta}(\mathbf{x}_n))$$

- f) In a similar way as in PW02 and PW03, implement the gradient ascent with the update rule :

$$\theta_i \leftarrow \theta_i + \alpha \frac{1}{N} \sum_{n=1}^N (y_n - h_{\theta}(\mathbf{x}_n)) x_{n,i}$$

- g) Test your implementation by running a gradient ascent. Hints : use a small α . e.g. 0.001, store the evolution of the objective function $J(\theta)$ during the epochs to make a plot, use a large number of epochs, e.g. 2000000.

- h) Compute the correct classification rate on `student-dataset-test.csv` after convergence as you have an estimator of the posterior probabilities with

$$P(y_n = 1 | \mathbf{x}_n; \theta) = h_\theta(\mathbf{x}_n) \\ P(y_n = 0 | \mathbf{x}_n; \theta) = 1 - h_\theta(\mathbf{x}_n)$$

This means that you can take the decisions $\hat{y}_n = 1$ if $h_\theta(\mathbf{x}_n) \geq 0.5$ and $\hat{y}_n = 0$ if $h_\theta(\mathbf{x}_n) < 0.5$.

- i) Draw the decision boundary of your system on top of the scatter plot of the testing data.
j) Compare the performance of the logistic regression system with the ones of previous's week.

b. Optional - Stochastic gradient ascent

Redo the experiments of 2.a with a stochastic gradient ascent.

c. Logistic regression classifier with non-linear decision boundary

Redo the experiments of 2.a by increasing the complexity of the model in order to have a non-linear decision boundary :

$$h_\theta(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \dots)$$

d. Using SciKit Learn

Redo one of the exercise a. or c. using SciKit Learn.

- a) Read the documentation of the function `SGDClassifier()` available in the toolkit SciKit Learn. This function implements stochastic gradient descent training for different linear systems such as Logistic Regression. For a logistic regression, the `loss` parameter should be set to `'log'`.
b) Use the `fit()` and `predict()` methods of this classifier on the student data.
c) Compute the performances and compare it to your own implementations.

Exercise 2 Optional - Classification on MNIST

Using the SciKit Learn toolkit, train a multi-class logistic regression on the MNIST problem and compare the performances with the KNN of PW2.

For the “daring” ones, implement a multi-class version based on your own implementation and run it against MNIST data.