IMC-4302C – Statistical learning
2.1. Classification : Logistic Regression

Kevin Zagalo
<kevin.zagalo@inria.fr>

January 13, 2020

Use a `Jupyter` notebook and send it to the depository before the next session. Any notebook sent after that won't be considered. Thank you !

Exercise 1

From the definition of the likelihood, show that maximizing the likelihood of the logistic function is equivalent to minimizing the function

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{m} [y_i \times \log(h_\theta(x_i)) + (1 - y_i) \times \log(1 - h_\theta(x_i))]$$

$\blacksquare$

Exercise 2 *Student scores*

In this exercise, you will train a logistic regression classifier using gradient descent method in a first time then a python optimization function. After that your classifier should predict if student will be admitted or not given his score in two main exams.

The exams.txt file contains 3 columns that represent the exam 1 and exam 2 scores and the result of 100 students (0: Not admitted, 1: Admitted).

Question 1

Open this file with a file editor to understand more the data. Load the data in `students_results` variable and check its size.

Question 2

Determine the number of student m from the input data `students_results`. Extract exam 1, exam 2 scores and the result columns respectively in x_1, x_2 and y. Determine the number of features n (number of columns of array X).

Question 3

Implement the `sigmoid` function

$$S(z) = \left[1 + e^{-z}\right]^{-1}$$

Implement the `MSE_cost_func` function that evaluate and return the mean squared error or the logistic function according to the given equation (make a vectorized implementation).

Question 4

Implement the `cost_func` function that evaluates and returns the logistic cost function.

Question 5

Implement the `grad_cost_func` function that evaluates the gradient of logistic cost function at the point θ considering the j^{th} component as given on the previous equation. Implement the update equation of the gradient descent algorithm

$$\theta = \theta + \alpha \nabla L(\theta)$$

Question 6

Set the learning rate α to the best value that helps to decrease quickly the cost function.

Question 7

From the learning curves the cost function seems to decrease more. Hence, try to run the gradient descent with 1000000 (1 million) iterations and describe the difference between two cases. Use the optimal theta calculated to predict the result of student who has a score of 11 in exam 1 and a score of 9.5 in exam 2.

Question 8

We note that the gradient descent algorithm takes a lot of time and it is not suitable for optimizing complex function like logistic cost function. Hence, we will use `fmin_bfgs` optimization function from the `scipy` library in python.

– The `fmin_bfgs` function work with theta and gradient array in form $(n,)$ and not $(n,1)$. Re-implement cost function (`cost_func2`) and gradient function (`grad_cost_func2`) that deal with this kind of arrays.
– Call `fmin_bfgs` function to calculate the optimal θ. This function take as parameters `cost_func2`, `grad_cost_func2` and the initial theta $theta_0$.

Hint: You could use `reshape` function to modify the shape of the gradient vector

■

Exercise 3 *Microchip testing*

The microchip data set contains 3 columns. The scores result of two test process on a manufactured microchips is presented in the 2 first columns. While the third column indicates if the corresponding microchip were accepted or rejected.

Question 1

Load data from microship.txt file and extract each column.

Question 2

Implement the `Poly_Features` function that concatenate to data array the different possible power (below deg) of feature vector f1 and f2 as shown below:

$$data = [data,\ f_1,\ f_1^2,\ \ldots,\ f_1^{deg},\ f_2,\ f_2^2,\ \ldots,\ f_2^{deg}]$$

Question 3

Same questions as Exercice 2.

■