TC3020 Machine Learning
Teamwork Assignment 2: Logistic Regression and Regularization

Dr. Rafael Pérez Torres

September 10, 2021

Abstract

In this assignment, you will implement the Logistic Regression (with regularization) and Gradient Descent algorithms.

1 Description

Overall goal: Implement the Logistic Regression (LR) classifier with regularization and the Gradient Descent algorithms studied during our previous sessions.

You must not employ specialized libraries (`sklearn` or similar) to produce your solution; it is ok to employ them to compare though.

You have access to a partially implemented source code for a multivariate LR and Gradient Descent algorithms. Take your time to study the code and understand its structure. There are four main elements:

- A launcher (`launcher.py`) that creates some instances of the LR and runs them. It defines the hyperparameters learning rate (α), regularization (λ), and epochs.
- A LogisticRegressor class (`LogisticRegressor.py`) that implements the core of the LR algorithm. **You will be working in this file**.
- A utils script (`utils.py`) that defines some functions to read the dataset, plot results, generate polynomial features, etc.

2 Implementation

- Locate the TODO comments in the `LogisticRegressor.py` file and implement the required functionality.
- Pay attention to the dimensions of the parameters employed in the functions, so that you can be sure your implementation is returning arrays/values with the proper dimensions..
- Debug as much as you can to understand the code flow and identify issues.

See that in the `launcher.py` we are running two LogisticRegressors per dataset, one without and one with regularization. Play around with the values of hyperparameters and see how the LogisticRegressor varies its output.

If you don't modify the seed used by `np.random` and the hyperparameter values (`epochs = 50000, alpha=0.001`) you shall see the following values as outputs:

- First dataset:
  - $\theta = [[-2.23443291, 0.02623414, 0.01792713]]$
  - $J(\theta) = 49.50509418925434$.
- Second dataset (regularization enabled as $\lambda = 0.1$)
  - $\theta =$

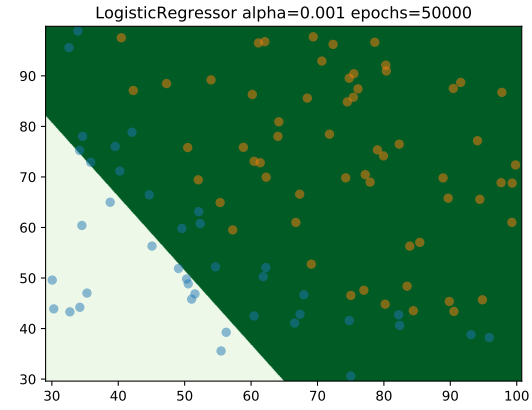
    - −0.23260776
    - 3.72297944
    - 1.54614923
    - 0.21677911
    - −1.56118956
    - 2.23697168
    - −1.56982431
    - 7.7289922
    - 9.13861257
    - −2.78723421
    - 5.38284744
    - 0.5717635
    - 1.20135082
    - 8.4967851
    - −9.10697916
    - −8.51155312
    - −9.644058
    - 6.59898662
    - 5.49622672
    - 7.34858681
    - 9.1424016
    - 5.64982499
    - −0.77627312
    - 5.53918537
    - −7.63749169
    - 2.71626902
    - −7.13810763
    - 8.42913721
  - $J(\theta) = 357.5812870032305$.

The predictions for the samples with index $[1, 70, 110]$ shall be [1 1 0].
The companion plots would be as shown in Fig. 1.

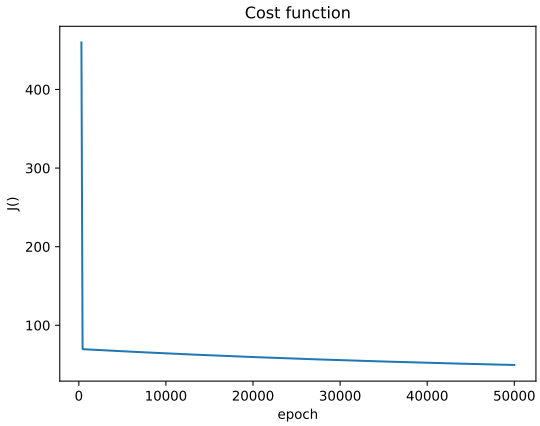
3 Further work

Prepare a short report (3 text pages top, could be more with cover, figures), where you present your findings when playing around with values for the hyperparameters.

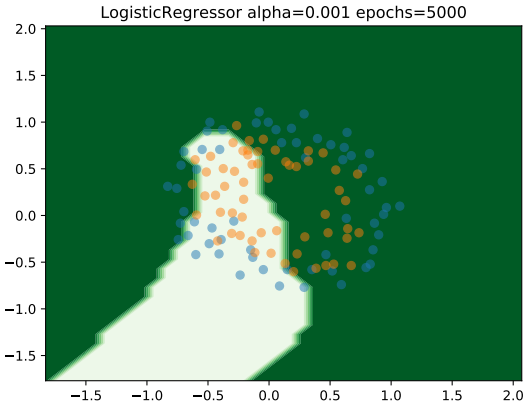
- You must elaborate on what happens when you set the alpha value (learning rate) large vs small, as well as with the regularization factor.
- Cases where the code fails (if any).
- The best combination of values you can find
- The impact of changing the number of iterations.
- How GD is influenced when regularization over the input dataset features (second dataset) is/is not performed.



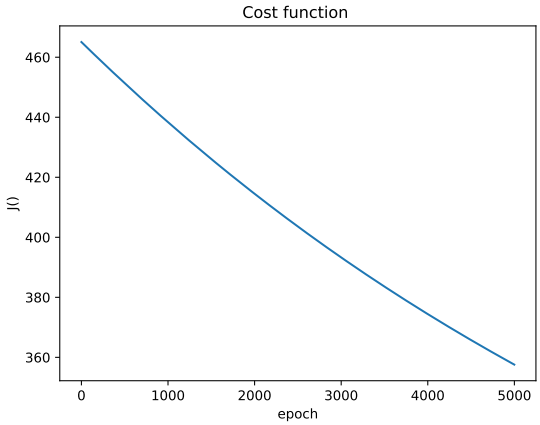
(a) LogisticRegressor decision boundary in first dataset



(b) LogisticRegressor cost function in first dataset



(c) LogisticRegressor decision boundary in second dataset



(d) LogisticRegressor cost function in second dataset

Figure 1: Expected results for the LR implementation

4 The datasets

You will be using two datasets. One dataset includes the marks for two exams and whether the student with such marks was accepted to a University (see Fig.2). The goal can be seen as to predict whether a student with two given marks would be accepted at the University. *Remember that we need to add a feature* $x_0 = 1$ *consisting of values of* 1 *per sample, this is already done in* `launcher.py`.

The second dataset is to implement Logistic Regression with Regularization. This dataset refers to a set of tests run to circuits made in a factory that verify whether a given circuit should be accepted. In Fig. 3 we can see the original appearance of the dataset.

Because we are using Logistic Regression, a single straight line is not going to be enough to separate our dataset. Hence, we rely on feature engineering to add combinations of the same features to create a new dataset (the actual combination involves all polinomial terms of input features up to the 6^{th} power). This new dataset has more features (28) that might help to separate the classes, but it could be prone to overfitting. As a result, for training this dataset you need to implement regularization in your cost function. The new transformed dataset is provided to you, so you can proceed to implement the LR and GD with support for regularization. *See that the* $x_0 = 1$ *feature has been already added for you directly in the dataset.*

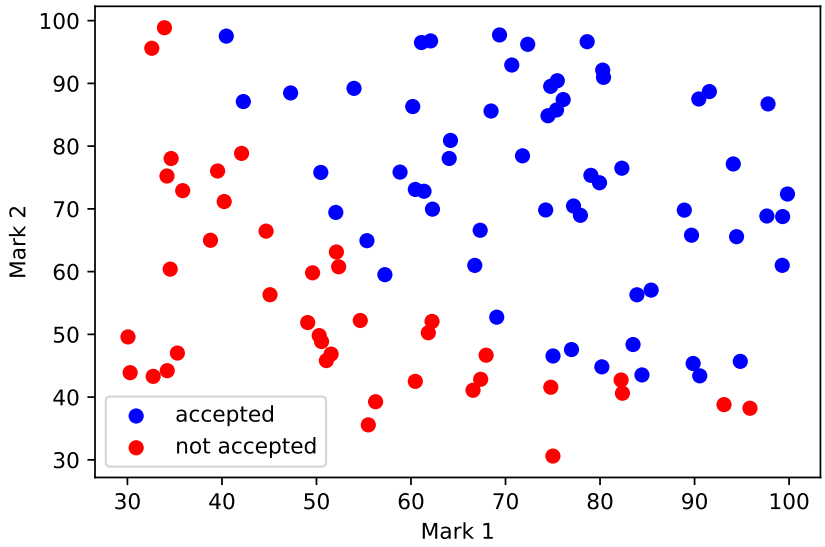


Figure 2: Marks and acceptance status in the dataset.

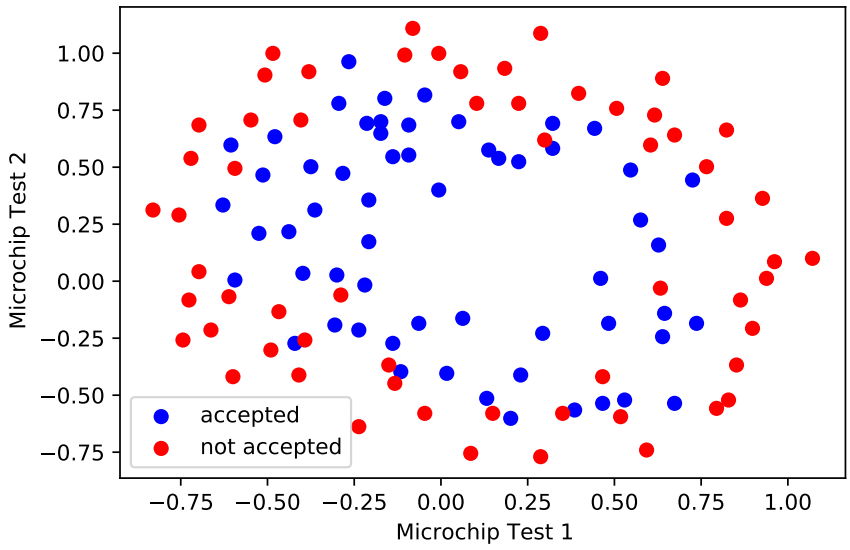


Figure 3: Tests on microchips and their acceptance in the QA process.

5 Notes

- Your code should be ready to be run by just executing the `launcher.py` script.
- Include everything (even datasets) in your submitted code so that it can be executed straightforwardly.
- **It will be penalized if your code can't be run directly.**

6 Deliverables

- Your implementation as a zip file with the source code.
  - Name your file as `A1_A2_A3-HW-02.zip` where A1, A2, A3 are the reg number (matrícula) of students.
  - Include the names of team members and student numbers as comments.
- Your report as a PDF.

A single submission per team is enough, there is no need for submitting more than once.