Tecnológico de Monterrey

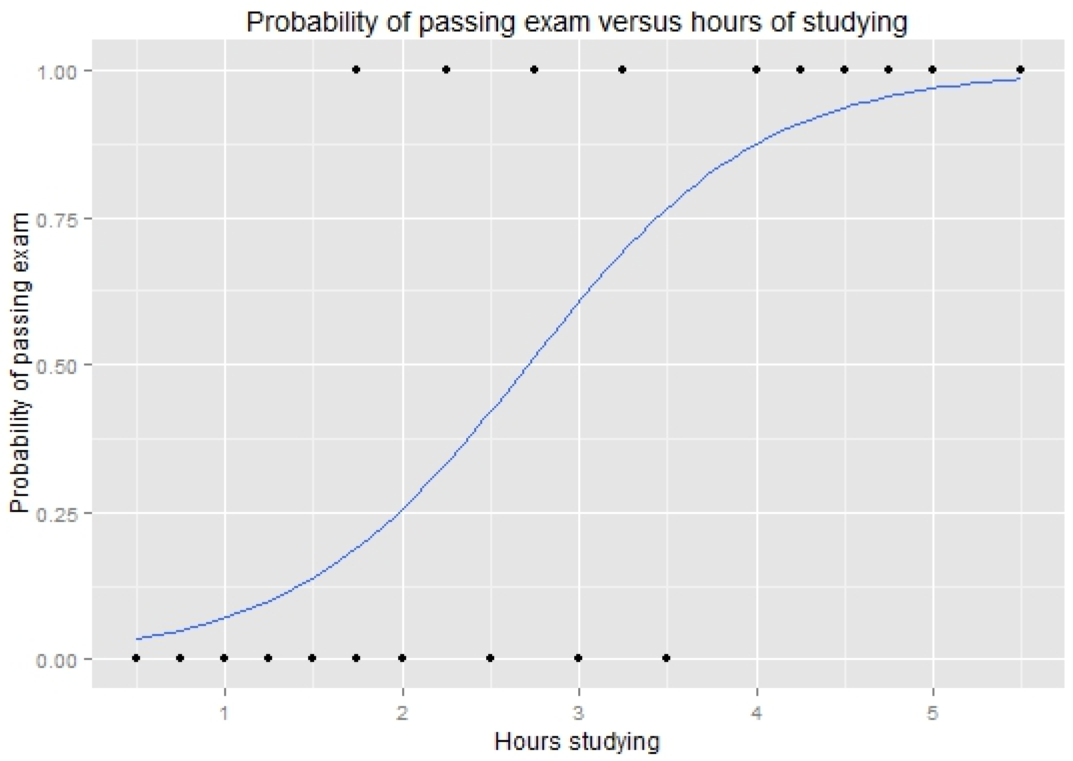
TC3020 Machine Learning (Gpo 2)

Prof. Rafael Pérez Torres
Team #04

Luis Felipe Álvarez Sánchez	A01194173
Jesús Omar Cuenca Espino	A01378844
Juan José González Andrews	A01194101
Rodrigo Montemayor Faudoa	A00821976

Description

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from the logistic unit, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.



Experimentation

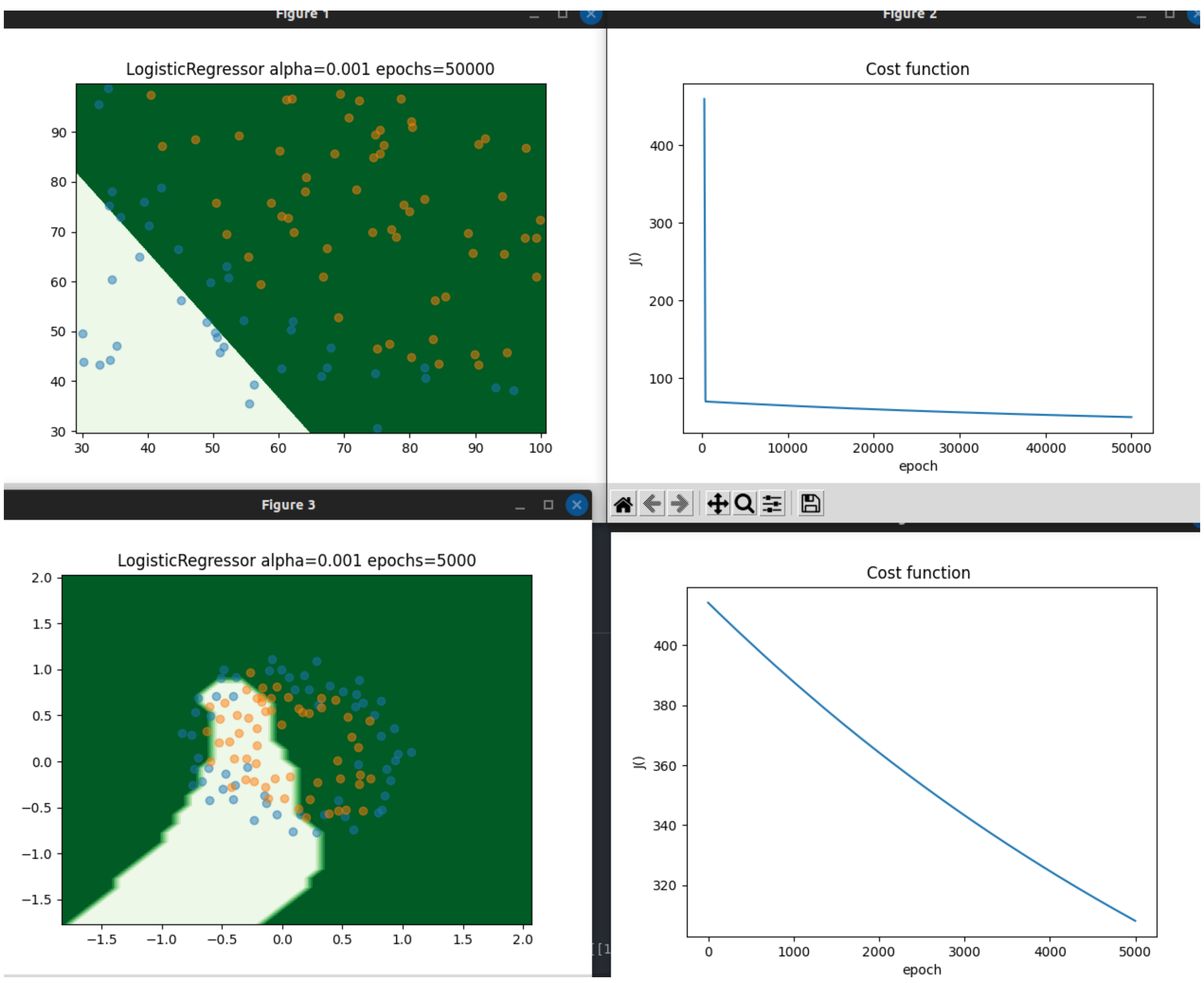


Fig.1 Grupo de control

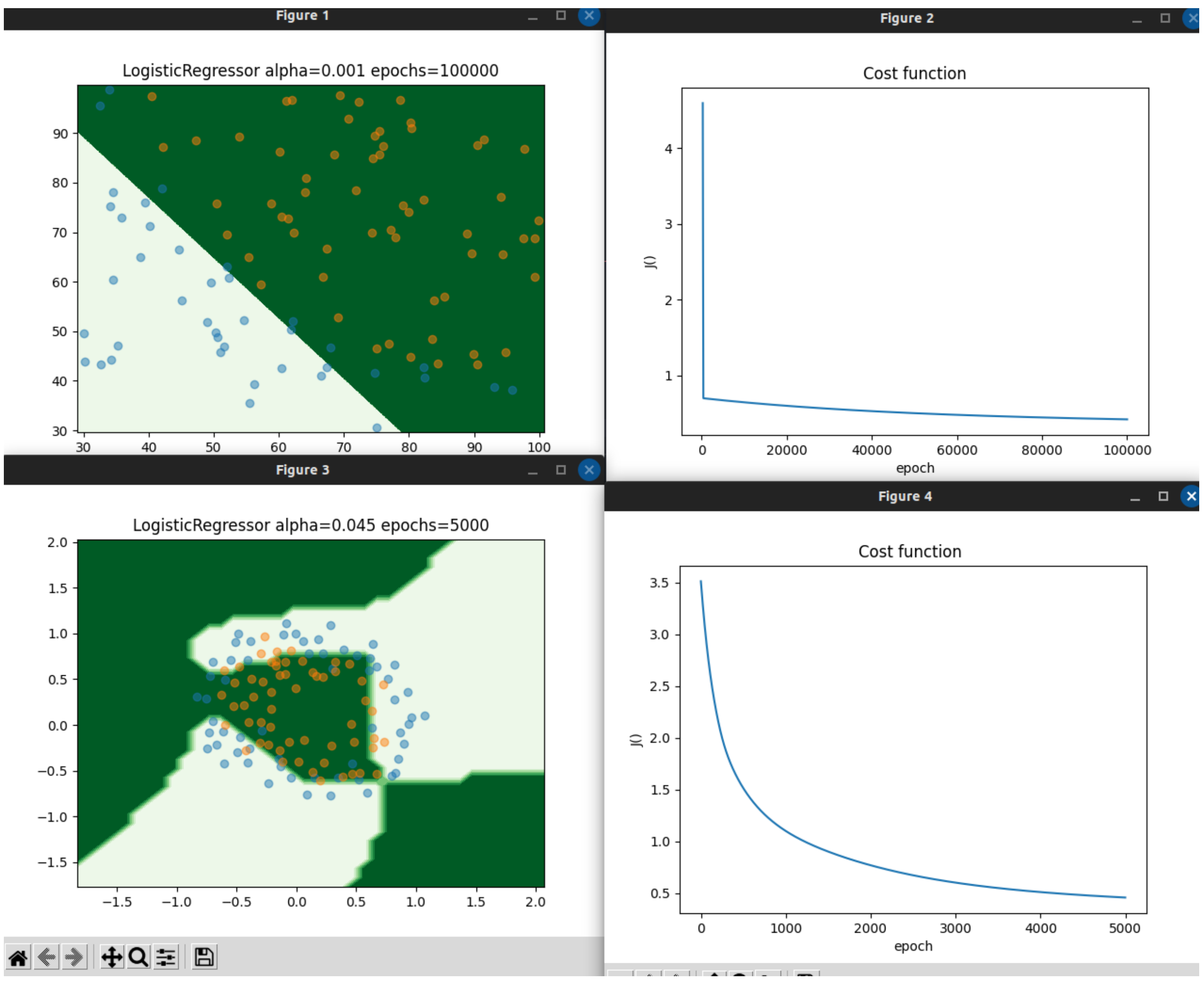
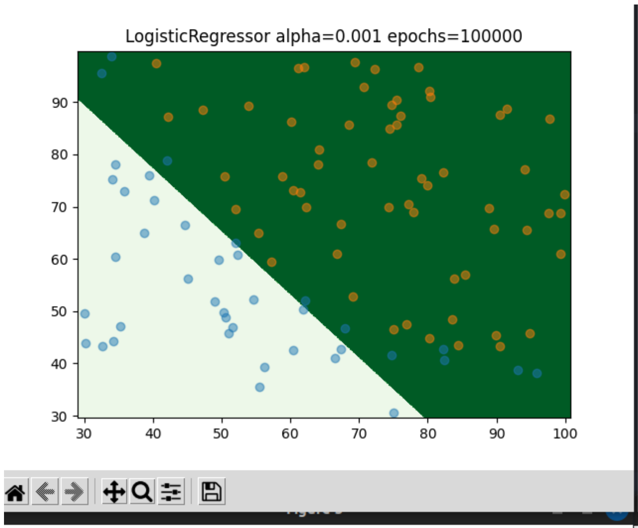
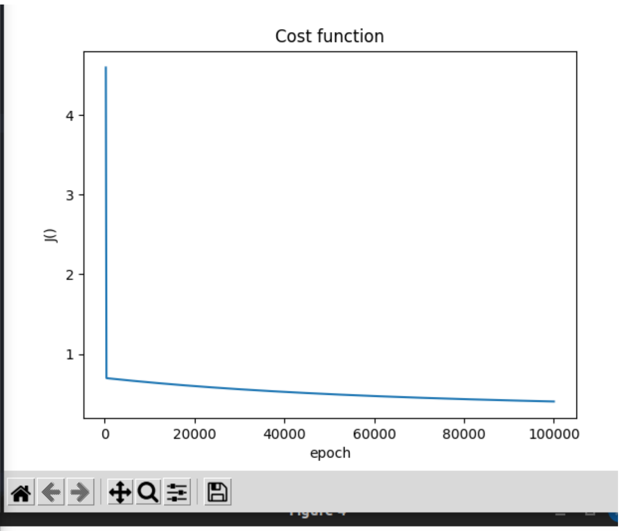
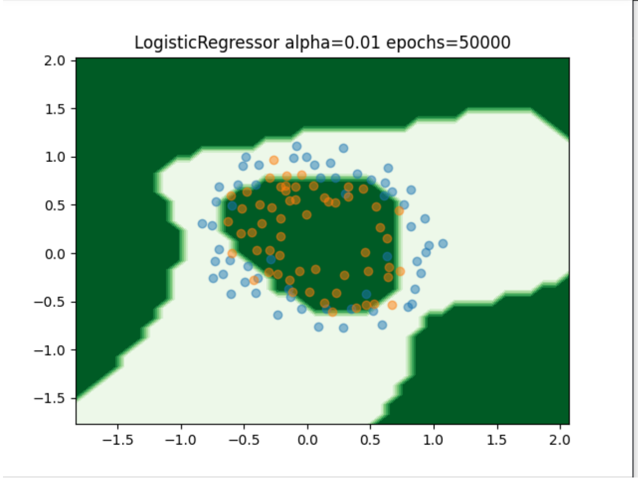


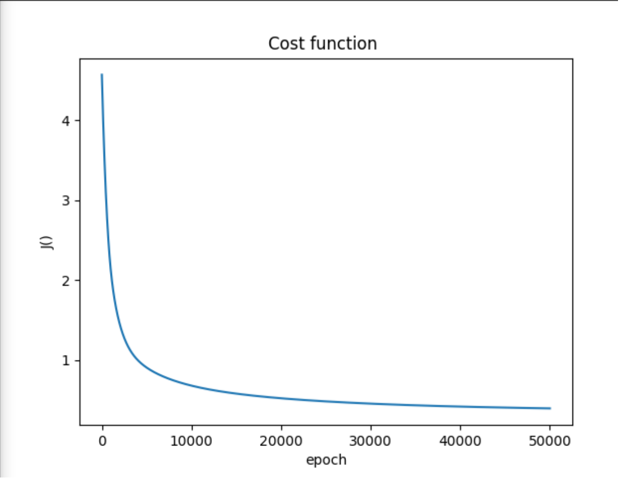
Fig 2. Experiments with good results

Best combination of values









These results were obtained by experimenting with different values for learning rate, epochs, and regularization constant.

```
Final  theta  is  [[-4.37329138     0.04198274     0.03474186]]  (cost:
0.4026080458642832)
[[ 1.          1.          1.        ]
[30.28671077 60.18259939 75.02474557]
[43.89499752 86.3085521  46.55401354]]
predicted as [[0 1 1]] (was [[0 1 1]])
```

```
Final  theta  is  [[   3.14317575     1.43465164     3.01873003   -8.87080067
-6.30651784
 ...
[-1.39810280e-02 -3.40238342e-04  7.73111761e-03]
[ 1.03255971e-01  7.83763594e-05  5.40024334e-03]]
predicted as [[1 0 0]] (was [[1 0 0]])
```

In order to obtain this results we used the following values for the first part of the experiment:

Alpha = 0.001
Epochs = 10,000

Without the use of regularization

And for the second part of the experiment the values used were:

Alpha = 0.01
Epochs = 50000

Using regularization with a constant value of 0.1

Effect of changing the number of iterations

Changing the number of iterations that you allow the algorithm to run, allows the “Neural Network” to get more comfortable with the data set, allowing it to overfit and get better at predicting results more accurately.

This is a technique that should be used with caution because we could eventually overfit the “Neural Network”, meaning that instead of getting good at predicting results. It memorized all the possibilities and is no longer able to generalize and start to fail when presented with new data that has never been seen before.

Also, increasing the number of epochs we allow an algorithm to run, means that we are gonna have a longer waiting time for results.

Regularization effect on Gradient Descent

Regularization is a form of regression whose objective is to reduce the number of errors by fitting on the function additional term that prevents coefficients from overfitting. Its effect on the Gradient Descent, which is an algorithm that focuses on minimizing the cost function by updating its parameters until convergence, is that it improves the generalization of the function, resulting in a better performance of the model. This allows us to improve the process of finding patterns in the dataset and generalizing them to predict new values of x. Regularization prevents the function from giving errors in predicting the correct target values. By reducing the magnitude of the parameters of the cost function, you avoid falling the features into overfitting in the model.