Digit Detector using Logistic Regression
Francisco Iván López Quihuis A01703167
Intelligent Systems Course
Professor Benjamín Valdés
May 31th 2021

**Abstract**

This document presents the implementation and analysis of a logistic regression model used to predict if a certain image corresponds to a *zero* or *one* digit. A binary classifier is implemented using data from the MNIST dataset.

**Introduction**

Logistic Regression is a statistical model used to obtain the probability of a certain class existing, based on certain input variables that are defined for the model. This type of regression is not intended to be use as a predictor; it is designed to serve as a classifier. The classifier implemented in this project is a binary classifier, which means that there are two different classes to be predicted, in more complex approaches to this model, classifiers can be for more than two classes. [4]

The problem to be worked on is to identify whether an image of size 28 by 28 pixels, corresponds to the representation of a digit for number one or a digit for number zero. In the following figure, an example for each class is presented, the goal is to determine with a logistic regression to which class an image corresponds.

**Figure 1**. Example of the two classes to be worked on.

**Logistic Regression**

A logistic regression is a model that must be trained, by using a dataset containing examples for both classes (0 and 1 digit in this project). The logistic function for regression can be represented with the following function, which is also called Sigmoid Function.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

**Figure 2**. Function for logistic regression. [2]

In the presented function, b0 corresponds to the bias of the model, one of the values to be trained and obtained. b1 to bn is going to represent the weights for the model, where each weight corresponds to one of the input values, these weights must also be trained. In this project the inputs are 784 pixels, each one with values between 0 and 255, for this task 784 weights are going to be trained, plus the bias which is an independent value useful to adjust the model.

Due to the nature of this function, when certain values for the weights, the inputs and the bias are evaluated, the result is always going to be between 0 and 1. For the purposes of this project, a resulting value of more than 0.5 is going to describe a number 1 digit, and a value of less than 0.5 is going to describe a number 0 digit.

**Dataset**

The dataset for this project is known as the MNIST dataset, and it contains 40,000 labeled samples, each one describing a different handwritten digit between 0 and 9. The dataset contain 785 columns, where one of the columns is the corresponding label for the sample, (e.g., 1, for a digit describing number one), while the other 784 columns correspond each one to the 784 pixels describing the image.

For a binary classifier, only two classes can be described, to accomplish this purpose, only the samples for 0 and 1 digits are going to be used from the dataset, to be able to decide whether any image of 28 by 28 pixels and grayscale corresponds to a 0 or a 1. The dataset contains 4684 instances for number one and 4132 instances for number zero, this samples were separated from the original dataset by using data frame instructions in Python programming language. The new dataset that only contains samples of zero and one was shuffled randomly and then 80% of this dataset was used for training the model, while the remaining 20% was used for testing the model.

The class or expected value for number 1-digit images is going to be "1.0", while the expected value for number 0-digit images is going to be "0.0", this is the way each sample in the dataset is presented.

**Cost Function**

In Logistic Regression models, a cost function is a function that takes two parameters as input, the hypothesis function of the model, which describes the predicted result by the model, and the target value, which describes the expected result by the model. The cost function is useful for determining the minimization needed to make the model perform better. The function that takes into consideration the logistic regression behavior is presented in the following figure, where "J" represents the cost. [3]

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)})$$

**Figure 3.** Cost Function for Logistic Regression. [1]

Cost function is going to be useful in determining with what cadence the error in the model is decreasing.

**Implementation of the model**

The logistic model was implemented using Python programming language, with added functions from the numpy library, useful for mathematical functions, and the pandas library, useful to extract and manipulate data. The first step in the implementation was to obtain the data of the MNIST dataset (which was in a *.csv file), extract the useful features (samples for "0" and "1" digits), to later separate all the data in the training and testing dataset.

After the data was separated correctly into training and testing datasets, the format for this information was changed from a data frame format to a numpy array format. Numpy arrays are more useful for performing mathematical operations, while data frames are better for data manipulation and separation. Once the data was in the correct format, the training of the model occurred, this was accomplished by applying gradient descent and the cost function to calculate the weights and bias, based on all the inputs from the training dataset. After the model was trained the accuracy was calculated for the trained dataset and for the testing dataset, where expected values were compared to the obtained values based on the same inputs.

**Results obtained from training cycle**

Learning rate is a parameter that determines how fast the gradient descent is applied to the weights being calculated. A higher value for the learning rate means the model is updated with "bigger steps", with the risk that model can be updated so fast that it no longer is trained correctly. In the following figure the cost functions for three different learning rates are presented, after 1500 learning cycles.

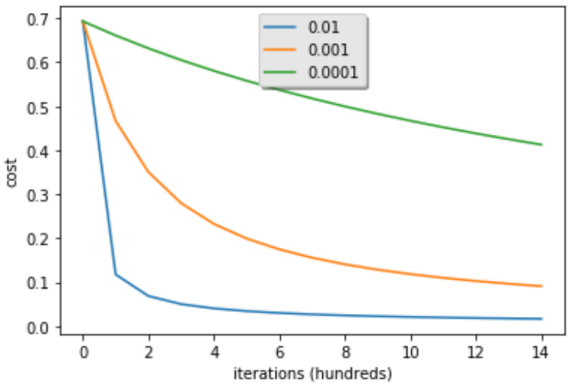


**Figure 4**. Cost function for three different learning rates.

As it can be observed, when 0.01 is used as learning rate, the cost function is minimized faster than when the other two values for learning rate are used. After observing this graph, it can be concluded that for the data being used and the model being trained, using a higher value for the learning rate, like 0.01, does not represent a risk of over fitting in the model, where the costs would increase due to poor performance from the model at certain learning rate.

For the three different learning rates, after 1,500 learning cycles, the train and test **accuracy obtained from the model was above 99%,** which concludes that the trained models from the three different learning rates are almost equally efficient for data coming from the MNIST dataset.

**Real data Results**

After obtaining the accuracy for the model and the cost function associated to the model, it could be concluded that the model is efficient, and it works as intended. For further analyzing the model, four different images were handwritten independently from the dataset used for training, the results for this test would be more concluding in the efficiency of the model. The 4 different images used as real data are shown in the following image.

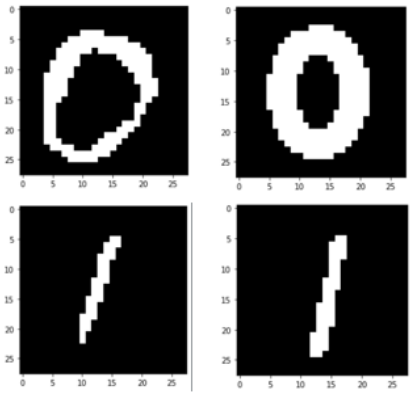

**Figure 5**. Real handwritten images for testing. (Image 1 to 4, from left to right and descending order)

The resulting value from the model, is going to be between 0 and 1, where images associated to a number 0-digit should be close to 0, while images related to 1-digit should be close to 1.

| | Learning Rates | | |
|---|---|---|---|
| Images | 0.01 | 0.001 | 0.0001 |
| Image 1 (0-digit) | 0.0052 | 0.0566 | 0.2827 |
| Image 2 (0-digit) | 7.13e-5 | 0.0081 | 0.1744 |
| Image 3 (1-digit) | 0.8723 | 0.7539 | 0.5541 |
| Image 4 (1_digit) | 0.9933 | 0.9292 | 0.6344 |

**Table 1**. Comparison of output values for different learning rates with real images.

Based on the previous results of the model with different learning rates, the weights obtained after using the learning rate of 0.01 are the better ones, since the obtained values are closer to 0 and 1 in each test.

Considering the accuracy for the model and the behavior with real data, it can be concluded that the is no overfitting nor underfitting, since the maximum error for the testing was around **12% in real data and 1% in training data** for the weights obtained with "0.01" as learning rate.

**Dataset Reference**

Kaggle (n.d.) Digit Recognizer, MNIST Data. Kaggle: https://www.kaggle.com/c/digit-recognizer

**Code Reference**

Valdés Aguirre, Benjamin (2021) perceptron vectorized.py. Google Drive

Ng, Andrew (n.d.) Deep learning AI Course. Coursera.

**References**

[1] octavian. (2017, may 10). *How is the cost function from Logistic Regression derivated*. Retrieved from StackExchange: https://stats.stackexchange.com/questions/278771/how-is-the-cost-function-from-logistic-regression-derivated

[2] Sayad, S. (2021). *Logistic Regression*. Retrieved from saedsayad.com: https://www.saedsayad.com/logistic_regression.htm

[3] Triangles. (2019, november 10). *The cost function in logistic regression*. Retrieved from Internal Pointers: https://www.internalpointers.com/post/cost-function-logistic-regression#:~:text=A%20better%20cost%20function%20for%20logistic%20regression&text=In%20words%2C%20a%20function%20C,label%20was%20y(i)

[4] Wikipedia. (n.d.). *Logistic regression*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Logistic_regression#:~:text=Logistic%20regression%20is%20a%20statistical,a%20form%20of%20binary%20regression