Introduction to Python: Gradient Descent, Back Propagation, and Numerical Gradient Checking

Lab Report # 8

|  |  |  |  |
| --- | --- | --- | --- |
| Sidney Carlo P. Lopez (11222522)  sidneylopez18@gmail.com | |  | Jeremy B. Reccion (11210192)  jeremybreccion@yahoo.com |
| Carl Michael O. Sy (11208929)  carl.michael.sy@gmail.com | Wellington S. Villasin (11209534)  wellvill08@gmail.com | | |
|  |  | | |

*Abstract*— A new programming language is utilized for this experiment which is Python. Anaconda is a packet manager and a collection of open source packages that is within the Python. This is required to program and code gradient descent, back propagation, and numerical gradient checking.

Keywords— gradient descent; backpropagation; numerical gradient checking; gradient; descent; classification; neural; networks; testing;data; Python; Anaconda

1. INTRODUCTION

Python[1] is a programming language where it is similar to Java. This is famous for the use of its interactive model which allows examining a number of snippet codes. It consists of a top level data structure which allows the connection of different components together. Most programmers usually utilize this language since it increase the productivity it supplies. Examples of programs that are made within this language are gradient descent, back propagation, and numerical gradient check. Gradient descent [2] consists of an algorithm which is a minimizes the given functions. If a set of parameters are given within a function , this would be minimized . Back Propagation[3] also stands for “Backward Propagation of errors” where it is a method that allows the instruction of neural networks that utilizes optimization method. This is useful since it would compute the gradient’s loss function. Lastly, Numerical gradient checking [4] is similar as the analytic gradient to the numerical gradient. This allows the user to check the derivative and every dimension of the gradient.

1. OBJECTIVES
2. Gradient Descent

* Focus on the theory of making a better prediction
* Initializing the network with the use of a forward method

1. Back Propagation

* To create an equation to add a code for the gradient
* Application of the chain rule
* To find the rate of change of yHat

1. Numerical Gradient Checking

* To test the gradient computation part of the code
* To provide a deeper understanding of what derivative is to solve the problem

1. IMPLEMENTATION
2. **Gradient Descent**

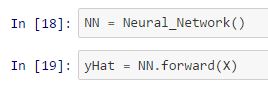
First step is to import the numpy as np and matplotlib.pyplot as plt

C:\Users\sidneycarlolopez\Desktop\1.jpg

Initialize the network with the use of a forward method

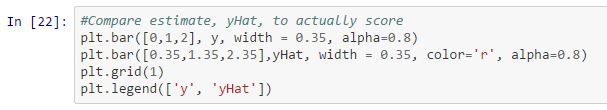


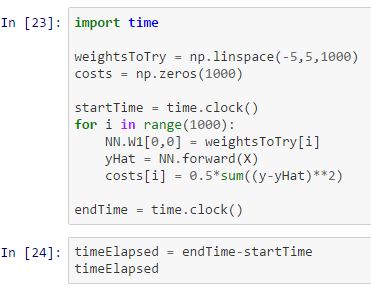
Codes imported from the partTwo

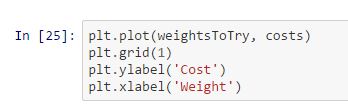


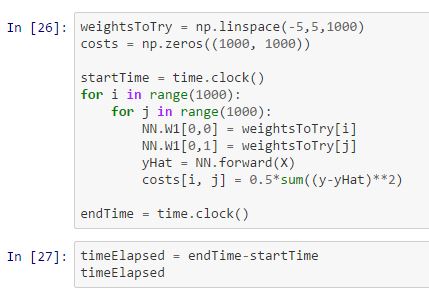
C:\Users\sidneycarlolopez\Desktop\1.jpg

Used to compare, yhat, to actually score





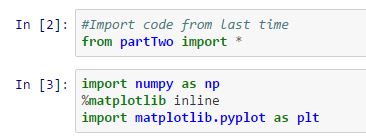


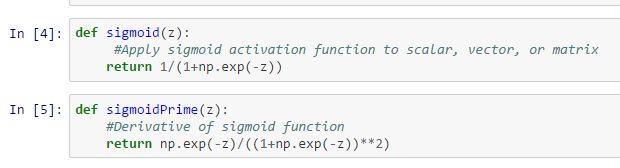


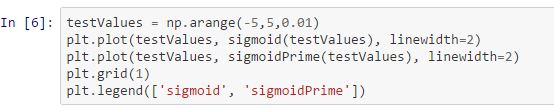
C:\Users\sidneycarlolopez\Desktop\1.jpg

1. **Back Propagation**

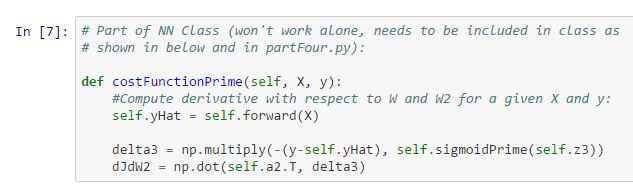
An addition of a new python method for the derivative of the sigmoid function and sigmoid Prime.



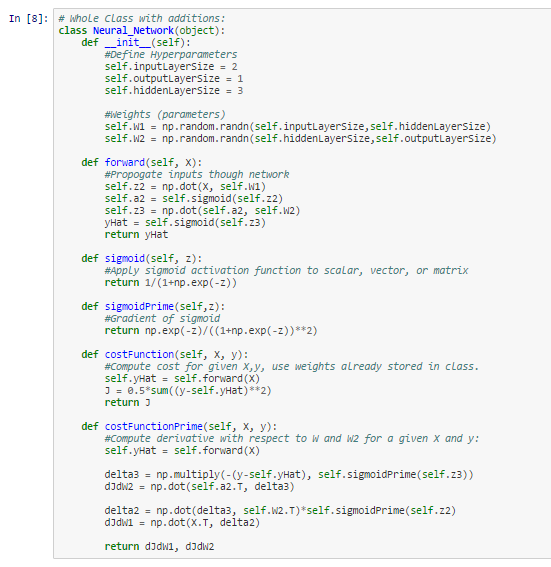




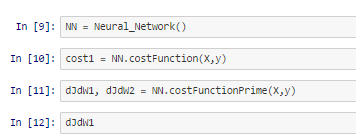
The gradients are coded up with a cost function primes. The dot method performs the matrix multiplication

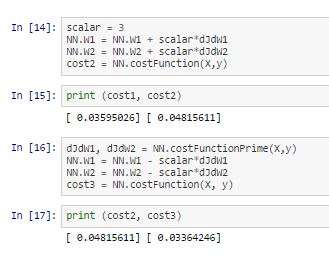


Creation of a deeper neural network, the operations are stacked up together



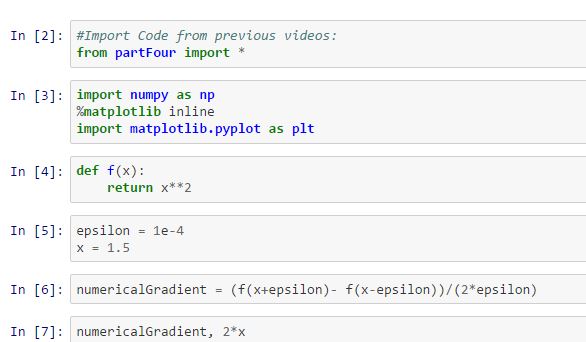
The computation of dJ/dW which is the 9 dimensional optimization space.

 C:\Users\sidneycarlolopez\Desktop\1.jpg



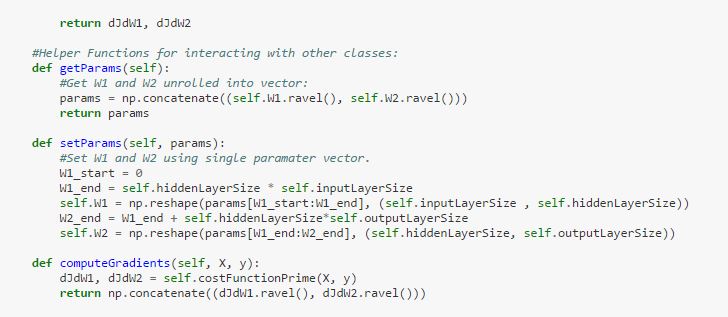
C.**Numerical Gradient**

Compare the results of the symbolic derivative

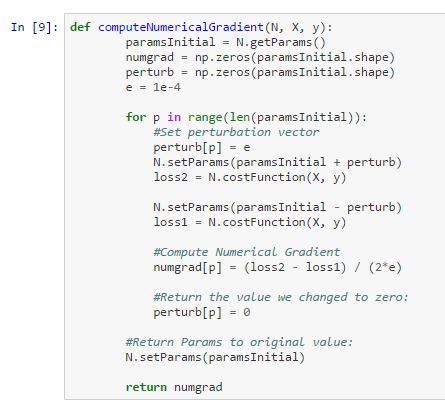
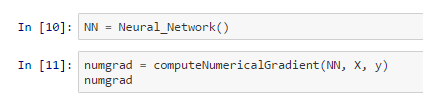


Add helper function to the neural network





Compute the slope within the two values:

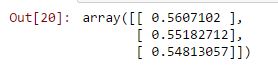
 The results are required to be quantified. C:\Users\sidneycarlolopez\Desktop\1.jpg

C:\Users\sidneycarlolopez\Desktop\1.jpg

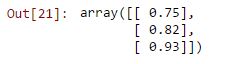
1. DATA AND RESULTS
2. **Gradient Descent**

Output of the yHat

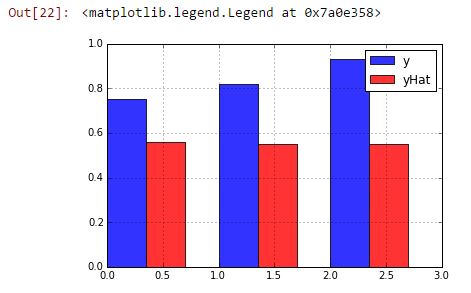
I

 Ou

Output of y



Output of the yhat, to actually score.

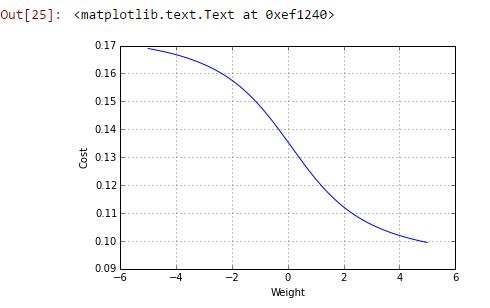


**High Yhat**

It showed a result of 0.04 seconds to know the 1000 different weigh values of the neural network

C:\Users\sidneycarlolopez\Desktop\1.jpg

Output of the cost and weight



Time elapsed:

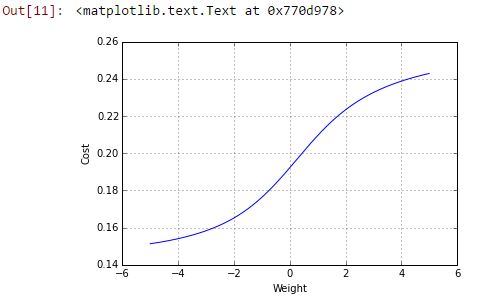


It would take this amount of time of years for the network

**C:\Users\sidneycarlolopez\Desktop\1.jpg**

**Low Yhat**

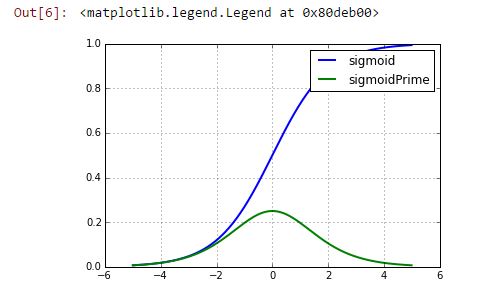
**C:\Users\sidneycarlolopez\Desktop\1.jpg**

**** **C:\Users\sidneycarlolopez\Desktop\1.jpg**

**C:\Users\sidneycarlolopez\Desktop\1.jpg**

1. **Back Propagation**

Output of the sigmoid and sigmoidPrime

****

Print( Cost1, Cost2, Cost3)

**C:\Users\sidneycarlolopez\Desktop\1.jpg**

**C:\Users\sidneycarlolopez\Desktop\1.jpg**

1. **Numerical Gradient**

Output of the numerical gradient:

**C:\Users\sidneycarlolopez\Desktop\1.jpg**



C:\Users\sidneycarlolopez\Desktop\1.jpg



1. ANALYSIS AND CONCLUSION

The experiment introduced a new program called Jupyter notebook which enabled the group to simulate the given files.

REFERENCES

1. <https://www.python.org/doc/essays/blurb/>
2. <http://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/>
3. <http://www.seattlerobotics.org/encoder/nov98/neural.html>
4. <http://cs231n.github.io/neural-networks-3/#gradcheck>