Basic Of Deep Learning - Mid Semester Project Part 1

Oz Gemer (208426460), Ofir Gur Cohen (206588642)

Submitted as mid-semester project report for Basic Of Deep Learning course, Colman, 2023

1 Introduction

This project is about image prediction. We've got a data set full of clothing images. By creating a neural network, we managed to take an image and label it correctly. The motivation was to label two kinds of pictures (clothing). Therefore, we split the inputs into two layers.

1.1 Data

A Data Set called "MNIST Fashion" was given to us. It contained 70,000 black and white images of clothing (T-shirts, trousers, pullovers etc.). The clothes pieces we have chose are sneakers and pullovers (represented them as "3" for dress and "8" for bag). Afterwards, for adjustments purposes, we converted those items into binary values (0/1).

1.2 Problem

Classification between the two labels that we chose.

2 Solution

2.1 General approach

1. Filtering the data into the chosen features.
2. Shuffling the data and splitting it into Training and Testings sets.
3. Implementing sigmoid activation function.
4. Implementing logLoss function.
5. Determine "NN" parameters: numbers of epochs, the learning rate and hidden layers.

6. Randomized weights and biases.
7. Train: We trained the Neural Network using forward and backward propagation and changed the weights and bias accordingly.
8. Visualizing the loss function's graph to see the "NN" learning curve.
9. Testing the performances on the "test group" and displaying the confusion matrix.
10. Testing the "NN" classification on inputs from the tests set.

2.2 Design

The platform we worked on is Google Colab. The training took us a couple of minutes to go through all the epochs. One of the technical problems that we have encountered with was to adjust the matrices into the right dimension.

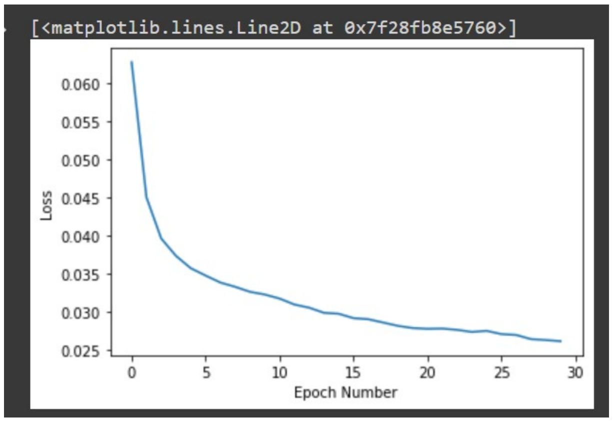
Algorithm 1 Gradient Descent

```
for i in epochs do
    avgEpochs loss = 0
    for j in numOfTraining do
        ————————Forward———————
        Z1 = np.matmul(W1,X[:,j].reshape(inputLayers,1)) + bias1
        A1 = sigmoid(Z1)
        W2.dot(A1) + bias2
        sigmoid(Z2)
        Yout = np.array([Y[j][0]]).reshape(1,1)
        ————————Loss———————
        loss = logLoss( A2, Yout)
        avgEpochLoss = avgEpochLoss + loss
        ————————Backward———————
        dZ2 = A2-Yout
        dW2 = dZ2 * A1.T
        db2 = Sum(dZ2)
        dA1 = sigmoidDeriv(Z1)
        dZ1 = W2.T * dZ2
        dW1 = input[j]
        db1 = Sum(dZ1)
        ————————Updating———————
        W2 = W2 -(learningRate * (dW2))
        b2 = b2 - (learningRate * db2)
        W1 = W1 -(learningRate * (dW1))
        b1 = b2 - (learningRate * db2)   =0
```

3 Base Model

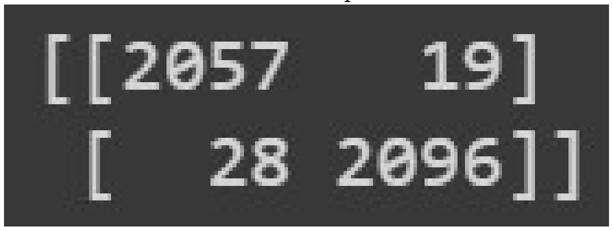
Our model had 784 inputs, with each input representing a pixel in an image. The model had a single hidden layer with 20 neurons and a single output node for binary classification. The activation function used was sigmoid and the loss function was cross-entropy. We trained the model for 30 epochs using a learning rate of 0.05 and obtained the following results.

3.1 Results and Metrics



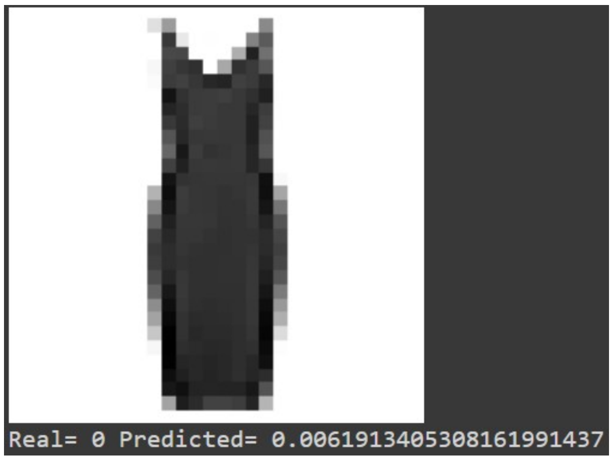
```
[<matplotlib.lines.Line2D at 0x7f28fb8e5760>]
```

Loss Graph



```
[[2057   19]
 [  28 2096]]
```

Confusion Matrix



```
Real= 0 Predicted= 0.006191340530816199143
```

End Result

4 Discussion

After building the model we discovered that it has high accuracy rate since it is only classifying two items - binary classification. We have reached the conclusion that the problem can be solved with a simpler "NN". The insights that we have gained during the process are:

- Changing the alpha to a larger value can cause the model to look for higher values rather than gather to a minimum.
- There is a correlation between the number of epochs and the loss's results.
- Enlarging the hidden layers to high number can cause the model to be over-fitted to the train.

5 Code

https://colab.research.google.com/drive/1D526ReHCm-Rju8F1h1ETJVEuhQFhQ84G?usp=share$_link$