Basic Of Deep Learning - Mid Semester Project Part 1

Stav Elizur Tommy Afek

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

1 Introduction

This project includes understanding the basics of Deep Learning neural networks. Which is understanding the basics of writing a neural network from scratch, understanding some basic terms such as Classification, Gradient Descent which is the combination of Forward and Back propagation, Epoch, Learning rate, Confusion matrix , Loss Function and Activation Function. The project made us use all these terms to be able to answer a simple binary classification problem.

1.1 Data

We used Fashion MNIST data set that contains 70,000 grayscale images in 10 categories. we split the data set into two classes - Dress and Ankle boot. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 784 columns. The first column consists of the class labels (see above), and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

1.2 Problem

Our problem was that we weren't able to classify an image between 2 options, Dress and Ankle boot.

2 Solution

Neural network that consists 784 input neurons + 1 Bias neuron, 1 hidden layer with 12 neurons + 1 Bias neuron and 1 output neuron.

2.1 General approach

Our project solved a binary classification problem between images of dress and ankle boot using the gradient descent algorithm and running over our data Set for a constant epoch number. Also, we have a constant neuron number in our hidden layer.

2.2 Design

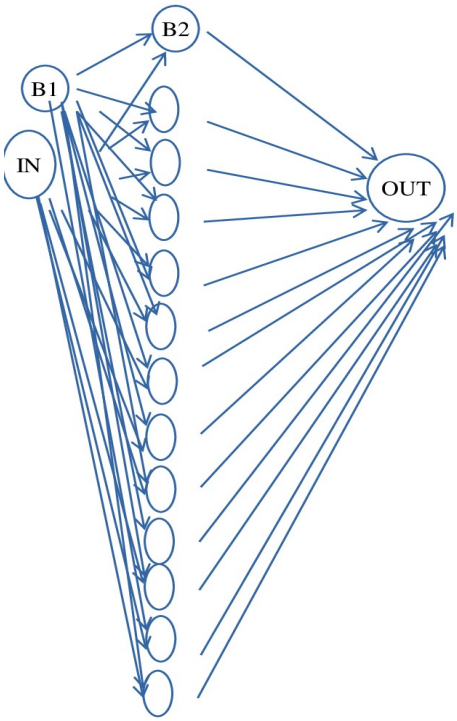
The architecture of our code has 3 main stages: Data processing, Training and Performance. In our code, we started by loading and processing the data set and then we needed to do Feature Scaling on it. Feature scaling is a method used to normalize the data. Now, we have to split over data to two classes that each class has 7000 images. After that, we shuffled our data set and split the data into train set which includes 12000 images and test set includes 2000 images, The train set is used basically to train our model using the Gradient Descent algorithm and the test set is used to authenticate that our data set is not over fitted and can be used on a different data and also to check overall performance. Since we want to provide a solution for a classification problem, we had to normalize our classification labels to 1 and 0. Then, we also tested our data by checking that the classes we selected are actually displayed and we had our data processed correctly as we wanted. Afterwards we wrote the Sigmoid activation function which is used in our hidden layer and output layer and the purpose of this function to convert and normalize a number to a number between 0 and 1. Also we wrote the loss function, which is a cross-entropy binary loss and we use it when we want to calculate the average loss in each epoch we train for evaluating our performance. We also defined our Neural Network Hyper Parameters that are the input layer, hidden layer, learning rate, and epochs. After initializing our hyper parameters we initialized our Weight and Bias and we started finally training our data set. The train process is built by running over 20 epochs on our training set and inside of that we use our Gradient Descent algorithm on each variable in our training set which does the Forward propagation, computes the loss and Back propagation and update the weights accordingly. The train process took for about 2 and half minutes. After that, we tested our performance by using our test set on the model we built by checking predictions versus the test set classifications and printing the confusion matrix in order to calculate the accuracy. Our technical challenges were that we had to follow the right shaping of the matrices in order to multiply it in the right order and not getting a running time exception which made us progress slower because we had to debug and fix it. Also selecting the Epoch and the number of hidden Neurons was a problem, because we had to try different numbers and infer from each number that we tried what is the next number to try.

Algorithm 1 Gradient Descent

$Z1 \leftarrow W1 \times np.asmatrix(X[:, j]).T + b1$ ▷ Forward propagation
$A1 \leftarrow Sigmoid(Z1)$
$Z2 \leftarrow W2 \times A1.T + b2$
$A2 \leftarrow Sigmoid(Z2)$
$Yout \leftarrow YTrain[j, 0]$
▷ Compute loss
$loss \leftarrow log_loss(A2, Yout)$
$avgEpochLoss \leftarrow avgEpochLoss + loss$
▷ Back propagation
$dZ2 \leftarrow (A2 - Yout)$
$dW2 \leftarrow dZ2 \times A1$
$db2 \leftarrow dZ2$
$dA1 \leftarrow dZ2 \times W2$
$dZ1 \leftarrow (((A1)[0, i]) * (1 - (A1)[0, i]) \times W2.T) \times dZ2$
$dW1 \leftarrow dZ1 \times X[:, j]$
$db1 \leftarrow dZ1$
▷ Update weights
$W2 \leftarrow W2 - learningRate \times dW2$
$b2 \leftarrow b2 - learningRate \times db2$
$W1 \leftarrow W1 - learningRate \times dW1$
$b1 \leftarrow b1 - learningRate \times db1$

3 Base Model

Our base model is a neural network that is built from 1 input layer of 784 neurons + 1 constant bias neuron, 1 hidden layer that has 12 neurons and 1 bias neuron, and 1 output layer that has the output neuron. Our base models gets and image of 28x28 being reshaped to a vector of 784 pixels and it computes by using the Forwarding method the predicted value of the input.

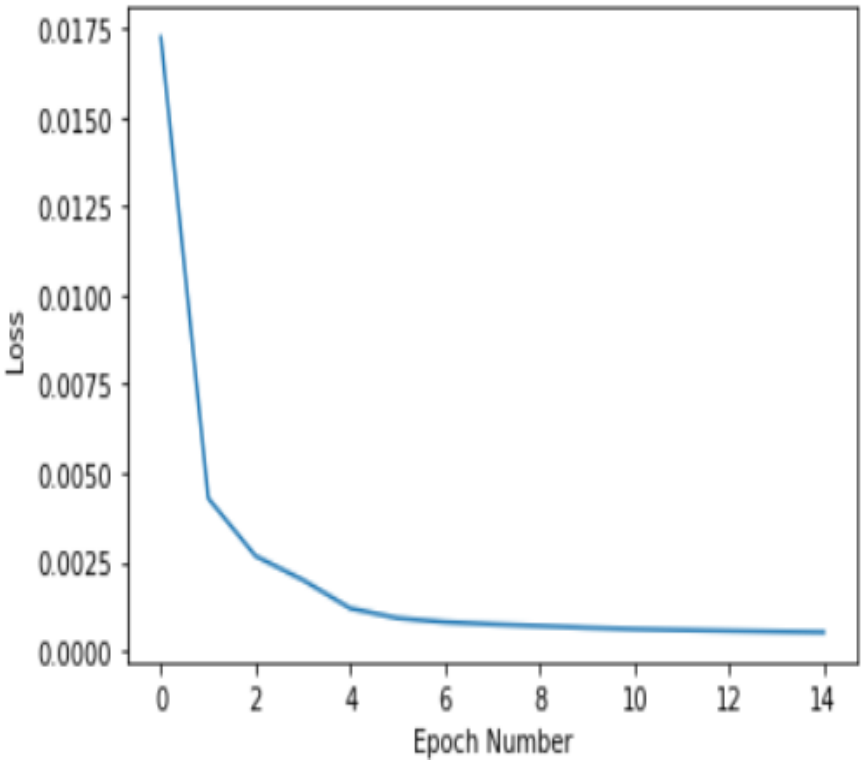


Notes:
B1 – Bias neuron of input layer
B2- Bias neuron of hidden layer
In- Input neuron layer – it has 784 Neurons inside.
OUT- Output neuron of the output layer

An image of the base model

3.1 Results and Metrics

After we ran the model over our training set we saw that our overall loss was very close to 0. Then we ran our test set over our base model and we checked the prediction against the test‘s label. Then, we built the Confusion Matrix, and we saw that we had close to 100 percent Accuracy and in some runs 100 percent Accuracy which showed us that our model is not over fitted to the training data and also that we have a good model to our requirements.



An image of loss by epoch

```
[[ 982    0]
 [   0 1018]]
```

An image of our confusion matrix

4 Discussion

To summarize, we found that building a neural network takes a lot of patience, experiments and of course math. Also we learned that having a small change in the number of epochs or the number of hidden neurons can make a great impact on the model itself.

5 Code

Colab project