CSE 574 Project 2: Multi-class Classification using Neural network and Convolutional Neural Network

Bhagyashri Thorat (Person ID: 50290581)
Email: bthorat@buffalo.edu

18th October 2019

Abstract

Computers don't process the images as humans can do, each image is represented as a matrix of pixels which can be one for more numbers. Having models that can classify the images is hence restricted by the curse of dimensionality for models like linear regression and classification. So, fixing the number of basis functions and making them adaptive during training is used for such large scale training problems like multilayer classification of images. In this project, the goal was the classify the MNIST fashion image dataset into 10 class labels. Neural network image classification was used to train the data.

1 Introduction

Machine learning is a subset of Artificial Intelligence. It's used to model the computer systems based on specific algorithms to perform a specific task based on the previous results and errors. \

The neural network or Convolutional neural networks are inspired by biological neurons in the brain and animal visual cortex respectively. The neural network classifiers consist of an input layer, hidden layers, and output layer or the classified layers. And the Convolutional neural networks are the regularized versions of multi-layer perceptions, in them, the receptive fields overlap such that they cover the entire filed.

The fashion MNIST dataset was 85.7% as training consisting of 60,000 images and 14.28% as testing or validation data consisting of 10,000 images. The model was trained on this training data and then accuracy, recall, and precision were calculated for validation data or training data using 3 different methods.

1.1 Neural Networks

Part 1

In part 1, the neural network was implemented without using any libraries using

- 1 hidden layer and
- softmax as activation function and
- cross-entropy as loss function

The softmax can be defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_K^{k=1} e^{z_k}} \text{for } j = 1, \ldots, k$$

The sigmoid function highlights the largest value and suppresses the smaller values and hence can be used in probability theory as a representation of categorical distribution.

Sigmoid can be defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Part 2

In part 2, the Tensor flow library was used using

- 2 hidden layers and
- ReLU, softmax as activation function and
- sparse_categorical_crossentropy as loss function and
- the “adam” optimizer.

Part 3

In part 3, Convolutional neural network was used having

- layers of ReLU activation and one softmax activation function layer and
- categorical_crossentropy as loss function and
- the “adam” optimizer.

The hyperparameters are:

1. Epochs
2. Learning rate
3. Number of neurons in the hidden layer and
4. Number of hidden layers

Deciding the number of activation function to use is one of the hyperparameters for all 3 parts. The weights are initialized to a random number.

1.2 Methodology

The steps used to predict the output are as follows:

1. Extract feature values and labels from the data
2. Data partitioning
3. Train using Neural Networks with One hidden layer
4. Train using multilayer neural network
5. Train using Convolutional neural network
6. Tune hyper-parameters
7. Test your machine learning scheme on the testing data

2 Dataset

In this project, the dataset used was of Fashion MNIST, consisting of 70,000 grayscale images in 10 categories. The images show individual articles of clothing at low resolution (28 by 28 pixels), as seen below out of which 60, 000 examples for training and 10, 000 examples for testing. Each image is 28 * 28 grayscale image associated with a label from 10 classes consisting of Top, Trousers, Pullover, Shirt, etc. The pixel values are of grayscale so they consist of numbers between 0 and 255.

3 Preprocessing

In the preprocessing step, the data is converted from raw data into clean data to achieve better results. The following steps were performed in the preprocessing step:

Part 1 and Part 2:

- By inspecting the first image in the dataset, it is seen that its value falls between 0 to 255. We, therefore, have to scale the pixel values to a range of 0 to 1.
  To do so, we just need to divide the training and the testing data by 255.

```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

  The testing and training data is preprocessed in the same way. To verify that the data is ready, we can display the first 25 images as shown above.
- Additionally, the input data needs to be flattened from (60000, 28, 28) into (60,000, 784) for both of these parts.

Part 3:

The CNNs uses little pre-processing compared to other image classification algorithms.

- The input data feed is can be 3D and has dimensions (28, 28, 1)
- Also, one hot encoding is done on the input data using Keras.util package's to_categorical method

4 Architecture

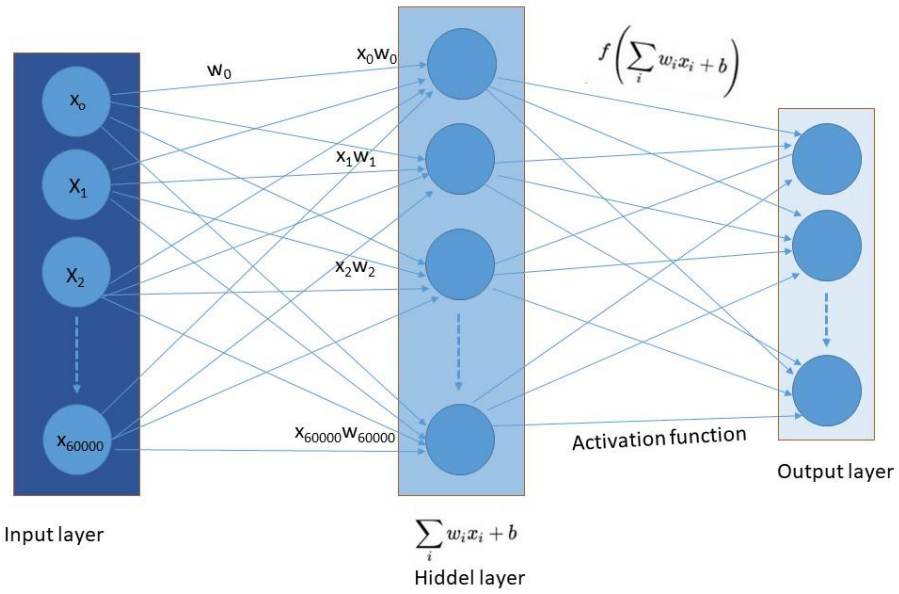
Part 1:

In the neural network, there are 2 phases:

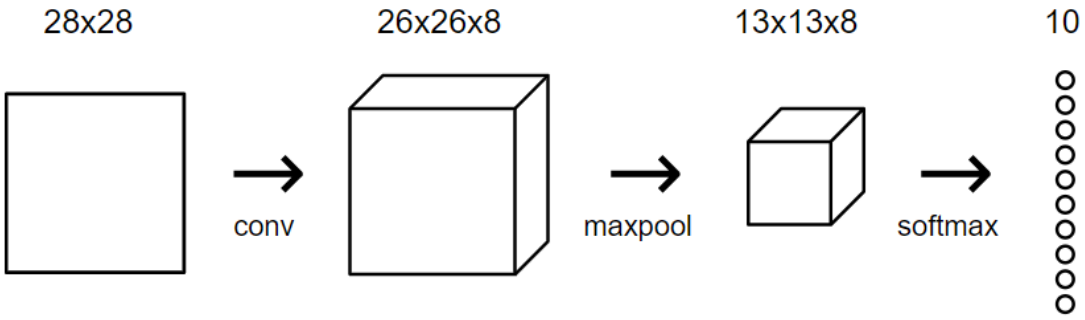
1. Feedforward
   a. Calculate error
2. Backpropagation

In the first pass, the feed-forward uses the data and weights of the network to compute a prediction. Then the error is calculated based on the prediction and the given labels. The final step propagates the error back to the starting layer and thus the weights and the biases get updated in each iteration.
The hyperparameters used are:

5. Epochs
6. Learning rate
7. Number of neurons in the hidden layer and
8. Number of hidden layers



Part 3:



Where bias is b and w1, w2 are weights.

Formulae:

Part 1 and 2:
Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

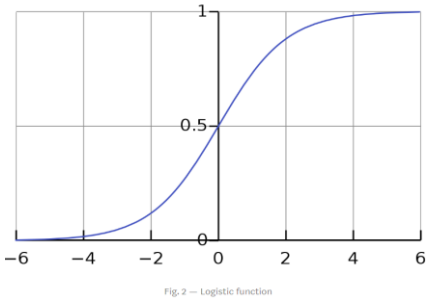


Fig. 2 — Logistic function

Derivative of sigmoid:

$$\frac{d\sigma(x)}{d(x)} = \sigma(x) \cdot (1 - \sigma(x))$$

Cross entropy function:

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$

where:
n is the total number of items of training data,
the sum is over all training inputs, x, and
y is the corresponding desired output.

Part 3:

Convolution Output

$$O = \frac{W - K + 2P}{S} + 1$$

W = input height or length

K = kernel length or size

P = padding

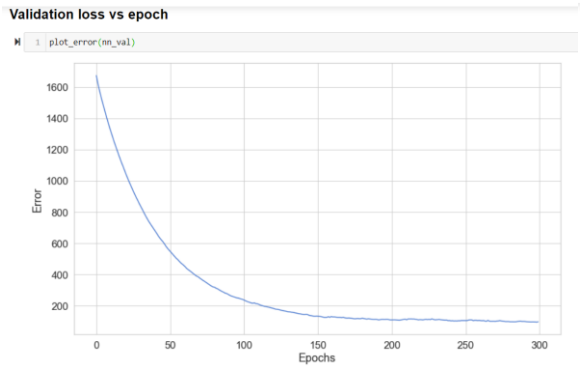
S = stride

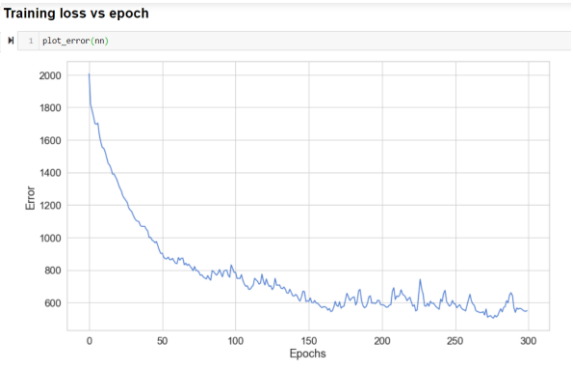
5 Results

Part 1:
Based on the trained model and

1. Hyperparameters:
    - n_hidden_units=50,
    - l2=0.5,
    - l1=0.0,
    - epochs=300,
    - learning_rate=0.001,
    - n_batches=25,

    The train accuracy is 76.73% and test accuracy is 73.66%





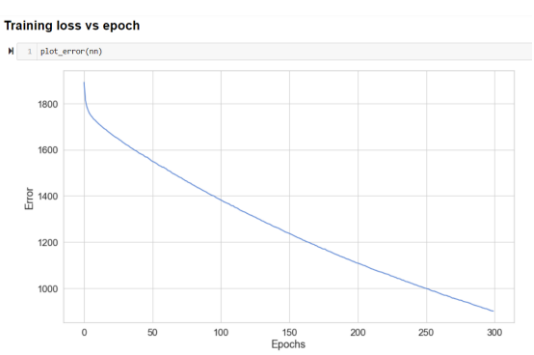
2. Hyperparameters:
    - n_hidden_units=50,
    - l2=0.5,
    - l1=0.0,
    - epochs=300,
    - learning_rate=0.01,
    - n_batches=25,

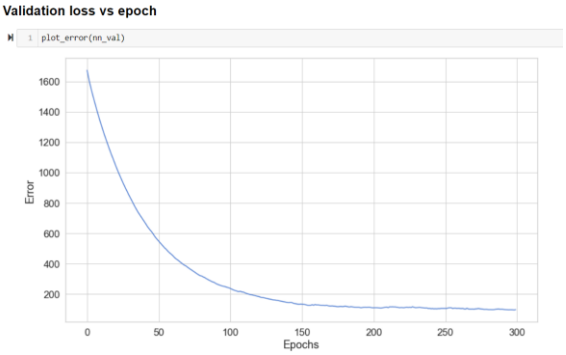
3. The loss values kept on increasing and increased to 100k, this proves that this 0.01 learining rate is very large. So, now by decreasing the learning rate in the next iteration we get:
4. Hyperparameters:

   n_hidden_units=50,
   l2=0.5,
   l1=0.0,
   epochs=300,
   learning_rate=0.0001,
   n_batches=25,

   Train Accuracy: 83.65%
   Test Accuracy: 82.07%

We see that this gives better accuracy.





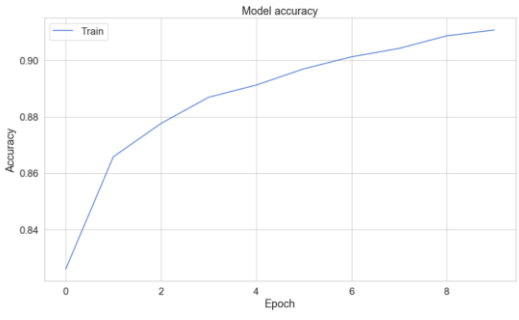
5. Also, to increase the accuracy, by applying the sklearn preprocessing scale
   On hyperparameters:
   - n_hidden_units=50,
   - l2=0.5,
   - l1=0.0,
   - epochs=300,
   - learning_rate=0.001,
   - n_batches=25,

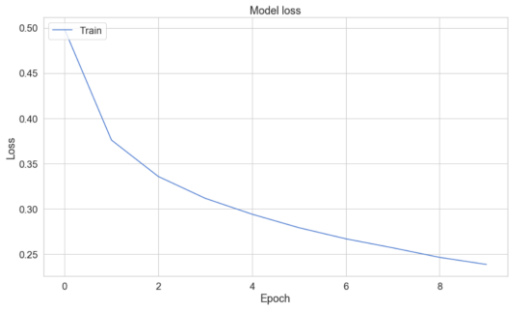
gives us accuracy of Train Accuracy: 92.13%, Test Accuracy: 87.03% which is better than the earlier accuracy of

Part 2

1. Hyperparameters:

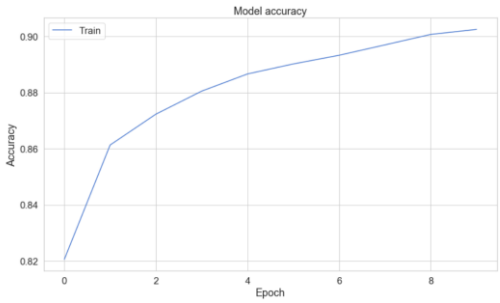
   Activation function: ReLU, Softmax
   Hidden layers: 128, 10
   Optimizer: adam
   Epochs: 10
   Training accuracy: 91.08
   Test accuracy: 88.61

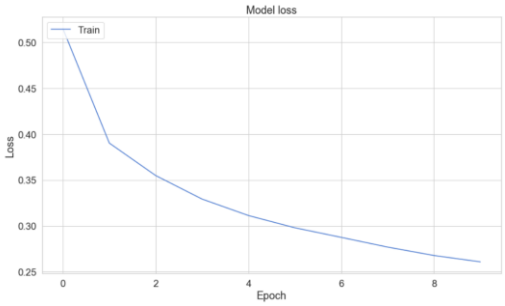




2. Hyperparameters:

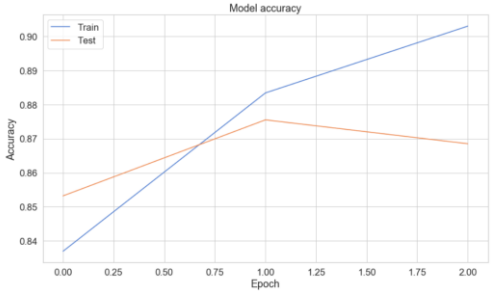
Activation function: ReLU, Softmax
Hidden layers: 64, 10
Optimizer: adam
Epochs: 10
Training accuracy: 90.25
Test accuracy: 86.67

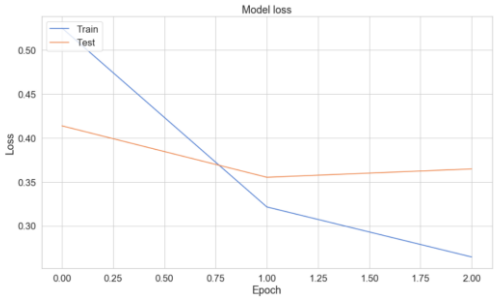




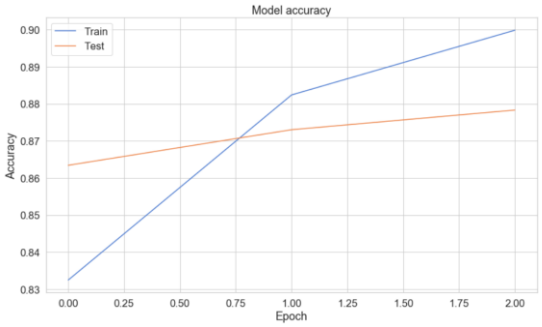
Part 3:

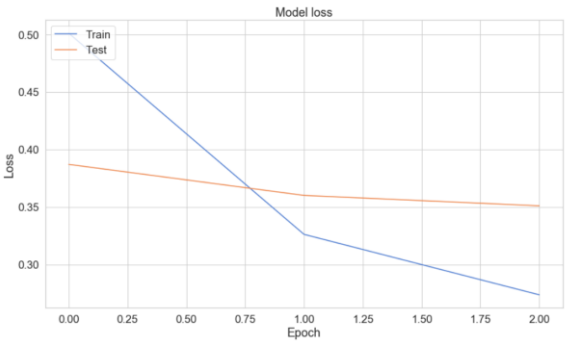
1. Hyperparameters:
   Activation function: ReLU, ReLU, Softmax
   Hidden layers: 64, 32
   Optimizer: adam
   Epochs: 3
   Training accuracy:90.3
   Testing accuracy: 88.61





2. Hyperparameters:
   Activation function: ReLU, ReLU, Softmax
   Hidden layers: 128, 32
   Optimizer: adam
   Epochs: 3
   Training accuracy: 89.99%
   Testing accuracy: 87.83%





Summary

Part 1						
hidden_units	epochs	learning_rate	batches	Test Accuracy	Training accuracy	Loss Function
50	300	0.001	25	73.66	76.73	Cross entropy
50	300	0.001	25	82.07	83.65	Cross entropy
50	500	0.01	25	60.27	66.67	Cross entropy
After scaling						
50	300	0.001	25	87.03	92.13	Cross entropy
50	300	0.0001	25	84.66	87.66	Cross entropy
50	500	0.01	25	0.27	66.67	Cross entropy
Part 2						
128, 10	10	adam: 0.01		88.61	91.08	sparse_categorical_crossentropy
64, 10	10	adam: 0.01		86.67	90.25	sparse_categorical_crossentropy
Part 3						
64, 32	3	adam: 0.01		88.61	90.3	categorical_crossentropy
128,32	3	adam: 0.01		87.83	89.99	categorical_crossentropy

6 Conclusion

Based on the results, of part 1, part 2, part 3 we have classified the data with an 82% accuracy and by using sklearn package for preprocessing, we got accuracy of 87 % for part 1.
For part 2, the maximum achieved accuracy was of 88.67%
For part 3, the maximum achieved accuracy was of 88.61%
Also, by observing the results, we can see that the testing accuracy is always less than the training accuracy which means our data is not getting overfitted with the hyperparameters provided which is one of the major concerns of neural network or Convolutional neural network.

7 References:

1. https://www.curiousily.com/posts/fashion-product-image-classification-using-neural-networks/
2. Lecture notes.
3. Deep learning at UB Workshop notes
4. http://neuralnetworksanddeeplearning.com/chap3.html