Assignment 2 Report:

Done By : Ashwath Raghav

Course: IST 597 - Deep Learning

Github Link:

https://github.com/ashwathraghav/IST597-Assignment2.git

Assignment 2

- Ashwath Raghav

Introduction:

In this assignment, we were asked to explore the Multi Layer Perceptron model and play with the hyper parameters to learn how a basic neural network works and to gain hands-on experience of training a neural network. The model was trained with 2 unique datasets MNIST_784 and Fashion-MNIST. Experiments were conducted to create an optimised model that neither underfits or overfits the model and with a good test accuracy. Graphs comparing the test accuracy of the different models are compared and the best model is chosen by applying statistical techniques.

Core Experiment 1:

Mnist dataset comprises 70,000 [28 x 28] pictures of 0-9 digits.

Data Preprocessing/Feature Engineering:

1. The dataset MNIST_784 is retrieved using fetch_openml. Since the dataset is in the form of pixels, we need to convert it into float data and the categorical labels should be converted to one-hot vectors. We use to_categorical() for this.

2. Since we have 28 x 28 pixels, each picture will be of size 784. Thus, we need to initialise the size of the input layer as 784.

3. There are 10 output labels. Thus, we need to initialise the size of the output layer as 10.

4. Now we need to continuously reduce the dimensionality from 784 to 10. So we assume the size of the first hidden layer as 128 and the second hidden layer as 64.

5. We now have to split the dataset into training and testing dataset. For this, I have employed train_test_split from sklearn. After that, we split the train and test data into batches and we feed them to the models.

Experiments performed on this dataset:

1. Baseline Model:

   Employed Model provided by Ankur Mali without making any modifications. Achieved really bad accuracy with this model probably because of not applying softmax optimisation to output. Achieved 12% accuracy.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU

Assignment 2

- Ashwath Raghav

2. Employing Softmax Activation to the Baseline Model:

   Employed softmax optimization to the output layer. Achieved a really good accuracy, about 80%.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU

3. Employing Dropout Regularisation technique during Forward Pass:

   Employed Regularisation technique, Dropout to improve accuracy. Achieved a small increase in accuracy, about 1%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = Dropout

4. Employing L1 Penalty Regularisation technique during Backward Pass:

   Employed Regularisation technique, L1 Penalty to improve accuracy. I observed that accuracy decreased because of applying this regularisation technique, about 10%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1 Penalty

5. Employing L2 Penalty Regularisation technique during Backward Pass:

   Employed Regularisation technique, L2 Penalty to improve accuracy. I observed that accuracy decreased because of applying this regularisation technique, about 15%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L2 Penalty

6. Employing Elastic Net[L1+L2] Regularisation technique during Backward Pass:

   Employed Regularisation technique, Elastic Net Regularization to improve accuracy. I observed that accuracy decreased because of applying this regularisation technique, about 16%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1+L2 Penalty

7. Employing Trial and Error Hyper Parameter Optimisation by changing Batch Size:

   Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other batch sizes apart from the original batch size. I chose 15 and 128 training batch size. After training the model, I noticed that accuracy was inversely proportional to the batch size for our dataset. So, when the model was fed with lower batch size, it performed better than a larger training batch size.

   Exp1: Hyper Parameters: Batch Size = 15, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 53%

   Exp2: Hyper Parameters: Batch Size = 128, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 77%

8. Employing Trial and Error Hyper Parameter Optimisation by changing Activation Function:

   Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other activation functions apart from ReLU. I chose Tanh and Sigmoid as activation functions in the input and hidden layers. After training the model, I noticed that accuracy dropped significantly because of employing these alternative activation functions. ReLU performed far better for our dataset.

   Exp1: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = sigmoid, Accuracy = 9.1810%

   Exp2: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = tanh, Accuracy = 12.9333%

9. Employing Trial and Error Hyper Parameter Optimisation by changing Learning Rate:

   Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other learning rates apart from the original learning rate, 1e-4. I chose the learning rate as 1e-3 and 1e-5 and performed the model training process. After training the model, I noticed that accuracy of our model for this dataset is inversely proportional to learning rate. Lower the learning, higher is the accuracy.

   Exp1: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-3, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 78.42%

   Exp2: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-5, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 66.43%

10. Employing Trial and Error Hyper Parameter Optimisation by changing Optimizer:

    Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other optimizers for backward pass apart from SGD. I chose Adagrad and Adam as optimizers. After training the model, I noticed that accuracy dropped significantly because of employing these alternative optimizers. SGD performed far better for our dataset.

Exp1: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = Adagrad, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 15.38%

Exp2: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = Adam, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 31.50%

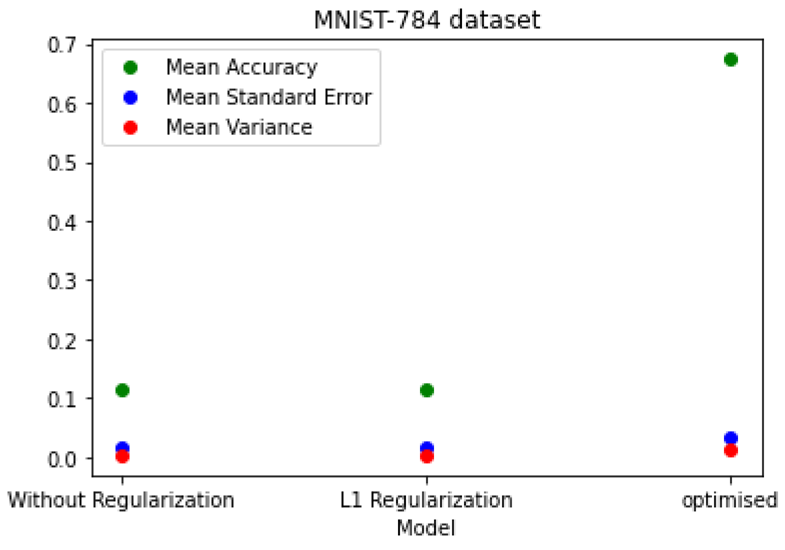
11. Calling the most optimal function 10 times and plotting the accuracy for each time:

    From all of these experiments, I observed that the best set of hyper parameters for our dataset are:

    Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = Dropout.

    After training and testing the model for 10 times, I observed 0 changes in the accuracy. There might have been minute changes in the decimal points but since I typecasted it, I was not able to plot the minute changes in accuracy, but the optimal model had an accuracy of 77.51%.

    Graph comparing variance, std error and accuracy of the optimised model, baseline and the L1 regularised model:



Assignment 2

- Ashwath Raghav

Core Experiment 2:

Fashion-Mnist dataset comprises 70,000 [28 x 28] pictures of items like shirts, shoes,etc.

Data Preprocessing/Feature Engineering:

1. The dataset Fashion-MNIST is retrieved using fetch_openml. Since the dataset is in the form of pixels, we need to convert it into float data and the categorical labels should be converted to one-hot vectors. We use to_categorical() for this.

2. Since we have 28 x 28 pixels, each picture will be of size 784. Thus, we need to initialise the size of the input layer as 784.

3. There are 10 output labels. Thus, we need to initialise the size of the output layer as 10.

4. Now we need to continuously reduce the dimensionality from 784 to 10. So we assume the size of the first hidden layer as 128 and the second hidden layer as 64.

5. We now have to split the dataset into training and testing dataset. For this, I have employed train_test_split from sklearn. After that, we split the train and test data into batches and we feed them to the models.

Experiments performed on this dataset:

1. Baseline Model:

   Employed Model provided by Ankur Mali without making any modifications. Achieved really bad accuracy with this model probably because of not applying softmax optimisation to output. Achieved 2.26% accuracy.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU

2. Employing Softmax Activation to the Baseline Model:

   Employed softmax optimization to the output layer. Achieved a really good accuracy, about 67.35%.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU

3. Employing Dropout Regularisation technique during Forward Pass:

Employed Regularisation technique, Dropout to improve accuracy. Achieved a small increase in accuracy, about 7%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = Dropout, Accuracy = 72.68%

4. Employing L1 Penalty Regularisation technique during Backward Pass:

   Employed Regularisation technique, L1 Penalty to improve accuracy. I observed that accuracy decreased because of applying this regularisation technique, about 20%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1 Penalty, Accuracy = 53.38%

5. Employing L2 Penalty Regularisation technique during Backward Pass:

   Employed Regularisation technique, L2 Penalty to improve accuracy. I observed that accuracy decreased because of applying this regularisation technique, about 10%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L2 Penalty, Accuracy = 63.23%

6. Employing Elastic Net[L1+L2] Regularisation technique during Backward Pass:

   Employed Regularisation technique, Elastic Net Regularization to improve accuracy. I observed that accuracy decreased because of applying this regularisation technique, about 12%. Thus, the best regularisation technique to employ for our dataset is dropout technique.

   Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1+L2 Penalty, Accuracy = 61.26%

7. Employing Trial and Error Hyper Parameter Optimisation by changing Batch Size:

   Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other batch sizes apart from the original batch size. I chose 15 and 128 training batch sizes. After training the model, I noticed that accuracy was inversely proportional to the batch size for our dataset. So, when the model was fed with lower batch size, it performed better than a larger training batch size.

Exp1: Hyper Parameters: Batch Size = 15, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 58.97%

Exp2: Hyper Parameters: Batch Size = 128, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 64.78%

8. Employing Trial and Error Hyper Parameter Optimisation by changing Activation Function:

   Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other activation functions apart from ReLU. I chose Tanh and Sigmoid as activation functions in the input and hidden layers. After training the model, I noticed that accuracy dropped significantly because of employing these alternative activation functions. ReLU performed far better for our dataset.

   Exp1: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = sigmoid, Accuracy = 5.18%

   Exp2: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = tanh, Accuracy = 18.1143%

9. Employing Trial and Error Hyper Parameter Optimisation by changing Learning Rate:

   Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other learning rates apart from the original learning rate, 1e-4. I chose the learning rate as 1e-3 and 1e-5 and performed the model training process. After training the model, I noticed that accuracy of our model for this dataset is inversely proportional to learning rate. Lower the learning, higher is the accuracy.

   Exp1: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-3, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 67.41%

   Exp2: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-5, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 65.55%

10. Employing Trial and Error Hyper Parameter Optimisation by changing Optimizer:

    Performed Hyper Parameter Tuning by employing the Trial and Error method. I chose two other optimizers for backward pass apart from SGD. I chose Adagrad and Adam as optimizers. After training the model, I noticed that accuracy dropped significantly because of employing these alternative optimizers. SGD performed far better for our dataset.

    Exp1: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = Adagrad, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 19.60%

Exp2: Hyper Parameters: Batch Size = 20, Learning Rate = 1e-4, Optimizer = Adam, Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 46.38%

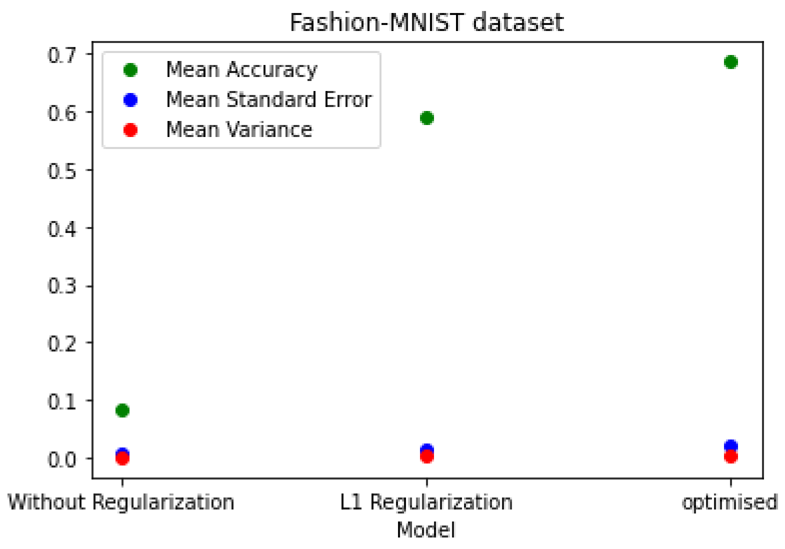
11. Calling the most optimal function 10 times and plotting the accuracy for each time:

    From all of these experiments, I observed that the best set of hyper parameters for our dataset are:

    Batch Size = 20, Learning Rate = 1e-4, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = Dropout.

    After training and testing the model for 10 times, I observed 0 changes in the accuracy. There might have been minute changes in the decimal points but since I typecasted it, I was not able to plot the minute changes in accuracy, but the optimal model had an accuracy of 68.42%.

    Graph comparing variance, std error and accuracy of the optimised model, baseline and the L1 regularised model:



Conclusion:

From all these experiments, we can observe that the test accuracy is not that great for both of our datasets even after employing an optimised version of MLP. This is probably because the depth of MLP is less, i.e.,the number of hidden layers is not sufficient enough for our datasets. So the best way to improve accuracy is to employ more complex neural networks like DNN, CNN,etc.