Assignment 3 Report:

Done By : Ashwath Raghav

Course: IST 597 - Deep Learning

Github Link:

https://github.com/ashwathraghav/IST597-Assignment3.git

Assignment 3

- Ashwath Raghav

Introduction:

In this assignment, we were asked to build a custom optimizer which is similar to Adam but had an extra momentum vector to converge faster. The model was trained with 2 unique datasets MNIST_784 and Fashion-MNIST. Experiments were conducted to compare the custom optimiser and other optimisers available in Keras and to compare the custom optimiser with default model and model that has been regularised using L2 regularisation technique to better understand how the addition of an extra moment vector improves the test and validation set accuracy. Graphs were plotted to compare the test accuracy of the different models.

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 keras.datasets.load_data() this time. 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 128.

5. We now have to split the dataset into training, validation and testing dataset. 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. It is a 3 layer multi layer perceptron model and it achieved test accuracy of 97%. The final train and validation set accuracy at the end of 20 epochs reached:

Train Accuracy: 0.9963, Validation Accuracy: 0.9753

Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, 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. This brought down the test accuracy to 77%. The final train and validation set accuracy at the end of 20 epochs reached:

   Train Accuracy: 0.7831, Validation Accuracy: 0.7719

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU

3. 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 3%. The test accuracy reached 94% with L1 regularization.

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1 Penalty, Epochs = 20.

4. Employing L2 Penalty Regularisation technique during Backward Pass:

   Employed Regularisation technique, L2 Penalty to improve accuracy. I observed that accuracy did not change. The test accuracy reached 97%. Thus, the best regularisation technique to employ for our dataset is Ridge Regression/L2 regularisation technique.

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L2 Penalty, Epochs = 20.

5. 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 6%. The test accuracy reached about 91% with L1+L2 regularisation.

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1+L2 Penalty, Epochs = 20.

Best Reg Technique that does not bring down the accuracy: L2. But since performing L2 regularisation/Ridge Regression does not improve the accuracy a lot, we will stick with the normal model to perform the other experiments.

6. Employing Different Optimisers to improve the accuracy:

   Performed Optimiser Tuning by employing different optimizers such as SGD, Adam and RMSprop. I also designed a custom optimizer similar to Adam but with an extra momentum vector. Through experiments, I've proved that my custom optimizer worked better than the existing Keras Optimizers.

   Exp1: Choosing Adam as the optimizer in Backward Pass:

   Hyper Parameters: Epochs = 20, Batch Size = 128, Learning Rate = 1e-4, Optimizer = Adam, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 96%.

   After 20 epochs:

   Train Accuracy: 0.9696, Validation Accuracy: 0.9638

   Exp2: Choosing SGD as the optimizer in Backward Pass:

   Hyper Parameters: Epochs = 20, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 97%

   After 20 epochs:

   Train Accuracy: 0.9967, Validation Accuracy: 0.9746

   Exp3: Choosing RMSProp as the optimizer in Backward Pass:

   Hyper Parameters: Epochs = 20, Batch Size = 128, Learning Rate = 1e-4, Optimizer = RMSProp, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 96%'

   After 20 epochs:

   Train Accuracy: 0.9776, Validation Accuracy: 0.9659

   Exp4: Choosing Custom Optimizer as the optimizer in Backward Pass:

Hyper Parameters: Epochs = 20, Beta1 = 0.9, Beta2 = 0.999, Beta3 = 0.999987, Epsilon = 1e-8, Alpha/Eta = 0.01, Batch Size = 128, Optimizer = Custom Optimizer with extra momentum vector, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 98%

After 20 epochs:

Train Accuracy: 0.9942, Validation Accuracy: 0.9744

Conclusion:

1. Comparing Custom Optimizer vs other Optimizer:
    - The Custom Optimizer performed better than the other existing Keras Optimizers. The Custom Optimizer could reach a test accuracy of 98%. 1-2% better than SGD, Adam and RMSProp

2. Comparing Custom Optimizer vs Original Model:
    - The Original Model with SGD Optimizer could achieve a 97% test accuracy while the model with custom optimizer could achieve a 98% test accuracy. So Custom Optimizer is better.

3. Comparing Custom Optimizer vs Original Model with L2 Regularization:
    - The Original Model with SGD Optimizer and L2 regularisation could achieve a 97% test accuracy while the model with custom optimizer could achieve a 98% test accuracy. So Custom Optimizer is better.

Core Experiment 2:

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

Data Preprocessing/Feature Engineering:

1. The dataset Fashion-MNIST is retrieved using keras.datasets.load_data() this time. 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.

Assignment 3

- Ashwath Raghav

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 128.

5. We now have to split the dataset into training, validation and testing dataset. 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. It is a 3 layer multi layer perceptron model and it achieved test accuracy of 87%. The final train and validation set accuracy at the end of 20 epochs reached:

   Train Accuracy: 0.9187, Validation Accuracy: 0.8792

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, 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. This brought down the test accuracy to 64%. The final train and validation set accuracy at the end of 20 epochs reached:

   Train Accuracy: 0.6783, Validation Accuracy: 0.6419

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, Softmax Activation at Output Layer, Activation Fn = ReLU

3. 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 3%. The test accuracy reached 83% with L1 regularization.

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1 Penalty, Epochs = 20.

4. Employing L2 Penalty Regularisation technique during Backward Pass:

Employed Regularisation technique, L2 Penalty to improve accuracy. I observed that accuracy did not change. The test accuracy reached 87%. Thus, the best regularisation technique to employ for our dataset is Ridge Regression/L2 regularisation technique.

Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L2 Penalty, Epochs = 20.

5. 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 9%. The test accuracy reached about 78% with L1+L2 regularisation.

   Hyper Parameters: Layers = 3, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Regularizer = L1+L2 Penalty, Epochs = 20.

Best Reg Technique that does not bring down the accuracy: L2. But since performing L2 regularisation/Ridge Regression does not improve the accuracy a lot, we will stick with the normal model to perform the other experiments.

6. Employing Different Optimisers to improve the accuracy:

   Performed Optimiser Tuning by employing different optimizers such as SGD, Adam and RMSprop. I also designed a custom optimizer similar to Adam but with an extra momentum vector. Through experiments, I've proved that my custom optimizer worked better than the existing Keras Optimizers.

   Exp1: Choosing Adam as the optimizer in Backward Pass:

   Hyper Parameters: Epochs = 20, Batch Size = 128, Learning Rate = 1e-4, Optimizer = Adam, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 86%.

   After 20 epochs:

   Train Accuracy: 0.8850, Validation Accuracy: 0.8705

Exp2: Choosing SGD as the optimizer in Backward Pass:

Hyper Parameters: Epochs = 20, Batch Size = 128, Learning Rate = 0.1, Optimizer = SGD, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 87%

After 20 epochs:

Train Accuracy: 0.9187, Validation Accuracy: 0.8792

Exp3: Choosing RMSProp as the optimizer in Backward Pass:

Hyper Parameters: Epochs = 20, Batch Size = 128, Learning Rate = 1e-4, Optimizer = RMSProp, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 84%

After 20 epochs:

Train Accuracy: 0.8721, Validation Accuracy: 0.8577

Exp4: Choosing Custom Optimizer as the optimizer in Backward Pass:

Hyper Parameters: Epochs = 20, Beta1 = 0.9, Beta2 = 0.999, Beta3 = 0.999987, Epsilon = 1e-8, Alpha/Eta = 0.01, Batch Size = 128, Optimizer = Custom Optimizer with extra momentum vector, No Softmax Activation at Output Layer, Activation Fn = ReLU, Accuracy = 85%

After 20 epochs:

Train Accuracy: 0.8910, Validation Accuracy: 0.8590

Conclusion:

1. Comparing Custom Optimizer vs other Optimizer:
    - The Custom Optimizer performed better than the other existing Keras Optimizers. The Custom Optimizer could reach a test accuracy of 85%. 1-2% lower than SGD and Adam, but better than RMSProp.
    - For this dataset, SGD is a better optimizer because it achieved a better accuracy than other optimizers.

Assignment 3

- Ashwath Raghav

2. Comparing Custom Optimizer vs Original Model:
    - The Original Model with SGD Optimizer could achieve 87% test accuracy while the model with custom optimizer could achieve 85% test accuracy. So SGD Optimizer is better. (Original Model)

3. Comparing Custom Optimizer vs Original Model with L2 Regularization:
    - The Original Model with SGD Optimizer and L2 regularisation could achieve 87% test accuracy while the model with custom optimizer could achieve 85% test accuracy. So SGD Optimizer with L2 Regularization is better.