

Deep Learning Homework – 3

Name: Vrushabh Desai, Rishabh Chadha

Q3. We had four different parameters which we varied in the following specific range:

- Number of Epochs: [50,100,150,**200**]
- Batch size: [**1000**,2000,3000,4000]
- Learning rate: [**0.1**,0.01,0.001,0.0001]
- Alpha: [0.1,0.01,**0.001**,0.0001]

The best test accuracy that we achieved with this highlighted hyper parameter:

Test Accuracy	91.92
Validation Accuracy	92.46
Training Accuracy	91.972
Test Loss	0.292649
Validation Loss	2.139
Train Loss	0.4127

On further tuning of the hyperparameters we noticed that keeping the Batch size: **1000**, Learning rate: **0.1** and Alpha: **0.001** constant and varying Number of Epochs, the loss on validation set kept reducing and test accuracy increased.

Case 1: Epoch Number: 200 Test Accuracy: 91.92

The screenshot shows the Spyder Python IDE with a file named `homework3.py` open. The code defines a function `train` that takes a model `W`, training data `X_tr`, training labels `y_tr`, validation data `X_val`, validation labels `y_val`, test data `X_te`, test labels `y_te`, and a parameter `W`. It trains the model for 200 epochs with a batch size of 1000, a learning rate of 0.1, and an alpha of 0.001. The code prints the training and validation loss, accuracy, and test loss and accuracy.

```
105 acc = accuracy(Xtr[start:stop,:],ytr[start:stop,:],w)
106 print(" Training, Loss: {} Epoch Number: {} Batch Number: {}".format(Loss,_,b))
107 print(" Accuracy: {}".format(a))
108
109 Loss = 0 # This is to avoid the overflow of the variable loss
110 return W
111
112
113 X_tr = np.load("H:\Masters Study\Deep Learning\Home Work\HW3\dataset\mnist_train_images.npy")
114 y_tr = np.load("H:\Masters Study\Deep Learning\Home Work\HW3\dataset\mnist_train_labels.npy")
115 X_val = np.load("H:\Masters Study\Deep Learning\Home Work\HW3\dataset\mnist_validation_images.npy")
116 y_val = np.load("H:\Masters Study\Deep Learning\Home Work\HW3\dataset\mnist_validation_labels.npy")
117 X_te = np.load("H:\Masters Study\Deep Learning\Home Work\HW3\dataset\mnist_test_images.npy")
118 y_te = np.load("H:\Masters Study\Deep Learning\Home Work\HW3\dataset\mnist_test_labels.npy")
119
120 X_tr = np.c_[np.ones((X_tr.shape[0],1)),X_tr]
121
122 W = np.random.rand(len(X_tr[-1]),len(y_tr[-1]))
123
124 print("Shape of X_tr:{}".format(X_tr.shape))
125
126 W = log_loss(Phot,ytr,W,0.001)
127
128 W, W, W, W, W = train(W,X_tr,ytr,epoch = 200,mini_batch_size = 1000,alpha = 0.001,learning_rate= 0.1)
129
130 W = normalize_2(X_val,W)
131 loss_val = log_loss(Yhat_val,yval,W,0.01)
132 print(" Validation Loss: {}".format(loss_val))
133
134 Yhat_test = normalize_2(X_te,W)
135 loss_test = log_loss(Yhat_test,yte,W,0)
136 print(" Test Loss: {}".format(loss_test))
137
138 Acc_train = accuracy(X_tr,ytr,W)
139 Acc_val = accuracy(X_val,yval,W)
140 Acc_test = accuracy(X_te,yte,W)
141 print("Validation Accuracy: {} \nTrain Accuracy: {} \n".format(Acc_val,Acc_test))
```

The console output shows the training progress for 200 epochs. The final output is:

```
Training, Loss: 0.5367369621566116 Epoch Number: 199 Batch Number: 40
Training, Loss: 0.49782648989383814 Epoch Number: 199 Batch Number: 41
Training, Loss: 0.5007484669263491 Epoch Number: 199 Batch Number: 42
Training, Loss: 0.43362799773540286 Epoch Number: 199 Batch Number: 43
Training, Loss: 0.5619171853168514 Epoch Number: 199 Batch Number: 44
Training, Loss: 0.5220499856321644 Epoch Number: 199 Batch Number: 45
Training, Loss: 0.44629159518041394 Epoch Number: 199 Batch Number: 46
Training, Loss: 0.5596868891651691 Epoch Number: 199 Batch Number: 47
Training, Loss: 0.469680861361894483 Epoch Number: 199 Batch Number: 48
Training, Loss: 0.47995563615180767 Epoch Number: 199 Batch Number: 49
Training, Loss: 0.4384086335181545 Epoch Number: 199 Batch Number: 50
Training, Loss: 0.43726846367981203 Epoch Number: 199 Batch Number: 51
Training, Loss: 0.44382192193546477 Epoch Number: 199 Batch Number: 52
Training, Loss: 0.35934148081312556 Epoch Number: 199 Batch Number: 53
Training, Loss: 0.41274693788989913 Epoch Number: 199 Batch Number: 54
Validation Loss: 2.1390601057252256
Test Loss: 0.2926499350333888
Validation Accuracy: 92.46
Test Accuracy: 91.92
Train Accuracy: 91.97272727272727
```

The variable explorer shows the following variables:

Name	Type	Size	Value
Acc_test	Float	1	91.92
Acc_train	Float	1	91.97272727272727
Acc_val	Float	1	92.46
Loss_test	float64	1	0.2926499350333888
Loss_val	float64	1	2.1390601057252256
W	Float64	(784, 10)	[[0.72012037 0.59224797 0.01882312 ... 0.53985333 0.9830322 0.9784943 ...

Case 2: Epoch Number: 100 Test Accuracy: 91.81

The screenshot shows the Spyder Python IDE with a file named `homework2.py` open. The code is a neural network training script for MNIST. The console output shows training progress from Epoch 79 to 100. At Epoch 100, the training loss is 0.82654864642416, validation loss is 5.9140118891431275, and test accuracy is 91.81. The variable explorer shows the following values:

Name	Type	Size	Value
Acc_test	float	1	91.81
Acc_val	float	1	91.82000000000001
Loss_test	float64	1	5.9140118891431275
Loss_val	float64	1	5.91109390939557
W	float64	(784, 10)	[[0.65569325 0.38488161 0.65993065 ...]]
X_te	float32	(10000, 784)	[[0. 0. 0. ... 0. 0. 0.]]

Case 3: Epoch Number: 80 Test Accuracy: 91.59

The screenshot shows the Spyder Python IDE with the same `homework2.py` file. The console output shows training progress up to Epoch 80. At Epoch 80, the training loss is 0.82654864642416, validation loss is 5.9140118891431275, and test accuracy is 91.59. The variable explorer shows the following values:

Name	Type	Size	Value
Acc_test	float	1	91.59
Acc_val	float	1	91.82000000000001
Loss_test	float64	1	5.9140118891431275
Loss_val	float64	1	5.91109390939557
W	float64	(784, 10)	[[0.65569325 0.38488161 0.65993065 ...]]
X_te	float32	(10000, 784)	[[0. 0. 0. ... 0. 0. 0.]]