MSML 603 Project 2

Colleen Aulton
Prashant Swarnapuri

February 7, 2021

1 Introduction

Both authors developed code independently and observations were combined subsequently. Implementation differences are expected. Differences where applicable are highlighted in the observations.

Colleen Aulton's observations will be referred to as Model 1, and Prashant Swarnapuri's observations will be referred to as Model 2.

2 Code Repository

Colleen Aulton (Model 1)
https://drive.google.com/drive/folders/1abGDDq7v9jUlLo0sRHjdQB-xVnJZy7kT
Prashant Swarnapuri (Model 2)
https://github.com/spprashant/msml603-project2 (Refer to README.md for details)

Both sets of code are expected to be run in a Jupyter Notebook environment using a Python 3 kernel.

3 MNIST Data Set

Model 1 - MNIST The MNIST data was split into training (60000) and testing (10000) sets by default and automatically labeled for each possible handwritten number.

Model 1 uses a set of linear layers ordered sequentially. The widths of the input and output layer are defaulted to the number of features in the dataset, while the widths and number of hidden layers are given as an input.

The training and testing data is then converted into a tensor format and loaded into PyTorch trainloader and testloader with a batch size given as an input. The neural network uses an SGD optimizer with a learning rate given as an input and a cross entropy loss criterion. The network loops through the training data in 10 epochs and calculates the loss and accuracy. The testing data is then fed into the network with the accuracy output from the function.

MNIST Dataset (Batch Size = 10, Learning Rate = 0.0001	
Hidden Layers: Number, Width	Accuracy
2, 500	15%
3, 500	17%
2, 1000	21%
3, 1000	23%
2, 1500	26%
3, 1500	29%

Figure 1: Neural Network Accuracy (MNIST Data set) - Model 1

For the MNIST dataset, the neural network achieved the highest testing set accuracy with a larger width and number of layers. All tests were performed with a batch size of 10 and learning rate of 0.0001. Accuracy was always improved by adding more hidden layers and significantly improved with each subsequent increase to the size of the hidden layers.

Model 2 - MNIST MNIST data was loaded from the torchvision library which is a part of the PyTorch package.

Model 2 uses a combination of convolutional layers and linear fully connected layers based on examples of other popular image recognition networks.

Layers:

- Convolution Layer inchannels=1, outchannels=10, kernelsize=(5, 5), stride=(1, 1)
- Max Pool 2D Layer - kernelsize=(2,2)
- Convolution Layer - inchannels=10, outchannels=20, kernelsize=(5, 5), stride=(1, 1)
- Convolution 2D Dropout Layer
- Max Pool 2D Layer - kernelsize=(2,2)
- Fully Connected Linear Layer - infeatures=320, outfeatures=50, bias=True
- Dropout Layer
- Fully Connected Linear Layer - infeatures=50, outfeatures=10, bias=True
- Softmax Layer

The training and testing data is then converted into a tensor format and loaded into PyTorch trainloader and testloader with a batch size given as an input. The neural network uses an SGD optimizer with different learning rates and a negative log likelihood loss criterion. The network loops through the training data for one epoch and calculates the loss and accuracy. The testing data is then fed into the network with the accuracy output from the function.

MNIST Data Set Momentum=0.5 Epochs=1		Batch Size		
		16	64	128
Learning Rate	0.01	93%	95%	94%
	0.001	94%	90%	89%
	0.0001	79%	56%	32%

Figure 2: Neural Network Accuracy (MNIST Data set) - Model 2

The model achieves higher accuracy at higher learning rate (0.01) with some difference observed between the batch sizes. As the learning rate decreases further the accuracy of the model drops, made even worse by larger batch sizes.

4 Poses Data Set

Model 1 - Poses Poses data set was read in from CSV files and combined in Python. The Poses dataset was labeled for each of the 68 subjects and Model 1 uses a test/train split of 50% each.

The training and testing data is then converted into a tensor format and loaded into PyTorch trainloader and testloader with a batch size given as an input. The neural network uses an SGD optimizer with a learning rate given as an input and a cross entropy loss criterion. The network loops through the training data in 10 epochs and calculates the loss and accuracy. The testing data is then fed into the network with the accuracy output from the function.

Poses Dataset Two hidden layers (width=1500)		Batch Size		
		10	50	100
Learning Rate	0.1	14%	25%	23%
	0.01	20%	22%	20%
	0.001	20%	19%	19%
	0.0001	62%	23%	24%

Poses Dataset Three hidden layers (width=1500)		Batch Size		
		10	50	100
Learning Rate	0.1	14%	18%	24%
	0.01	14%	22%	21%
	0.001	25%	21%	29%
	0.0001	65%	34%	41%

Figure 3: Neural Network Accuracy (Poses Data set) - Model 1

For the Poses dataset, the neural network achieved the highest testing set accuracy (65%) with a smaller batch size (10) and learning rate (0.0001). At every batch size, accuracy increased with every subsequent decrease to learning rate. At higher levels of learning rate, accuracy was less affected by batch size, varying by about 10%. The increased number of layers improved the highest accuracy by 3% (62% with two hidden layers vs. 65% with three hidden layers). The width of the layers also improved the accuracy only slightly, with fewer hidden layers having larger improvements. (59% with two 1000-width layers vs. 62% with two 1500-width layers; 64% with three 1000-width layers vs. 65% with three 1500-width layers)

Model 2 - Poses Data was loaded from the poses.mat file provided in project 1. Model 2 uses a test/train split of 76% to 24%. The first 10 sets for each subject were used as training set and other three were used for test.

The training and testing data is then converted into a tensor format and loaded into PyTorch trainloader and testloader with a batch size given as an input.

A few changes were made to the network previously designed for the MNIST dataset. - 1 more convolution layer was added, to make the network deeper. The network was made slightly wider to account for more variability in the data. The neural network uses an SGD optimizer with different learning rates and a cross entropy criterion. The network loops through the training data in 40 epochs and calculates the loss and accuracy. The testing data is then fed into the network with the accuracy output from the function.

Layers:

- Convolution Layer - inchannels=1, outchannels=20, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1)
- Max Pool 2D Layer - kernelsize=(2, 2)
- Convolution Layer - inchannels=20, outchannels=20, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1)
- Max Pool 2D Layer - kernelsize=(2, 2)
- Convolution Layer - inchannels=20, outchannels=32, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1)
- Fully Connected Linear Layer - infeatures=3840, outfeatures=600, bias=True
- Dropout Layer
- Fully Connected Linear Layer - infeatures=600, outfeatures=68, bias=True

Poses Data Set Momentum=0.9 Epochs=40		Batch Size		
		16	64	128
Learning Rate	0.01	1%	37%	41%
	0.001	54%	58%	54%
	0.0001	61%	33%	12%

Figure 4: Neural Network Accuracy (Poses Data set) - Model 2

The model achieves higher accuracy at a learning rate of 0.001 with some difference observed between the batch sizes. The maximum accuracy of the model is around 60%, with a maximum of 61% observed for learning rate 0.0001 and batch size of 16.