Daniel Kim, Chad Underhill, Spyridon Antonatos

CS4341 Project 3 Written Report

9/6/2017

Set of Experiments Performed

We experimented first with the given one hidden layer and the output layer. We used 784 nodes for the hidden layer, the activation function relu, and the kernel initializer glorot\_normal. The batch size for training was 512, and the number of epochs were 100. This resulted in around 89.5% training accuracy and 88.6% validation accuracy.

Next, we tried to double the number of epochs to 200. This resulted in 92.6% training accuracy and 88.9% validation accuracy. The training accuracy was much better than the last experiment, but the validation accuracy was not much different, and running the program with 200 epochs took much more time, so we decided to stick with 100 epochs.

After this, we tried changing just the batch size from 512 to 256. This ran a bit slower than the first experiment, but the resulting training accuracy was 91.9% and the validation accuracy was 90.8%. This was overall much better than the first experiment, so we tried decreasing the batch size further to 128. However, this yielded only a 2% increase in training accuracy and a 1% increase in validation accuracy. A batch size of 64 only increased the training accuracy by 2%, so we decided to stick with 128 as our batch size.

Changing the activation function to selu resulted in a training accuracy of 92.8% and a validation accuracy of 88.7%. This was worse than the relu activation function. We also tried the tanh activation function which gave a training accuracy of 91% and a validation accuracy of 90.7%. The relu activation function performed the best of these, so we decided to stick with relu as our activation function.

For our final experiment using only one hidden layer, we tried changing the weight initialization scheme, the kernel initializer. Originally, it was set to glorot\_normal, which used a truncated normal distribution centered on 0 using the number of input and output units. We changed it to he\_normal, which is the same as glorot\_normal, except it does not use the number of output units. This resulted in a very slight(< 1%) increase in both accuracies, so we tried random\_normal next, which initializes weights with a normal distribution. This decreased the accuracies. We tried many more weight initialization schemes, but these generally gave equal or worse accuracies.

After testing with one hidden layer, we decided to test two hidden layers. We added a layer with 784 nodes using relu as the activation and glorot\_normal as the weight initialization. This did not yield any change in the accuracies. We tried changing the number of nodes in the second hidden layer to 196. This gave a slight increase in both accuracies. At this point our training accuracy was 96.5% and our validation accuracy was 91.8%. Next, we tried decreasing the number of nodes in the second layer even further to 49, which gave no increase in accuracy.

We also tried adding a third hidden layer. We kept the second hidden layer with 196 nodes using relu and glorot\_normal and added a third layer with the same fields. This resulted in a training accuracy of 97.2% and a validation accuracy of 92.5%, which was a slight increase. Using 49 nodes in the third layer increased the validation accuracy to 93.7%.

\*\*\*Write about best model\*\*\*

current: 3 layers → 1st layer 784 nodes, 2nd layer 196 nodes, 3rd layer 49 nodes

(relu, glorot\_normal)

**Model & Training Procedure Description**

The most accurate model we were able to obtain was using:

model structure

# of layers

# of neurons in each layer

weight initialization scheme

activation function

# of epochs for training

batch size for training

**Graph**

epoch vs training set accuracy

epoch vs validation set accuracy

**Model Performance & Confusion Matrix**

confusion matrix

overall accuracy of model

Look at problem specifications for specific format

Visualization

3 visualizations of misclassified images by best model

observations on why misclassified

\*\*\*find how to backtrack to image in data that was misclassified\*\*\*