EECE6036 HW5

Jack Fliegel

# Problem 1

**System Specification**

As specified in the Homework pdf, I used the same network as in Problem 1 of HW4 but applied pretrained weights (labeled pretrained\_weights.txt in /HW5\_Datafiles) from the autoencoder network to the input/hidden layer weights. Case I did not train these weights and Case II did train the weights.

**Results**

A graph with a number of error fractions

Description automatically generated

Figure : Error Fractions vs Epoch HW5

A graph with a line graph

Description automatically generated with medium confidence

Figure : HW4 Error Fractions vs Epoch

A chart of training and prediction

Description automatically generated with medium confidenceA chart of training set case ii

Description automatically generated

Figure : Confusion Matrix Training Set (Cases I (a) and II (b))

A chart with a yellow line

Description automatically generatedA chart with a yellow line

Description automatically generated

Figure : Confusion Matrix Test Set (Cases I (a) and II (b))

**Analysis**

*Q1: Did initializing the hidden weights from the autoencoder make training go faster in this this homework compared to that in HW 4 (for both Case I and Case II)?*

Initializing the hidden weights made the training slower and fail to classify for Case I and Case II compared to HW4. As we can see in figure 1 the model has a higher error fraction within 5 epochs compared to HW4’s model where the weights were set at random (not pretrained).

With these results we can assume that the model failed to classify for both cases I and II. I do not believe that the autoencoder found relevant features that could be applied to the model.

*Q2: Did training both layers (Case II) substantially improve performance over training only the output*

*layer (Case I)?*

There was not any significant changes from Case I vs Case II when comparing performance of each. Both seemed to have poor performance and failed to classify in different ways as seen in figures 3 and 4. The accuracy of both cases stayed around 15% which makes sense because the model was failing and only predicting one output for every input.

# Problem 2

**System Description**

The learning rate for this network was 0.3 (Smaller than previous networks as suggested by homework pdf). The network makes 5 passes (epochs) through the entire training dataset which costs a significant amount of time compared to other networks. Figure 5 is the sudo code for the SOFM network as given in Lecture 11 of this class and used in this network.

**A screenshot of a cell phone

Description automatically generated**

Figure : Lecture 11 SOFM Training Algorithm

**Results**

A group of squares with different shades of orange and black

Description automatically generated

Figure : SOFM heatmap of each digit 0-9

A number in a row

Description automatically generated with medium confidence

Figure : features found for each neuron

A screenshot of a number grid

Description automatically generated

Figure : Lecture 11 SOFM results

**Discussion**

Figure 5 shows that for each digits class there are clear and distinct areas of classification. Each map is scaled differently but we can still see patterns arise and some neurons for certain digit classes ‘win’ more than others. The majority of the neurons for each class did not win at all indicated by the dark areas of each heatmap. The strongest winning neurons found in the particular trial run in Figure 5 was in class 1 where 2 neurons won about 50% of the time.

In Figure 7 we see the features of each individual neuron produced by the SOFM network. The images resemble variations of the digits from the training set of digits and it appears that relevant features were found. Figure 8 shows the expected output of a trained SOFM network and comparing it to the results in Figure 7, we can see that my network did not train on the data all that well. The weights are not close to their actual digits and are not legible.

# Problem 3

**System Description**

For Problem 3, I used the same network I used for Problem 1 of this homework but instead of using the pretrained weights from the autoencoder I initialized the weights for the network to the SOFM weights. This means I had to change the hidden layer neurons to 144 to fit the SOFM weights but kept the output neurons the same at 10 so that it could attempt to classify the 10 digits.

**Results**

**A chart with a yellow line

Description automatically generated**

Figure : Test data through SOFM pretrained network

**Discussion**

After training the classifier network using the features found in the Self Organized Feature Map (SOFM) network, we can see that the classifier failed to produce any useful classification. This may be because the features it found were irrelevant to separate the classes or the weights were too far off for the network to update ( I did not change the number of epochs or learning rate). This is similar to the performance of Problem 1 of this homework and the confusion matrices in figures 3 & 4 where the network failed and guessed one output for every input that was sent through the network. Comparing the results in Figure 9 to Problem 1 of HW4 where the confusion matrix was almost a perfect identity matrix, which means the model classified the digits very well, the performance for these SOFM pretrained is poor.