Deep learning exercise 2

Olof Markstedt

November 21, 2016

A

The implementation for the neural network in this task is in the testImageRec.m script.

For this task I reshaped the data which is described and done in the setVariables.m script so for the script testImageRec.m to run, one has to run the setVariables.m script first. The target matrix was created in a way that divided up all the threes from the other numbers. So I looped through all the vectors in the target matrix and created a new matrix based on the previous one. This was done by checking if the value in row number 3 was equal to 1, if this was true then a row vector of size 1x1 with the value 1 in it and appended this to a matrix, else a row vector of size 1x1 with the value 0 was appended to the same matrix. By doing this the confusion matrix created was of size 2x2 and not 10x10 which the original target matrix would have created. The network I used was a patternnet with default settings, the default setting for performance was cross-entropy and training optimization method was scaled conjugate gradient.

Figure 1: Confusion matrix, created using a classifier which detects images containing the digit 3 and the remaining ones. 4489 images classified as ”not a 3”, 111 images are incorrectly classified as ”not a 3”, 12 images are incorrectly classified as being a 3.

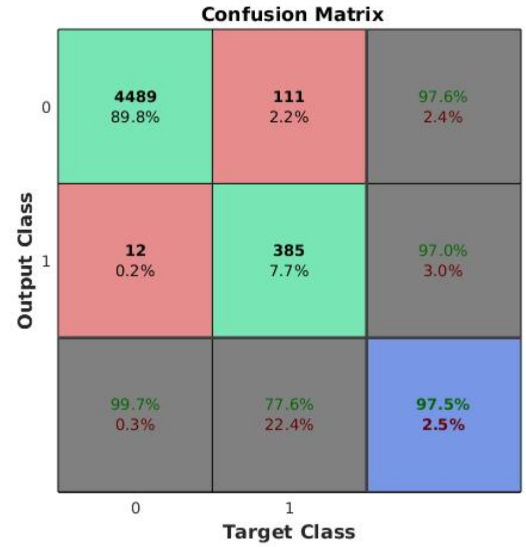
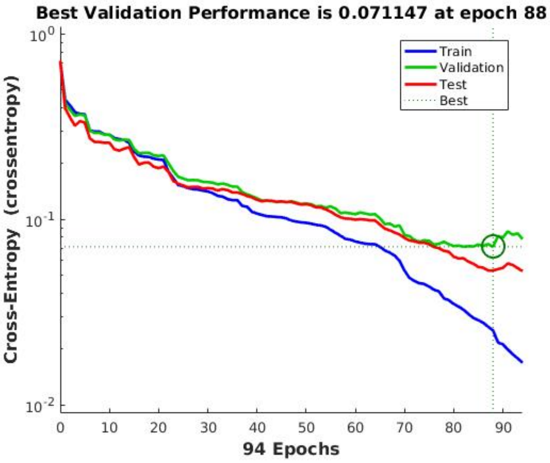


Figure 2: Performance plot that plots error vs. epoch for the training, validation and test performances of the training record. The best performance is taken from the epoch with the lowest validation error.



In total 97.5% of the data was classified correctly and 2.5% was classified incorrect.

B

i)

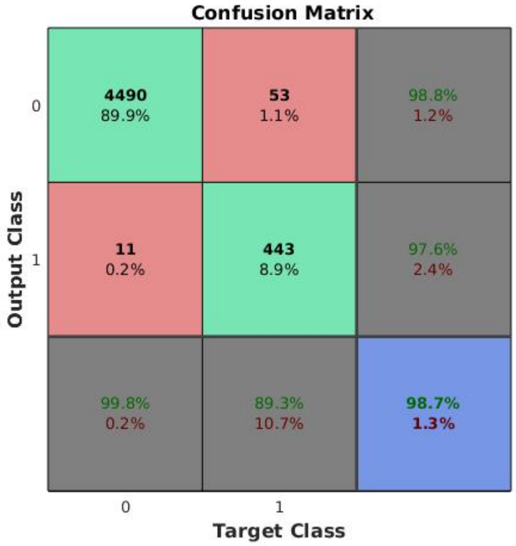
For this task I changed the network performance from cross-entropy to mean squared error. The performance function change did not contribute to anything that changed the output a lot. However the performance measured by using the "perform" method significantly decreased when using mean squared error in comparison to cross-entropy. Cross-entropy returned a performance result of around 0.045 and mean squared error return a result of around 0.015 thus mean squared error improves the performance of the network in terms of estimating the target values. However in the documentation cross-entropy is the better choice most of the times. The mean squared error is calculated in the following way: $MSE = 1/n \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 = 1$. The mean squared error measures the average of the squares of the errors which is the difference between the estimator and what is estimated. The mean squared error can be calculated in the above fashion if Ŷ is a vector containing n predictions and Y is the vector of observed values corresponding to the inputs of the function that generated the predictions. It basically calculates the mean difference of the computed outputs and the targets (desired).

Cross-entropy is calculated in the following way: $CE = -\sum_i y'_i ln(y_i)$ where y_i is the predicted probability and y'_i is the true probability of that class (i.e the target).

ii)

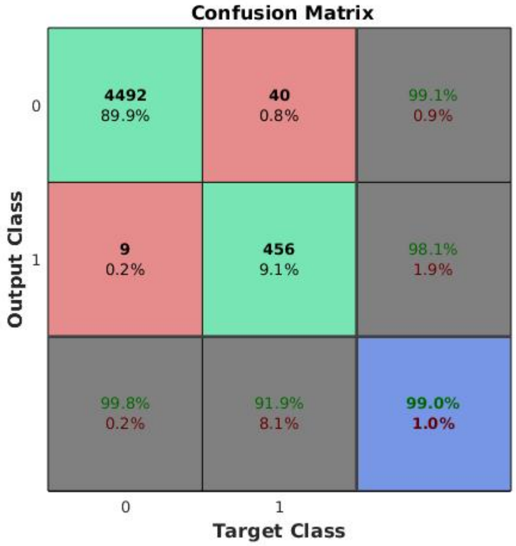
By changing the layer size the network performance does not increase or decrease significantly nor does the classification error.

Figure 3: Confusion matrix using a hidden layer size of 10 and default settings of a patternnet. The overall error is 1.3% wrong classified images. The number of threes correctly classified are 97.6%.



	0	1	
0	**4490** 89.9%	**53** 1.1%	98.8% 1.2%
1	**11** 0.2%	**443** 8.9%	97.6% 2.4%
	99.8% 0.2%	89.3% 10.7%	**98.7%** **1.3%**

Figure 4: Confusion matrix using a hidden layer size of 25 and default settings of patternnet. The overall error is 1.0% wrong classified images. The number of threes correctly classified are 98.1%.

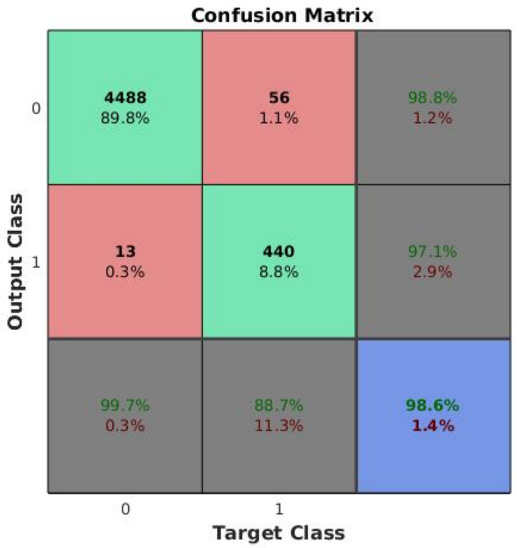


As can be seen when comparing figures 1, 3 and 4 the number of correctly classified images does not significantly increase or decrease.

C

The input weights is a matrix of size 25x784, and the layer weights are two matrices of size 10x25 and 1x10. The net biases are 2 matrices of size 25x1, 10x1 and a matrix of size 1x1 with a single value. The performance does not change when using two hidden layers of size 25 and 10. See figure 5 and compare to the other confusion matrices. The time it takes for the network to run 1000 iterations is 1min and 2 sec, however this was completed by ”forcing” it to actually run 1000 iterations by changing the network parameters. This is not optimal since the network will be overfitted and thus the network does not ”generalize” well meaning it will perform poorly on other data than the data used for training.

Figure 5: Confusion matrix using the network architecture: 784-25-10-1.



D

The entire code for this task is in autoencoderNetwork.m.

i)

The computing time for the tasks A, B and C was around 5 seconds (very fast). For the script in task D the time was around 3 minutes. However the performance has increased a lot and the classification has an error of only 0.1% in total, see figure 6.

Figure 6: Confusion matrix using the trained autoencoders and the following network architecture: 784-25-10-1.

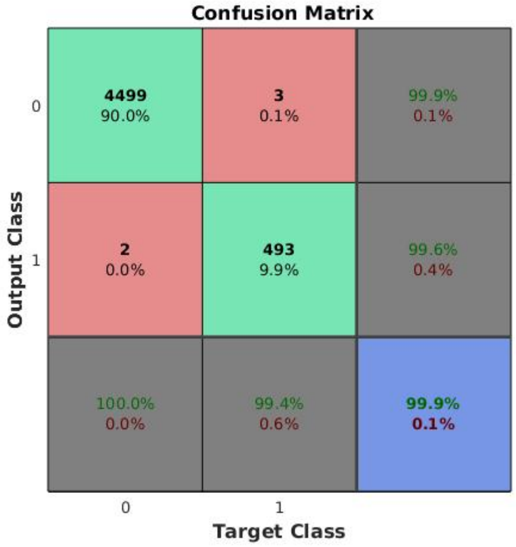
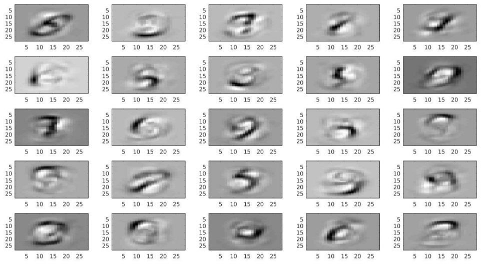


Figure 7: Weights of the first hidden layer. The values on the y-axis and x-axis are the pixels (the picture size is 28x28).



The figures in figure 7 are representing the most important features that the network uses to identify the digits within the pictures. The magnitude of the weights determines the importance of that feature, so if a weight has a large magnitude then that feature will have a high impact when during classification.

iii)

I have not been able to solve this.

E

I have not been able to solve this.

F

Most parts of this exercise has been very interesting and fun to solve, however I felt that D iii and E were very difficult. I think the relevance of the tasks were high and I would much like to learn how to completely solve D iii and E, it's probably very important to know if I want to become good at deep learning. I have understood how difficult deep learning is to master since I figured that the exercises so far has only been scratching the surface of what deep learning is. There is a lot of mathematics behind it that I have had to read up on to refresh my memory, anyhow I need to further refresh my memory regarding a lot of theory and the mathematics behind neural networks. That said I still feel that these exercises (so far) are increasing in difficulty in a ”balanced” way, however it would have been nice to get some more tips on how to solve D iii and E, maybe I'm expected to be able to solve it but I had trouble understanding the question. For improvements I think it would be better if some of the tasks were more informative, especially D iii and E, but most of the tasks were understandable once I read through them a couple of times.