Andrew Robbertz alrobbertz@wpi.edu
CS 4341 A18
09/10/2018

MINST Data Classification Using Artificial Neural Networks

Software

Python:	2.7.15
Keras:	2.2.2
Tensorflow:	1.10.1
Scikit-Learn:	0.19.2
Scipy:	1.1.0

Setup

Unless otherwise noted in the test description, assume the following options. I may refer to this as the default model.

Model

- Input & Hidden Layer Activation: relu
- Input Layer Shape: (784, 0)
- 10 Hidden layers: 50 nodes/layer
- Output Layer: 10 nodes
- Output Layer Activation: softmax

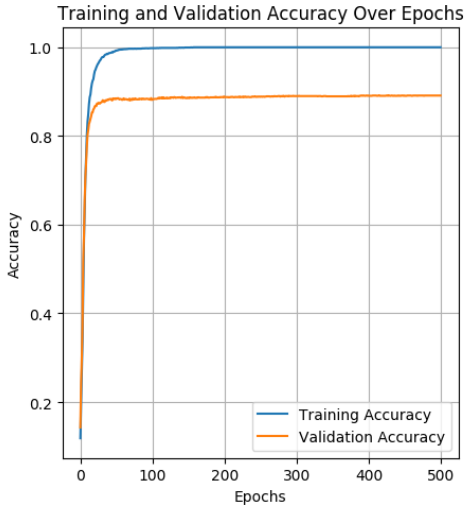
Learning

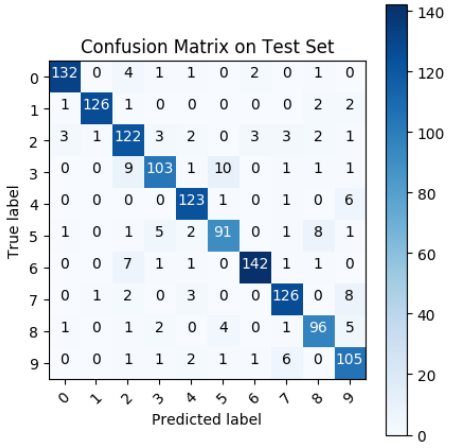
- Optimizer: stochastic gradient descent (sgd)
- Learning Rate: 0.001
- Loss Function: categorical cross-entropy
- Epochs: 500
- Batch Size: 512

Evaluating ANN Performance with and without Dropout

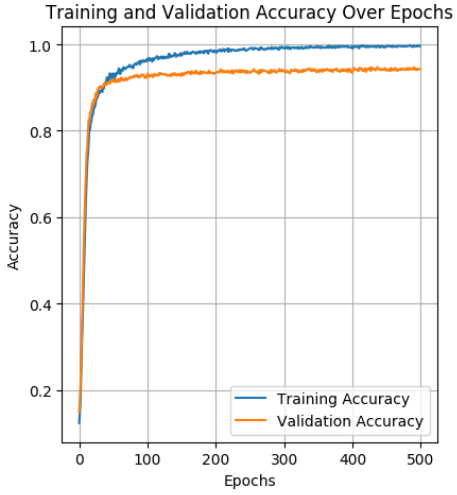
For this test I split 80% of the data for the training and validation sets. The validation set was set to be 1/3 of the training set. The remaining 20% was used in the test set. I tested the default model against a test model with 20% dropout in the input layer. Below is the training and validation accuracies for the default model as well as the confusion matrix on the test set.

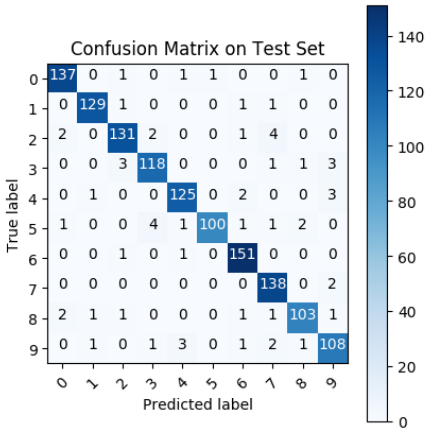
Default Model





Test Model With Dropout



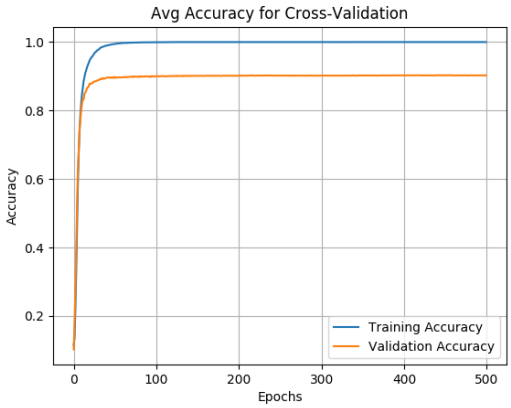


The default model achieved 86.7% accuracy on the test set and 89.1% accuracy on the validation set. The test model with dropout achieved 95.3% accuracy on the test set and 94.3% accuracy on the validation set. Comparing the validation accuracy for the two models, I calculated a t-statistic of -8.85 and p-value of 3.99e-18. Based on these results we can conclude the test model with dropout significantly outperforms the default model with a strong 95% confidence level. This suggests that the default model was likely overfitting data in the training sets to a small degree. The confusion matrix is also noticeably better with dropout than without, suggesting that we may have been overfitting.

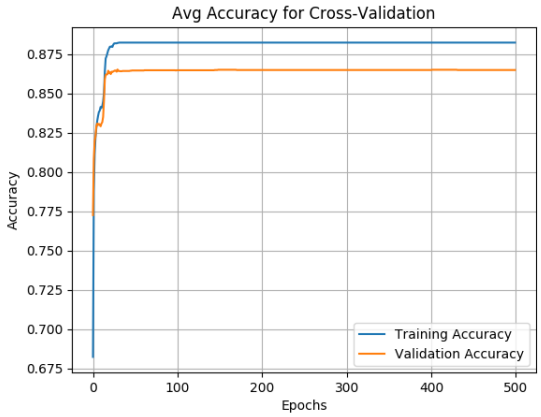
Evaluating ANN Performance with Varying Hidden Layers

I first compared the default model with a test model having only two hidden layers rather than ten. Second, I compared the 2-layer model with a 1-layer model having only one hidden layer. Using 3-fold cross-validation the following results were found tracking training and validation accuracy over each fold.

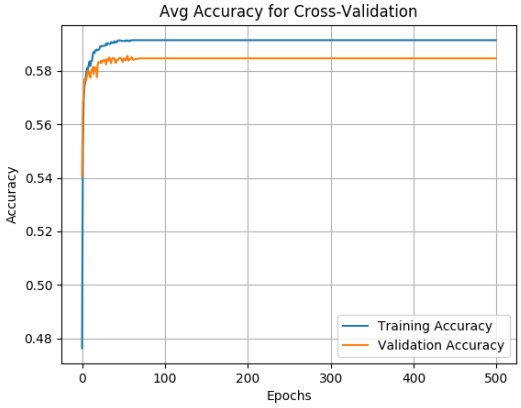
Default Model



2-Layer Model



1-Layer Model

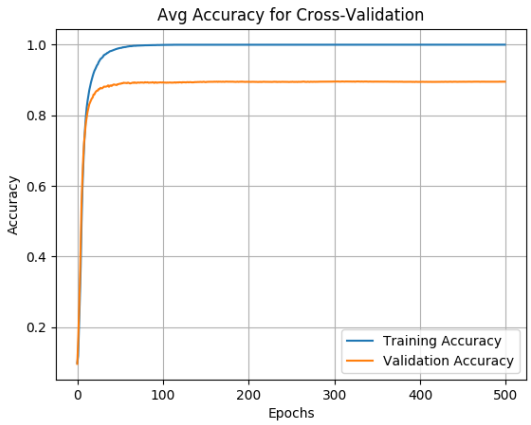


As seen in the data, the default model clearly outperforms the 2-layer model. Note the differences in scale for each of the figures. The default model achieved an average 90.3% accuracy across the three folds, the 2-layer model achieved 86.5% accuracy across the three folds, and the 1-layer model achieved 59.1% accuracy across the three folds. Comparing the average accuracy of the default and 2-layer models, I calculated a t-statistic of 9.65 and p-value of 4.11e-21. Based on these results we can conclude the default model significantly outperforms the 2-layer with a strong 95% confidence level. Comparing the average accuracy of the 2-layer and 1-layer models, I calculated a t-statistic of -20.1 and p-value of 1.11e-75. Based on these results we can conclude the 2-layer model significantly outperforms the 1-layer with a strong 95% confidence level. These results are not surprising as the default model simply has more parameters to lean on, giving it a significant advantage. The 2-layer model beats the 1-layer model for the same reason.

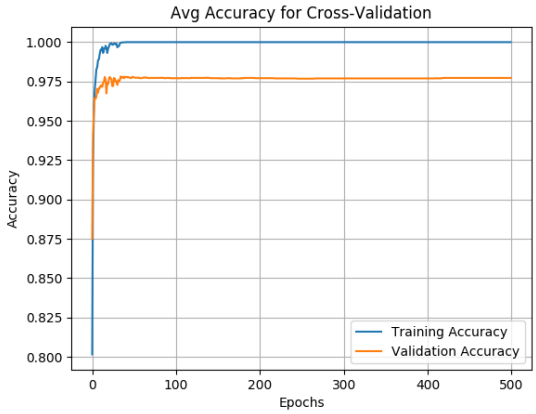
Evaluating ANN Performance with Varying Batch Sizes

Here compared the default model (batch size 512) with a test model having batch size 32. Using 3-fold cross-validation the following results were found tracking training and validation accuracy over each fold.

Default Model



Small-Batch Model



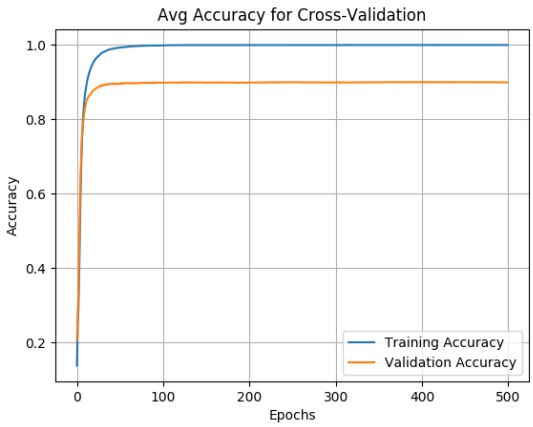
The default model achieved an average 89.5% accuracy across the three folds, while the small-batch model achieved an average 97.7% accuracy. Doing statistical analysis, I calculated a t-statistic of -29.6 and p-value of 7.86e-129. Based on these results we can conclude the small-batch model significantly outperforms the default model with a strong 95% confidence level.

This is not surprising because smaller batch sizes do help to fine-tune the network better than larger batches. This can also be seen as both training and validation accuracies plateau earlier with a batch size of 32 than with 512. The main downside of running smaller batches is that computation time increases significantly because more batches are needed per each epoch. The performance gains are non-trivial; however, it may not be worthwhile to decrease batch size on sufficiently large datasets where computation time will be impacted most.

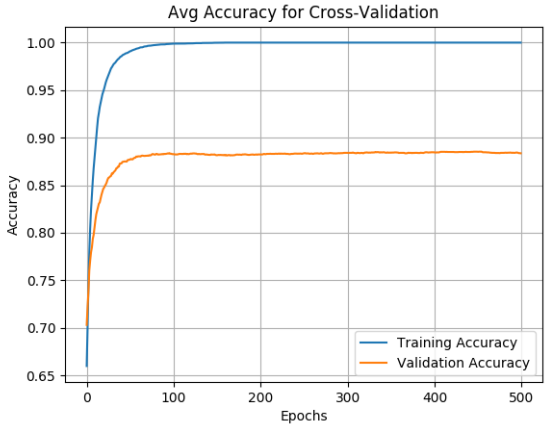
Evaluating ANN Performance with Varying Activation Functions

It was not clear to me how slightly varying activation functions can have such a large impact on ANN performance, therefore I did one final experiment to see how the accuracy might vary. I compared the default model, using the relu function, against a test model using the tanh activation function. Using 3-fold cross-validation the following results were found tracking training and validation accuracy over each fold.

Default Model



tanh Model



The default model achieved an average 90.0% accuracy across the three folds, while the tanh model achieved an average 88.4% accuracy. Comparing the average accuracy of the default and tanh models, I calculated a t-statistic of 5.03 and p-value of 5.91e-07. Based on these results we can conclude the default model significantly outperforms the tanh model with a 95% confidence level. Many thanks to professor Heffernan for recommending we initially use the relu function, as it clearly is the better of the two.