Nikhil Chakravarthy

Problem 2 – Effects of Dropout

Evaluate the effect of different dropout rates on a neural network.

Problem description:

I have taken images from the Fashion MNIST database, representing 10 possible different classes of clothing images. I then create separate networks with dropout values ranging from 0.0 to 0.9, with intervals of 0.1 on training sets from the database. Then, I test these networks on test data from the database, and collect the accuracies and losses of each network. I do this 3 times for each network, and collect the average accuracies and losses.

Each network has an input layer of the flattened Fashion MNIST image (28x28 to 1x784), hidden layers of sizes 50, 100, 100, 100 (where all have the dropout ratio given by the current experiment settings), and an output layer of size 10 for the 10 different classes possible.

The purpose of this is to learn if dropout rates can help prevent overfitting. This can be judged by comparing the accuracies of networks trained on data from the same domain (Fashion MNIST domain in this case), and seeing if some levels of dropout result in higher accuracies than no dropout at all.

Experiment:

I load 60000 images and labels for the training set and 10000 images and labels for the test set. I divide both sets into 30 different sets, because there are 110values of dropout from 0.0 to 0.9, and each network with a specific dropout value will be created and tested 3 times for a total of 30 different training and test sets needed. Using tensorflow and keras, I loop through the list of dropout values, and for each value, I create a network with one of the 30 training sets with the dropout ratio equal to the dropout value this iteration is working with. Then, I test the network on one of the 30 test sets, and store accuracy and loss. This is done 3 times for each possible dropout value.

Results (found in prob2/results/dropout_.csv)

These 10 values are the average accuracies of each network based on the test set, with the dropout values of the networks on the left side of the colon. From here, we can see that no dropout performed better than networks with dropout ratios of 0.4 or higher. However, networks with dropout values of 0.1 to 0.3 performed better than the network with no dropout, with the most accurate network being the 0.2 dropout ratio network with an accuracy of almost 0.82. Obviously, once the dropout ratio exceeded 0.5, the accuracy started to drastically reduce, since the network was becoming too sparse to properly learn.

0.0: 0.7999886117234855
0.1: 0.8109816404430008
0.2: 0.8190046333169411
0.3: 0.816981652013744
0.4: 0.7980225735320542

0.5: 0.7550364737786981
0.6: 0.5649661637096092
0.7: 0.4640418865852142
0.8: 0.24597651576521107
0.9: 0.10199420994244983

These 10 values are the average loss values of each network based on the test set, with dropout values of the networks on the left side of the colon. It is noticeable that the loss values does not strongly correlate with the accuracy of the network, as the most accurate dropout value, 0.2, did not have the lowest loss value. Instead, the 0.4 dropout ratio network average loss was the lowest with 0.57. Loss increased drastically once the dropout ratio went over 0.5, as with accuracy, the networks must have been too sparse to learn properly.

0.0: 0.6867696763976355
0.1: 0.6533132363783233
0.2: 0.6960869534847111
0.3: 0.5926545384143919
0.4: 0.5742934469301332
0.5: 0.6959309047454733
0.6: 1.1392159158030102
0.7: 1.3953222555569802
0.8: 1.8261419383164463
0.9: 2.302556776788494

Overall, for this data, it seems that the ideal dropout ratio is around 0.2 to 0.3, as that is where the both the accuracies are around the highest and the losses are around the lowest out of all the dropout ratio networks. With the small improvements in loss and accuracy, I can conclude (with some doubt because of the minute gains) that dropout does in fact help reduce overfitting, as dropping 20% to 30% of the network nodes did improve accuracy by almost 2%.

Used tensorflow and keras to create these networks, special thanks to:

https://www.tensorflow.org/guide/keras