Abhijay Thammana

Project 2 Report

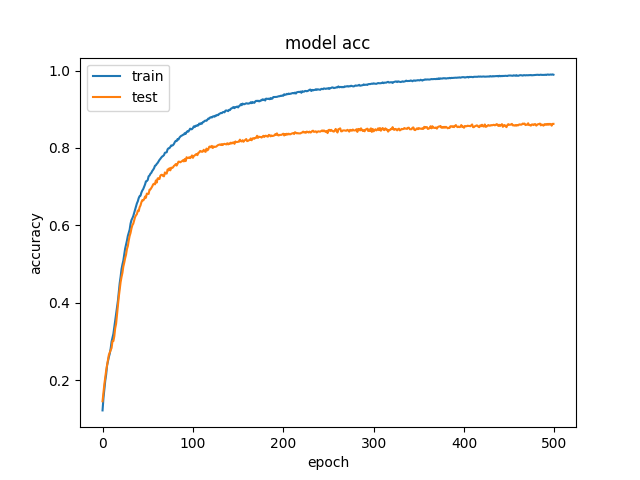
9/8/19

CS 4341

Before running the program the following dependencies must be installed:

keras, numpy, matplotlib, and sklearn

A picture containing screenshot

Description automatically generated**Task 2**

To run Task 2 uncomment the task 2 function at the bottom of the program. The data pulled from running task 2 was done with these parameters: learning rate of 0.001, 10 hidden layers of 50 nodes each, batch size of 512, and 500 epochs. The plots from the neural network are below:

*Task 2 before dropout rate: loss = '0.6168550401467543', acc = '0.8523076860721295'*

*Confusion matrix:*

*[119 0 2 1 0 8 3 1 1 3]*

*[ 0 166 2 1 0 0 0 0 2 1]*

*[ 0 3 104 6 3 0 1 2 1 1]*

*[ 1 3 5 91 0 6 1 6 6 0]*

*[ 1 0 3 0 111 2 3 1 1 7]*

*[ 6 0 1 6 5 94 3 1 8 2]*

*[ 0 0 3 1 3 2 96 0 1 0]*

*[ 0 3 3 0 2 2 0 110 3 3]*

*[ 1 1 4 4 3 9 2 1 112 1]*

*[ 1 0 1 0 9 1 0 6 2 105]*

These were used as a baseline to measure all the other data to. We can also see from these graphs that the network was clearly overfitted. The test accuracy is significantly lower in the graph. The training data’s accuracy also reaches a value of 1, which we do not want. The data from the neural network with a dropout rate of 20 % on the input layer is below:

A picture containing screenshot

Description automatically generatedA screenshot of a cell phone

Description automatically generated

*Task 2 after dropout rate: loss = '0.28233517000308406', acc = '0.9238461571473342'*

*Confusion matrix:*

*[132 0 1 0 0 3 1 0 1 0]*

*[ 0 168 0 0 0 0 0 0 3 1]*

*[ 0 2 111 2 5 0 1 0 0 0]*

*[ 0 1 2 104 0 4 1 2 4 1]*

*[ 0 1 0 0 119 1 1 0 0 7]*

*[ 1 0 0 2 0 116 3 0 2 2]*

*[ 0 0 0 0 1 1 103 0 1 0]*

*[ 1 3 2 0 2 0 0 113 1 4]*

*[ 1 1 2 7 0 6 1 0 120 0]*

*[ 1 0 0 1 2 0 0 5 1 115]*

The graphs with the dropout rate on the input layer had a lot more noise. This may be because some of the epochs were run with with inaccurate/incomplete data so the prediction might have made it worse that before, where we just overfit to whatever data we fed. For this graph the accuracy does not reach 1 for the training data. The test and training data also are about the same place in accuracy near the end so it was probably not overfitted.

**Task 3**

To run task 3 comment out the task 2 block and uncomment the task 3 block. Task 3 works by making 3 splits on the training data for where the validation data will come from. The format of the splits will be listed below. The data from the first cross is below:

The format for the data was ([train] | [train] | [validation])

A picture containing screenshot

Description automatically generatedA screenshot of a cell phone

Description automatically generated

*loss = '0.23014887667619266', acc = '0.950000001833989'*

*Confusion matrix:*

*[137 0 0 0 0 0 0 1 0 0]*

*[ 0 166 1 0 0 1 0 0 3 1]*

*[ 0 0 116 0 2 2 0 0 1 0]*

*[ 0 0 6 90 0 10 0 2 11 0]*

*[ 0 0 1 0 115 1 1 0 1 10]*

*[ 0 0 1 0 1 116 3 0 5 0]*

*[ 1 0 1 0 2 0 102 0 0 0]*

*[ 0 0 2 0 1 0 0 111 1 11]*

*[ 0 0 0 0 0 2 0 0 136 0]*

*[ 1 0 0 0 0 1 0 1 0 122]*

A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generatedThe format for the next block was ([train] | [validation] | [train])

*loss = '1.1239180751947255', acc = '0.7861538432194637'*

*Confusion matrix:*

*[133 1 0 1 0 0 2 1 0 0]*

*[ 0 170 0 0 0 0 0 1 0 1]*

*[ 3 4 93 6 5 0 4 0 4 2]*

*[ 3 2 3 99 1 1 2 5 1 2]*

*[ 0 2 1 0 101 0 3 0 0 22]*

*[ 7 4 1 23 11 59 13 0 5 3]*

*[ 3 5 1 0 2 1 94 0 0 0]*

*[ 3 10 2 0 1 0 1 89 0 20]*

*[ 3 9 2 12 8 10 2 1 84 7]*

*[ 1 0 1 1 18 0 0 4 0 100]*

A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generatedThe format for the final block was ([validation] | [train] | [train]):

*loss = '0.1767846933236489', acc ='0.9430769260113055'*

*Confusion matrix:*

*[135 0 1 0 0 0 0 1 1 0]*

*[ 0 167 1 1 0 0 0 0 2 1]*

*[ 0 0 113 3 3 1 0 1 0 0]*

*[ 0 0 3 109 0 2 0 2 3 0]*

*[ 0 1 0 0 121 0 0 1 0 6]*

*[ 0 0 3 5 0 115 1 0 1 1]*

*[ 0 0 3 0 2 1 100 0 0 0]*

*[ 0 1 0 1 1 0 0 118 1 4]*

*[ 0 0 2 3 0 4 0 0 129 0]*

*[ 1 0 0 3 0 0 0 2 0 119]*

The graphs above are very different from each other. There were 3 folds, so each block was 33% of the data. The validation data was in the last block, middle block, and first block. When the validation data was in the second block there was a weird spike in the wrong direction, about halfway through. That seemed to be about when the training data switched from the 1st block to the 3rd block, so it might’ve been human error. The final graph also jumped from around 0 to a high accuracy value so there was a very weird jump there. All in all the accuracy was lowest for the second block and it was about the same for the 1st and the 3rd.

**Task 4**

The tests and the outputs are listed below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 10 layers, 500 epochs, 512 batch, 20% input dropout | 10 layers, 100 epochs, 512 batch size, 20 % dropout to input: | 1 layers, 500 epochs, 512 batch size, 20 % dropout to input: | 10 layers, 500 epochs, 512 batch size, 20 % dropout to input and 20% to every other layer: | 10 layers, 100 epochs, 32 batch size, 20 % dropout to input: |
| Fold 1 acc | 0.157180194 | 0.582618352 | 2.044972463 | 2.299035164 | 0.08649045 |
| Fold 1 loss | 0.953076925 | 0.816923072 | 0.872307686 | 0.139999998 | 0.973846154 |
| Fold 2 acc | 14.55447647 | 1.230884033 | 1.58240318 | 2.280416181 | 0.827918283 |
| Fold 2 loss | 0.096923077 | 0.752307691 | 0.478461541 | 0.20076923 | 0.861538462 |
| Fold 3 acc | 0.210593292 | 0.393665456 | 0.294025453 | 2.057025896 | 0.143494277 |
| Fold 3 loss | 0.930769234 | 0.873076928 | 0.930769234 | 0.293076921 | 0.960769231 |
| TTest Value | Baseline | 0.212709282 | 0.242274698 | 0.265234892 | 0.199212846 |

From the TTest value we can see that all of them will reject the null hypothesis since they are greater than .05, But this value is not very reliable since fold 2 is so unreliable. It keeps jumping from finding results to having an accuracy of 10% from just guessing. The human predictions are as follows (from only looking at folds 1 and 3): Changing the epoch size decreases the accuracy; Changing the number of layers to 1 decreases the accuracy a lot; Dropping 20% after every hidden layer makes the accuracy a lot worse; Making the batch size very small slows it down considerable and may have had a very significant positive impact on the network. All the other test took about the same amount of time.