Report :: TIPR Assignment - II

NIDHI KUMARI (15127)

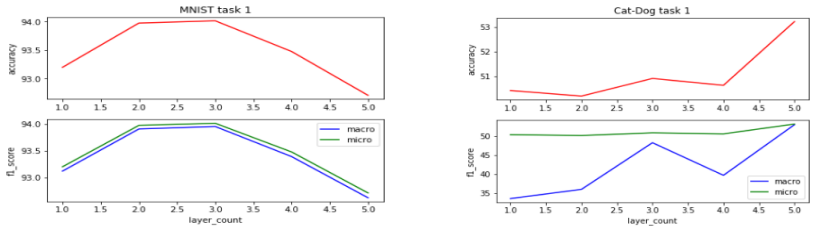
04-03-2019

1 INDRODUCTION

Python version -3
In this assignment we have implemented Neural Network from scratch and compared it with the inbuilt keras implementation . Furthermore compare the results on some dataset with KNN and Naive bayes. Dataset to work upon are MNIST and Cat-Dog

2 Task1

We have to experiment with different no. of layers. i have fixed the no.of of neurons in each layer. For MNIST it is 30 neurons per layer and for Cat-Dog it is 50 neurons each layer. for each task i have divided the data into train(70%), test(30%)



(a) MNIST (b) Cat - Dog

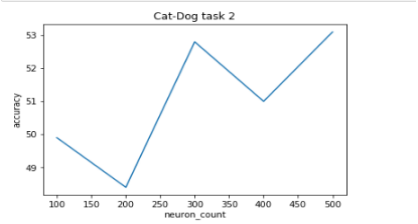
92% Accuracy on test set while using 1 layers with 30 neurons for MNIST data Accuracy keeps on decreasing with increase in no. of neurons . this is not the case with cat-dog it first decreases then increases then keeps on decreasing.

Figure 1

3 Task2

In task 2 we have to fix the no. of layers and experiment with no.of neurons. so i choose the no.of layer = 1 and experimented with no.of neurons

(a) MNIST



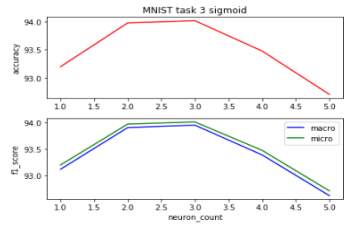
(b) Cat - Dog

94% Accuracy on test set while using 3 layers with 30 neurons each for MNIST data. While it is not giving good result on cat and dog dataset, it is 54% while using 5 layers.

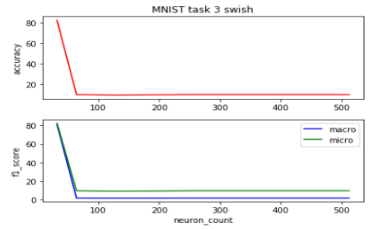
Figure 2

4 Task3

in this we have to play with different activation function .



(a) MNIST - Sigmoid

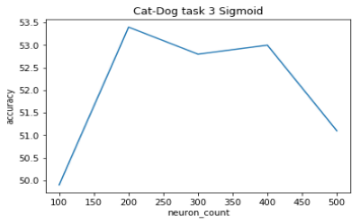


(b) MNIST - Swish

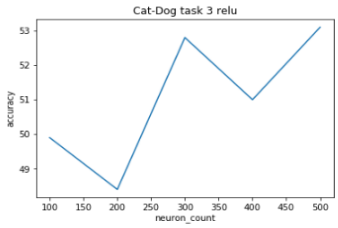
Sigmoid is giving good accuracy as compared to Swish and Relu in case of MNIST data

Figure 3

Figure 4: MNIST - Relu



(a) Cat - Dog Sigmoid



(b) Cat - Dog Relu

Both Sigmoid and Relu are giving accuracy of 53% . The only difference is Sigmoid is giving it at some less no. of neuron.

Figure 5

5 Task4

using different weight initialization technique not getting any observable difference.

6 Task5

implementing neural network using keras.

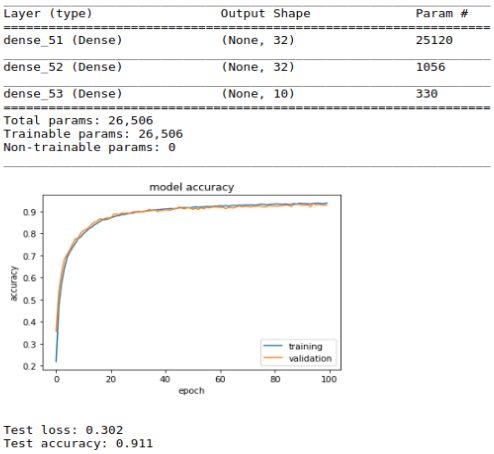
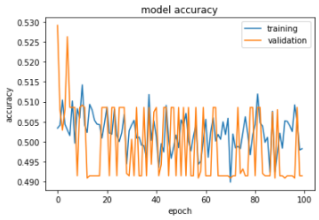


Figure 6: MNIST Keras implementation

Layer (type)	Output Shape	Param #
dense_16 (Dense)	(None, 1000)	120001000
dense_17 (Dense)	(None, 64)	64064
dense_18 (Dense)	(None, 2)	130

Total params: 120,065,194
Trainable params: 120,065,194
Non-trainable params: 0



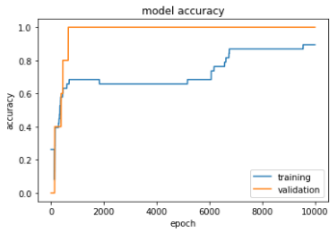
Test loss: 0.699
Test accuracy: 0.498

7 Part-3

In this part we have to work with the dataset of the first assignment and compre its result with KNN and Naive Bayes

Layer (type)	Output Shape	Param #
dense_35 (Dense)	(None, 5)	165
dense_36 (Dense)	(None, 4)	24

Total params: 189
Trainable params: 189
Non-trainable params: 0

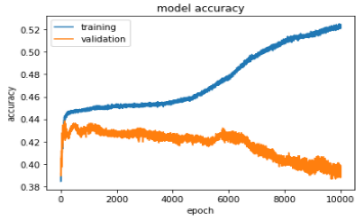


Test loss: 0.393
Test accuracy: 0.947

The accuracy which we are getting on dolphins with neural network is 95%. where as with KNN we were getting 91% and Naive Bayes 92%.

Figure 9: Dolphins

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_63 (Dense)             (None, 10)                1290
_________________________________________________________________
dense_64 (Dense)             (None, 10)                110
_________________________________________________________________
dense_65 (Dense)             (None, 3)                 33
=================================================================
Total params: 1,433
Trainable params: 1,433
Non-trainable params: 0
_________________________________________________________________
```

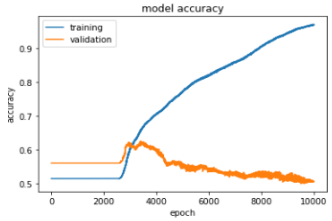



```
Test loss: 1.14
Test accuracy: 0.401
```

The accuracy which we are getting on pubmed with neural network is 40.1 %. where as with KNN we were getting 37% and Naive Bayes 43%.

Figure 10: Pubmed

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_66 (Dense)             (None, 10)                29790
_________________________________________________________________
dense_67 (Dense)             (None, 10)                110
_________________________________________________________________
dense_68 (Dense)             (None, 3)                 33
=================================================================
Total params: 29,933
Trainable params: 29,933
Non-trainable params: 0
_________________________________________________________________
```




```
Test loss: 2.61
Test accuracy: 0.512
```

The accuracy which we are getting on Twitter with neural network is 51.2 %. where as with KNN we were getting 39% and Naive Bayes 46%.

Figure 11: Twitter

8 Github link

`https://github.com/Nidhi-kumari/tipr-second-assignment`