BITS F464 Machine Learning
Fischer's Linear Discriminant

1. Introduction:

In this report, we try to analyse the performance (accuracy and fscore) of a Fischer's Linear Discriminant. The implementation of this model was done using Jupyter Notebook, NumPy, Pandas and Matplotlib.

2. About the Model:

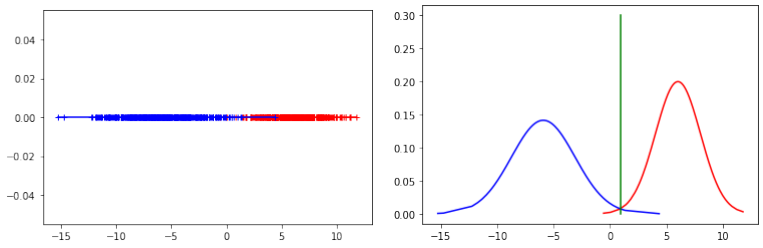
The first model we built was a 2-layer neural network (i.e. with one hidden layer). We experimented with activations like linear, ReLU and tanh for the hidden layer. The input layer had 10 neuron units with a ReLU activation (for best results. A slight drop in accuracy was observed when linear activation was used for the input layer). The final layer had an activation of sigmoid for binary classification. Output value greater than or equal to 0.5 was taken to represent positive classes and less than 0.5 to represent negative classes. `np.random.random()` and `np.random.randn()` methods were used to randomly initialize weights and biases associated network to initialize the weights according to a Uniform or Guassian distribution respectively. Best results were obtained using uniformly distributed weights/biases with a seed=5 and 4 hidden neurons with ReLU and learning_rate=0.001 with 20000 training epochs. The loss was observed to have more fluctuations using this configuration after 40000 epochs and thus lowering the accuracy (images attached below). The avg. loss using this configuration was observed to be 0.7 (per data point). The highest accuracy achieved by this model using this configuration was 87.67% and the lowest accuracy was 80% with an average accuracy of 84%.

6. Conclusion:

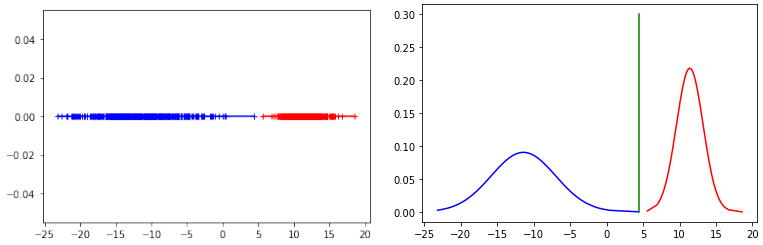
- In general, a three layer network performs better than a two layer network. Adding more layers/more neurons to the hidden layers provides more trainable parameters which help the network to fit the model in a better way. But the drawback of adding multiple layers/mulitple neurons is that, the model requires more iterations to train. In this case, there is a higher chance that the model might overfit the data, and hence giving a low testing accuracy.
- A lower learning rate requires more iterations to converge, but conversely, using a higher learning rate might lead to fluctuation in the loss/exploding gradient.
- Different initialization of weights leads to convergence of the model at different local minima (initializations mentioned above).
- Different activation functions for the hidden layers provides drastically varying results. We have used ReLU (Rectified Linear) which is by and the far, the de-facto standard activation used for hidden layers. Other activations like tanh or sigmoid gave very low accuracy as compared to that of ReLU. ReLU suffers from the *dying neuron problem*, in which the output of a neuron might be 0 for most of the training (i.e. that neuron is not contributing to the network), and hence we tried using leaky-ReLU, but the accuracy was found to more or less the same.

[results on next page]

8. Results:



2-Layer NN using uniform distribution



3-Layer NN using uniform distirbution

Fig5.png

Fig6.png

2-Layer NN using normal distirbution

Fig7.png

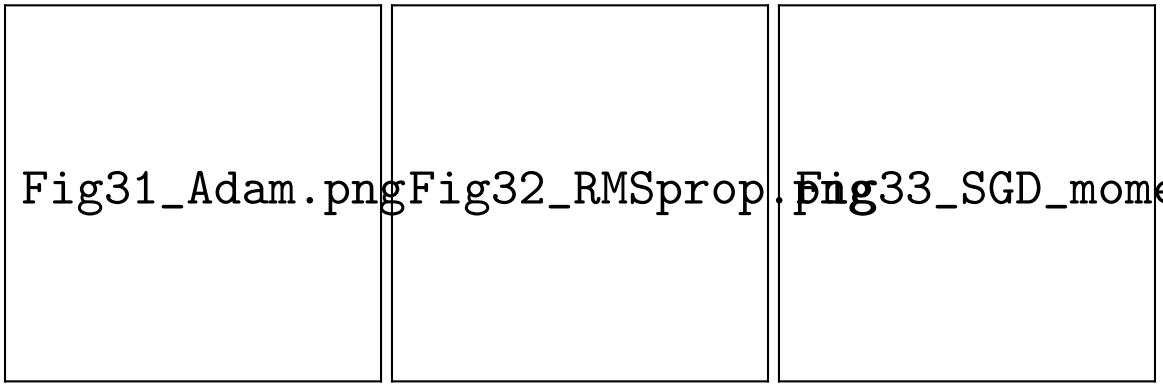
Fig8.png

3-Layer NN using normal distirbution

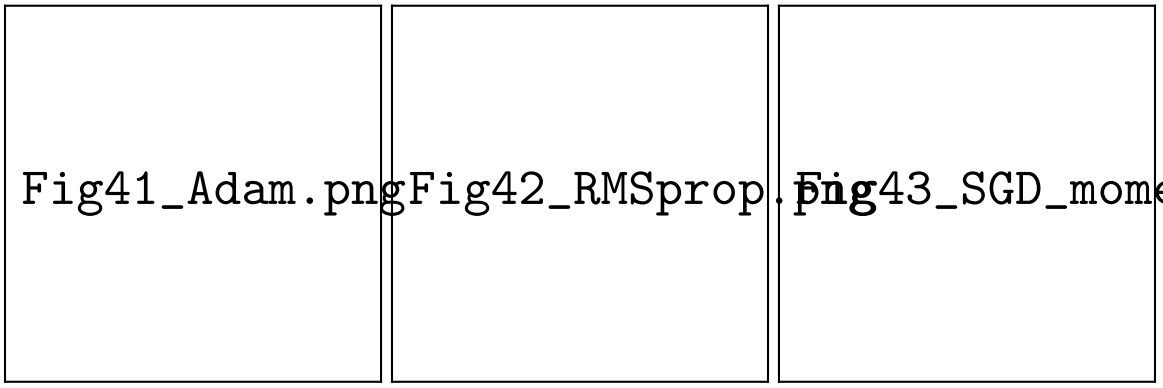
”

’’’’

9. Additional Results:



Loss vs Epochs for Adam, RMSprop and Momentum respectively



3-Layer NN with Adam, RMSprop and Momentum respectively for 3750 epochs