Bank Note

Github Link: https://github.com/praneeth300/HappyMonk-/blob/main/Bank_Note.ipynb

Problem Statement:

Given a specific activation function

g(x) = k0 + k1x (1)

and categorical cross-entropy loss, design a Neural Network on Banknote, MNIST or IRIS data where the activation function parameters k0, k1 are learned from the data you choose from one of the above-mentioned data sets. *Your solution must include the learnable parameter values i.e. final k0, k1 values at the end of training, a plot depicting changes in k0, k1 at each epoch, training vs test loss, train vs. test accuracy and a Loss function plot.*

Dataset:

	variance	skewness	curtosis	entropy	class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Shape of the dataset:

Number of columns: 5

Number of observations: 1372

Skewness:

As we can observe that some of the variables in our dataset have some skewness i.e.,curtosis and entrophy show right and left skewnees whereas in variance and skewness display's like it is a gaussina distribution. We need to handle it by using the some of the feature transformation techniques such as "Power transform","Box cox transformation" or "Log normal transformation". Genrally logarthemic transformation perform well on the skewness data, it converts the data in to gaussian distribution or normal distribuiton.

Selecting Parameters:

Given that the dataset is small, a small batch size is probably a good idea, e.g. 16 or 32 rows. Using the Adam version of stochastic gradient descent is a good idea when getting started as it will automatically adapt the learning rate and works well on most datasets.

Model architeture:

We will develop a Multilayer Perceptron (MLP) model for the dataset using TensorFlow.

We cannot know what model architecture of learning hyperparameters would be good or best for this dataset, so we must experiment and discover what works well**.**

After Experimented with most of the activation functions **LeakyRelu (alpha=0.01)** performs well on the training data and validation data , by considering the all parameters of the model.

N_features = 4

feat = x.shape[1]

Initialize the model

l_model = Sequential()

First Hidden layer with 10 Node's and Input layer with the shape of X

l_model.add(Dense(10,activation=LeakyReLU(alpha=0.01),input_shape=(feat,)))

Output Layer with 1 node,because it is an binary classification model

l_model.add(Dense(1,activation='sigmoid'))

Compile the Model with loss is 'Binary Croossentropy"

l_model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

Fit the Model to the training data.

hist_lrel = l_model.fit(x_train,y_train,validation_data=(x_val,y_val),epochs=50,batch_size=32,verbose=1)

Intial Epoch:

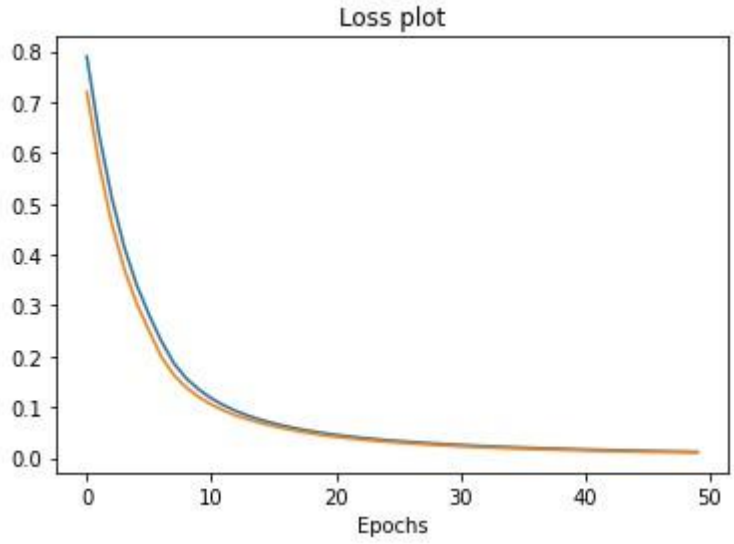
Epoch 1/50 35/35 [==============================] - 1s 6ms/step - loss: 0.7897 - accuracy: 0.6427 - val_loss: 0.7202 - val_accuracy: 0.6691

Final Epoch:

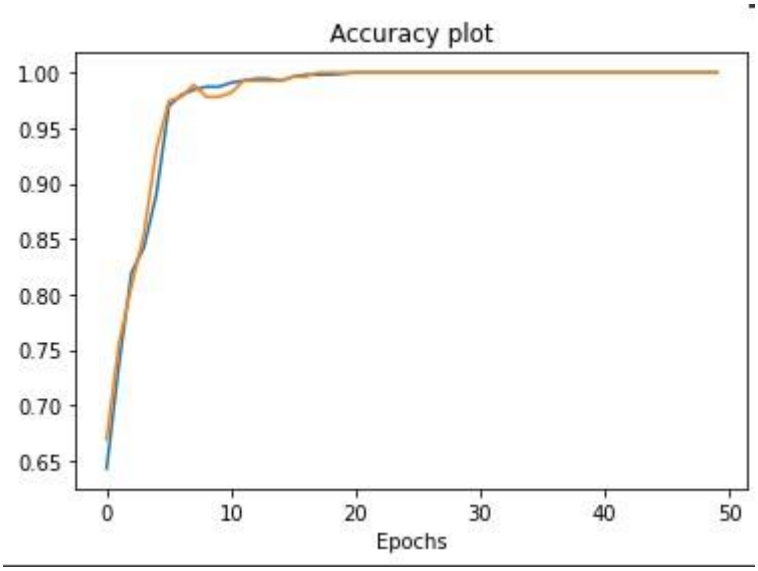
Epoch 50/50 35/35 [==============================] - 0s 3ms/step - loss: 0.0120 - accuracy: 1.0000 - val_loss: 0.0104 - val_accuracy: 1.0000

Plots:

1. **Loss plot:**



2. **Accuracy plot:**



Test data accuracy:

Accuracy: 1.00

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	157
1	1.00	1.00	1.00	118
accuracy			1.00	275
macro avg	1.00	1.00	1.00	275
weighted avg	1.00	1.00	1.00	275

Weights:

```
<tf.Variable 'dense_56/kernel:0' shape=(4, 10) dtype=float32, numpy=
array([[-1.0191457 ,  0.43691105, -0.24920474,  0.91669387,  1.1209365 ,
         0.17461905,  0.30388984,  0.38420722, -1.1818974 , -0.73866546],
       [ 0.12706555,  0.8034793 , -0.09776583,  0.34244478,  0.5177767 ,
        -0.1604144 ,  0.9389532 ,  0.05366395, -0.3934279 , -0.01984872],
       [ 0.64960986, -0.17276475, -0.56822956, -0.01981766,  0.8746438 ,
         0.28390008,  0.78429264, -0.73570484, -0.09717001, -0.9764796 ],
       [ 0.21970525,  0.14473073, -0.7532152 , -0.37142417, -0.3458716 ,
         0.26863584,  0.07149642,  0.34521458,  1.3007131 ,  0.13457708]],
      dtype=float32)>,
<tf.Variable 'dense_56/bias:0' shape=(10,) dtype=float32, numpy=
array([ 0.30863136, -0.35084096,  0.6415316 ,  0.05500666,  0.44968727,
       -0.01826408, -0.04380936,  0.17380986,  0.8931041 ,  0.7091936 ],
      dtype=float32)>,
<tf.Variable 'dense_57/kernel:0' shape=(10, 1) dtype=float32, numpy=
array([[ 0.2933338 ],
       [-0.37113386],
       [ 0.5880925 ],
       [-0.44298902],
       [-0.6951934 ],
       [-0.3590017 ],
       [-0.61465156],
       [ 0.542976  ],
       [ 1.0765252 ],
       [ 1.0486454 ]], dtype=float32)>,
<tf.Variable 'dense_57/bias:0' shape=(1,) dtype=float32, numpy=array([0.24318656], dtype=float32)>]
```

Preseneted By:

P.Praneeth Kumar

Contact: +918309116050

Email: praneethsanthosh555@gmail.com

Thank You