

Introduction to ML HW-05

Problem Statement:

The problem statement of this classification problem is to categorise individuals into one of three categories based on the ratio of citations in 2022 to citations in 2021, approximated to two decimals.

1. Low (ratio < 1.05)
2. Medium (ratio: 1.06 - 1.15)
3. High (ratio > 1.15)

To solve this problem, a 1-hidden layer neural network with 6-6-3 architecture is used. This aims to learn from the dataset and make predictions on the classification of individuals based on their citation ratios.

The dataset is split into two parts: 80% for training and 20% for testing.

The input features to the neural network comprises citation numbers from 2017 to 2022, for effective model training.

Approach:

1. Imported the important and useful libraries.
2. Loaded the data set into the python notebook using pandas.
3. Explored and analysed the data.
4. Calculated the citation ration of citations in 2022 to citations in 2021 of each individual.
5. Splitting the data into training (80 %) and testing (20 %) set using sklearn. This division allows for model training and evaluation on separate databases
6. Normalising both the train and test data.
7. Neural Network Architecture:
 - I defined neural network architecture as a sequential model using TensorFlow.
 - It consists of two layers:
 1. A hidden layer with 6 neurons and hyperbolic tangent(tanh) activation function.
 2. An output layer with 3 neurons and the sigmoid activation function.
 - I used this model and architecture to accommodate the classification task, with the output layer having three neurons to represent the three categories: Low, Medium and High.
8. Hyperparameters and Optimization:
 - I tried a different set of learning rates such as 0.1, 0.01, 0.001, 0.2, and 0.3 to control the step size during weight updates.
 - I used the Stochastic Gradient Algorithm (SGD) optimization algorithm to optimise the model.
 - I used the Categorical Cross Entropy loss function to measure the difference between predicted and actual labels.

9. Evaluation of Results:

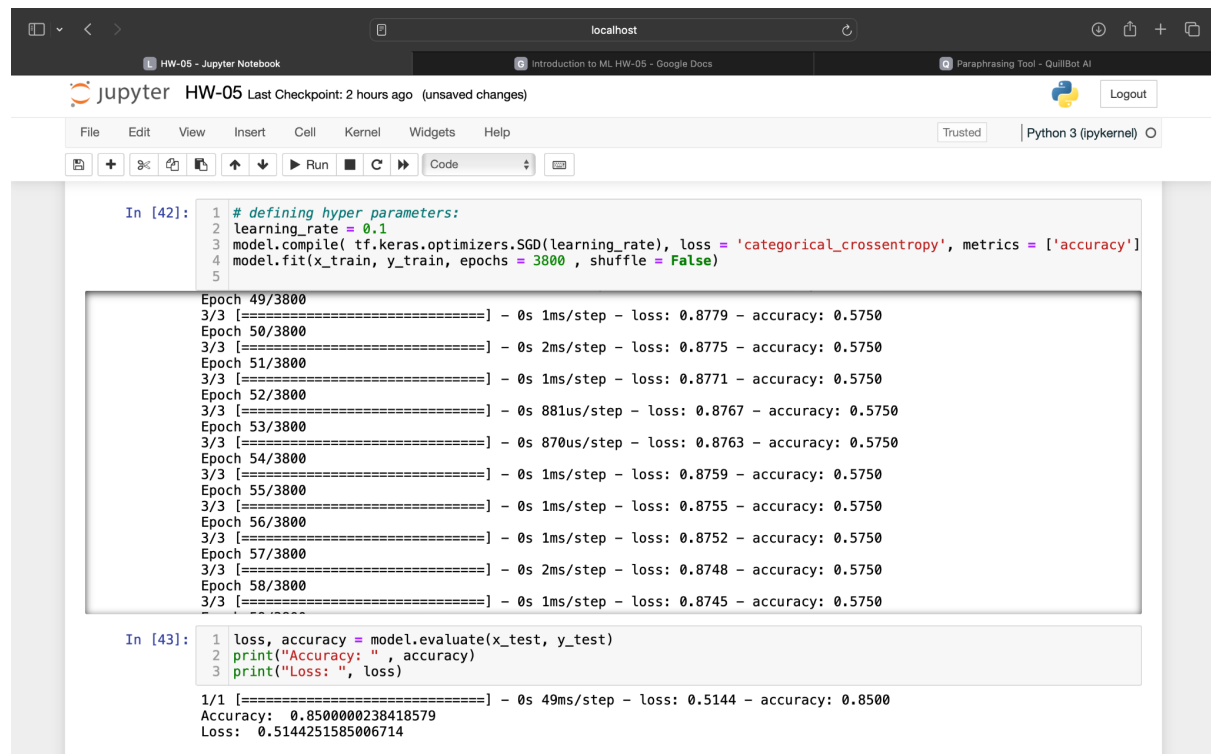
- After training, I evaluated the model's performance using the remaining 20% of the test data.
 1. Accuracy: To find how correct model predictions are.
 2. Loss: To measure the error between model predictions and actual labels.
- The evaluation metrics provided insights into how well the neural network has learned to classify individuals based on the citation ratios.

Result:

Epoch	Learning rate	Loss	Accuracy
1000	0.1	0.7631	0.75
3000	0.1	0.5650	0.800
3500	0.1	0.5319	0.8500
3800	0.1	0.5144	0.8500
100	0.01	0.8714	0.6500
1000	0.01	0.8614	0.6000
2000	0.01	0.8659	0.6000
1000	0.2	0.6710	0.7500
3000	0.2	0.5256	0.7500
1000	0.3	0.6477	0.7000
2000	0.3	0.5574	0.7500

Optimal Output:

Epoch: 3800	Learning Rate: 0.1	Loss: 0.5144	Accuracy: 0.8500
--------------------	---------------------------	---------------------	-------------------------



```
In [42]: 1 # defining hyper parameters:
2 learning_rate = 0.1
3 model.compile( tf.keras.optimizers.SGD(learning_rate), loss = 'categorical_crossentropy', metrics = ['accuracy'])
4 model.fit(x_train, y_train, epochs = 3800 , shuffle = False)
5

Epoch 49/3800
3/3 [=====] - 0s 1ms/step - loss: 0.8779 - accuracy: 0.5750
Epoch 50/3800
3/3 [=====] - 0s 2ms/step - loss: 0.8775 - accuracy: 0.5750
Epoch 51/3800
3/3 [=====] - 0s 1ms/step - loss: 0.8771 - accuracy: 0.5750
Epoch 52/3800
3/3 [=====] - 0s 881us/step - loss: 0.8767 - accuracy: 0.5750
Epoch 53/3800
3/3 [=====] - 0s 870us/step - loss: 0.8763 - accuracy: 0.5750
Epoch 54/3800
3/3 [=====] - 0s 1ms/step - loss: 0.8759 - accuracy: 0.5750
Epoch 55/3800
3/3 [=====] - 0s 1ms/step - loss: 0.8755 - accuracy: 0.5750
Epoch 56/3800
3/3 [=====] - 0s 1ms/step - loss: 0.8752 - accuracy: 0.5750
Epoch 57/3800
3/3 [=====] - 0s 2ms/step - loss: 0.8748 - accuracy: 0.5750
Epoch 58/3800
3/3 [=====] - 0s 1ms/step - loss: 0.8745 - accuracy: 0.5750

In [43]: 1 loss, accuracy = model.evaluate(x_test, y_test)
2 print("Accuracy: ", accuracy)
3 print("Loss: ", loss)

1/1 [=====] - 0s 49ms/step - loss: 0.5144 - accuracy: 0.8500
Accuracy:  0.8500000238418579
Loss:  0.5144251585006714
```

Now, I have evaluated my neural network model on the basis of loss and accuracy. This means that:

1. How well my model learns from the training data and tries to make predictions as close as possible to the actual training data.
2. How well the model performs on the new and unseen data. The accuracy score tells us how many of the new data points the model classifies correctly or if a model can perform accurate predictions on the data it has never seen before.

Conclusion:

In conclusion, I believe that with **epochs = 3800**, **learning rate = 0.1**; the model evaluates **loss of 0.5144** and **accuracy of 85% (i.e. 0.8500)**. This means that this configuration not only fits the training data well (low loss) but also does a great job at making correct predictions on the new and unseen data (high accuracy). It's basically a model that not only understands what it has seen before but can also use that knowledge to make accurate predictions on the things it hasn't seen before. Hence, I can say that this is a strong and reliable model.

Y - test:

```
In: 1 y_test
```

	Low	Medium	High
83	1	0	0
53	1	0	0
70	1	0	0
45	1	0	0
44	1	0	0
39	1	0	0
22	1	0	0
80	1	0	0
10	0	0	1
0	1	0	0
18	0	0	1
30	0	0	1
73	0	1	0
33	1	0	0
90	0	0	1
4	0	0	1
76	0	1	0
77	1	0	0
12	1	0	0
31	1	0	0

Y -predicted:

```
5 y_pred
array([[1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0],
       [0, 0, 1],
       [1, 0, 0],
       [1, 0, 0],
       [0, 0, 1],
       [0, 0, 1],
       [1, 0, 0],
       [0, 0, 1],
       [0, 0, 1],
       [1, 0, 0],
       [1, 0, 0],
       [0, 0, 1],
       [0, 0, 1],
       [1, 0, 0],
       [1, 0, 0],
       [1, 0, 0]])
```