CS353 ML Lab 9

Name: K V Sumanth Reddy

Roll No: 181CO225

Batch: Section 2

Date: 06/04/2021

Q: Implementation of Artificial Neural Network for **NOR Logic Gate** with 2-bit Binary Input.

Import Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from sklearn.metrics import accuracy_score,precision_score,recall_sc
from keras import models, layers, losses, optimizers, metrics
```

Making a Dataset

Truth table for NOR gate

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

```
x = np.random.randint(0,2,(10000,2))
y = x[:,0] + x[:,1]

for i in range(len(y)):
  if y[i]==0:
    y[i]=1
  else:
```

```
    y[i]=0
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size = 0.20
```

Neural Network

```
model = models.Sequential()
model.add(layers.Dense(1, input_shape=(2,), activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam',
              metrics=[metrics.categorical_accuracy])
```

```
result = model.fit(x_train, y_train, validation_split=(0.2),
                   batch_size=100, epochs=40, verbose=1)
```

```
Epoch 12/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1229 - categ
Epoch 13/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1208 - categ
Epoch 14/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1168 - categ
Epoch 15/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1151 - categ
Epoch 16/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1133 - categ
Epoch 17/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1110 - categ
Epoch 18/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1069 - categ
Epoch 19/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1040 - categ
Epoch 20/40
64/64 [==============================] - 0s 2ms/step - loss: 0.1039 - categ
Epoch 21/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0987 - categ
Epoch 22/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0974 - categ
Epoch 23/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0944 - categ
Epoch 24/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0915 - categ
Epoch 25/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0904 - categ
Epoch 26/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0871 - categ
Epoch 27/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0871 - categ
Epoch 28/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0818 - categ
Epoch 29/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0804 - categ
Epoch 30/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0786 - categ
Epoch 31/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0768 - categ
```

```
64/64 [==============================] - 0s 2ms/step - loss: 0.0768 - categ
Epoch 32/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0734 - categ
Epoch 33/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0716 - categ
Epoch 34/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0689 - categ
Epoch 35/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0676 - categ
Epoch 36/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0664 - categ
Epoch 37/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0645 - categ
Epoch 38/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0613 - categ
Epoch 39/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0611 - categ
Epoch 40/40
64/64 [==============================] - 0s 2ms/step - loss: 0.0584 - categ
```

```
output = model.evaluate(x_test,y_test)
```

```
63/63 [==============================] - 0s 1ms/step - loss: 0.0581 - categor
```

Predictions

```
y_pred=model.predict(x_test).round()
```

```
model.predict([[0,0]]).round()
```

```
array([[1.]], dtype=float32)
```

```
model.predict([[0,1]]).round()
```

```
array([[0.]], dtype=float32)
```

```
model.predict([[1,0]]).round()
```

```
array([[0.]], dtype=float32)
```

```
model.predict([[1,1]]).round()
```

```
array([[0.]], dtype=float32)
```

Results

Accuracy:

$$\frac{TP + TN}{P + N}$$

Precision:

$$\frac{TP}{TP + FP}$$

Root Mean Squared Error:

$$\frac{TP}{TP + FN}$$

F1 Score:

$$\frac{2TP}{2TP + FP + FN}$$

```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: %f' % accuracy)

precision = precision_score(y_test, y_pred)
print('Precision: %f' % precision)

recall = recall_score(y_test, y_pred)
print('Recall: %f' % recall)

f1 = f1_score(y_test, y_pred)
print('F1 score: %f' % f1)
```

```
Accuracy: 1.000000
Precision: 1.000000
Recall: 1.000000
F1 score: 1.000000
```

✓ 0s completed at 12:49