CS353 ML Lab 8

Name: K V Sumanth Reddy

Roll No: 181CO225

Batch: Section 2

Date: 30/03/2021

Q: Implementation of **AND Gate using Artificial Neural Network**. To understand the working of neural networks using AND Gates implemented through neural network.

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

```
x = np.random.randint(0,2,(8000,2))
y = x[:,0]*x[:,1]
```

```
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=30, verbose=1)
```

```
Epoch 2/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2779 - categ
Epoch 3/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2720 - categ
Epoch 4/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2632 - categ
Epoch 5/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2579 - categ
Epoch 6/30
52/52 [==============================] - 0s 3ms/step - loss: 0.2561 - categ
Epoch 7/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2515 - categ
Epoch 8/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2384 - categ
Epoch 9/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2325 - categ
Epoch 10/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2271 - categ
Epoch 11/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2197 - categ
Epoch 12/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2143 - categ
Epoch 13/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2128 - categ
Epoch 14/30
52/52 [==============================] - 0s 2ms/step - loss: 0.2072 - categ
Epoch 15/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1999 - categ
Epoch 16/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1951 - categ
Epoch 17/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1917 - categ
Epoch 18/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1837 - categ
Epoch 19/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1807 - categ
Epoch 20/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1776 - categ
Epoch 21/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1716 - categ
Epoch 22/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1682 - categ
Epoch 23/30
52/52 [==============================] - 0s 3ms/step - loss: 0.1636 - categ
Epoch 24/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1616 - categ
Epoch 25/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1613 - categ
Epoch 26/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1542 - categ
Epoch 27/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1511 - categ
Epoch 28/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1493 - categ

Epoch 29/30
52/52 [==============================] - 0s 2ms/step - loss: 0.1445 - categ
Epoch 30/30
```

```
Epoch 50/50
52/52 [==============================] - 0s 2ms/step - loss: 0.1433 - categ
```

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

```
50/50 [==============================] - 0s 1ms/step - loss: 0.1377 - categor
```

Predictions

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

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

```
array([[0.]], 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([[1.]], 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
```