類神經網路作業 1 - 設計感知機類神經網路

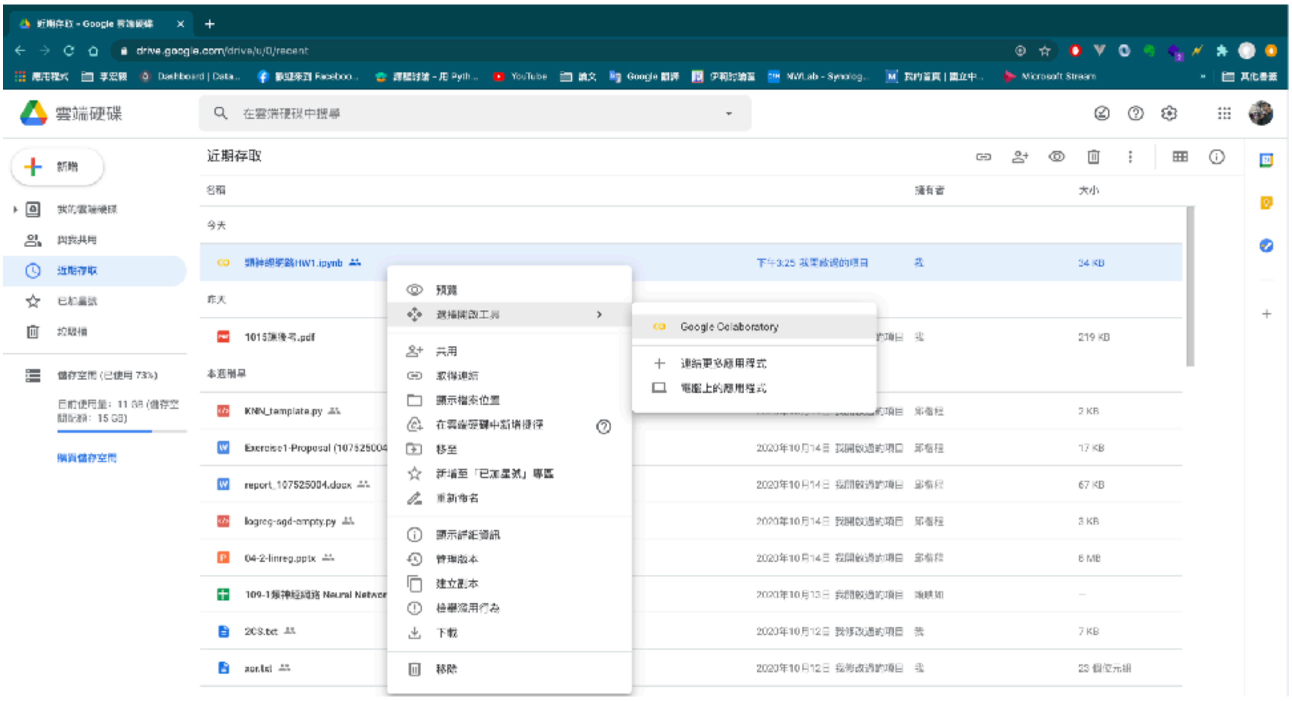
109525009張祐綸

A, 程式執行說明

我用的程式語言是Python, 環境是cola 開發自google,

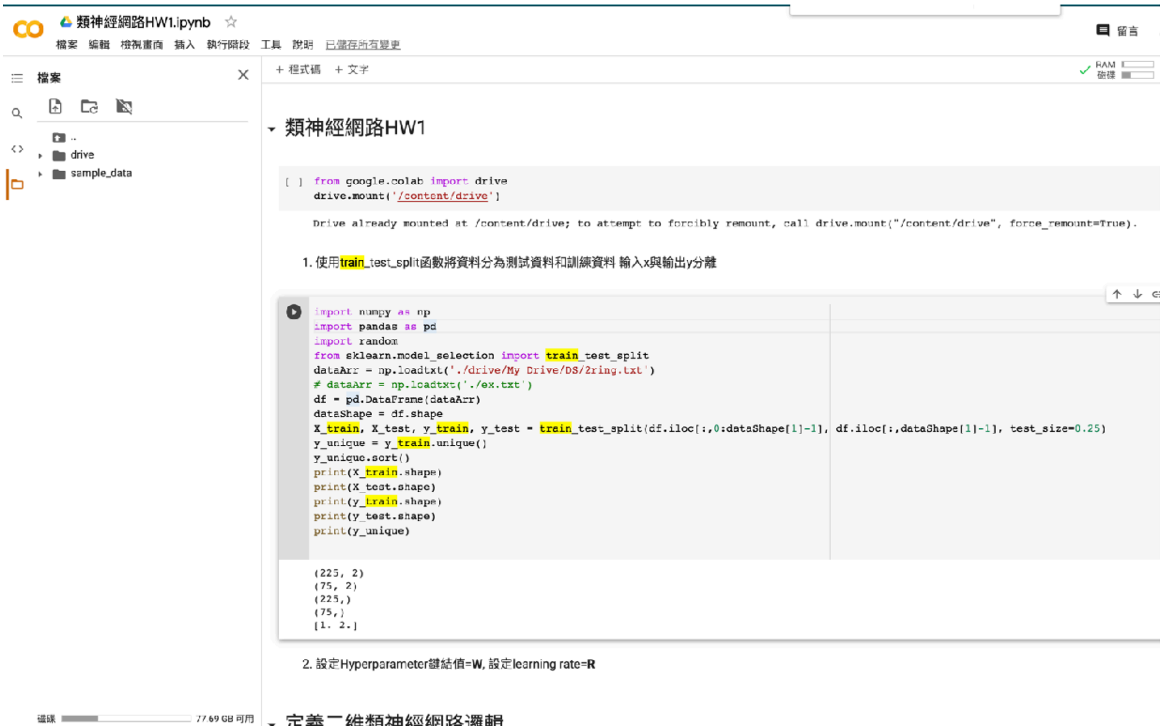
點進每一個並按下 Shift+Enter 即可執行, 程式必須從上執行到下, 其中, 第二個cell 裡面的程式碼決定使用的資料：

dataArr = np.loadtxt(‘./sample_data/2ring.txt’)



第一個cell 是我自己在用的時候連結我的google drive

若助
教沒
有要
連到



類神經網路HW1

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

1. 使用train_test_split函數將資料分為測試資料和訓練資料 輸入x與輸出y分離

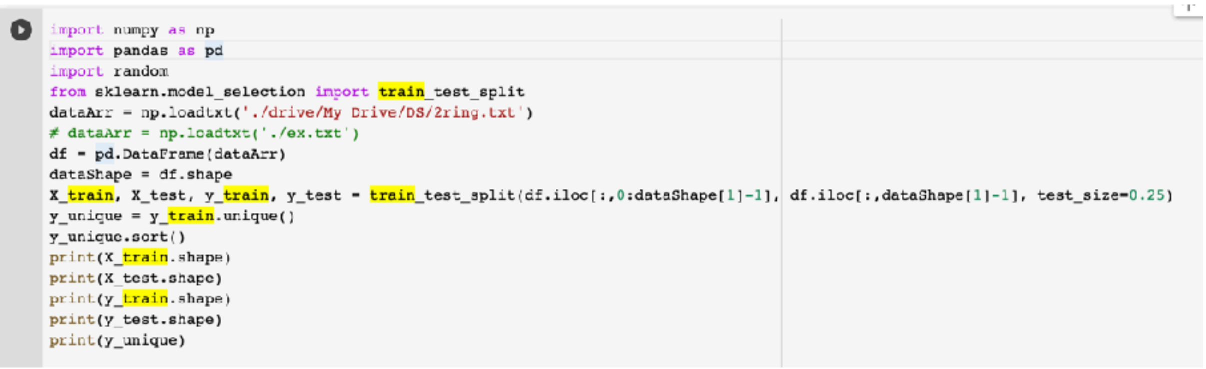
```
import numpy as np
import pandas as pd
import random
from sklearn.model_selection import train_test_split
dataArr = np.loadtxt('./drive/My Drive/DS/2ring.txt')
# dataArr = np.loadtxt('./ex.txt')
df = pd.DataFrame(dataArr)
dataShape = df.shape
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,0:dataShape[1]-1], df.iloc[:,dataShape[1]-1], test_size=0.25)
y_unique = y_train.unique()
y_unique.sort()
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
print(y_unique)
```

```
(225, 2)
(75, 2)
(225,)
(75,)
[1. 2.]
```

2. 設定Hyperparameter鍵結值=W, 設定learning rate=R

定義二維類神經網路邏輯

google drive 就不用執行



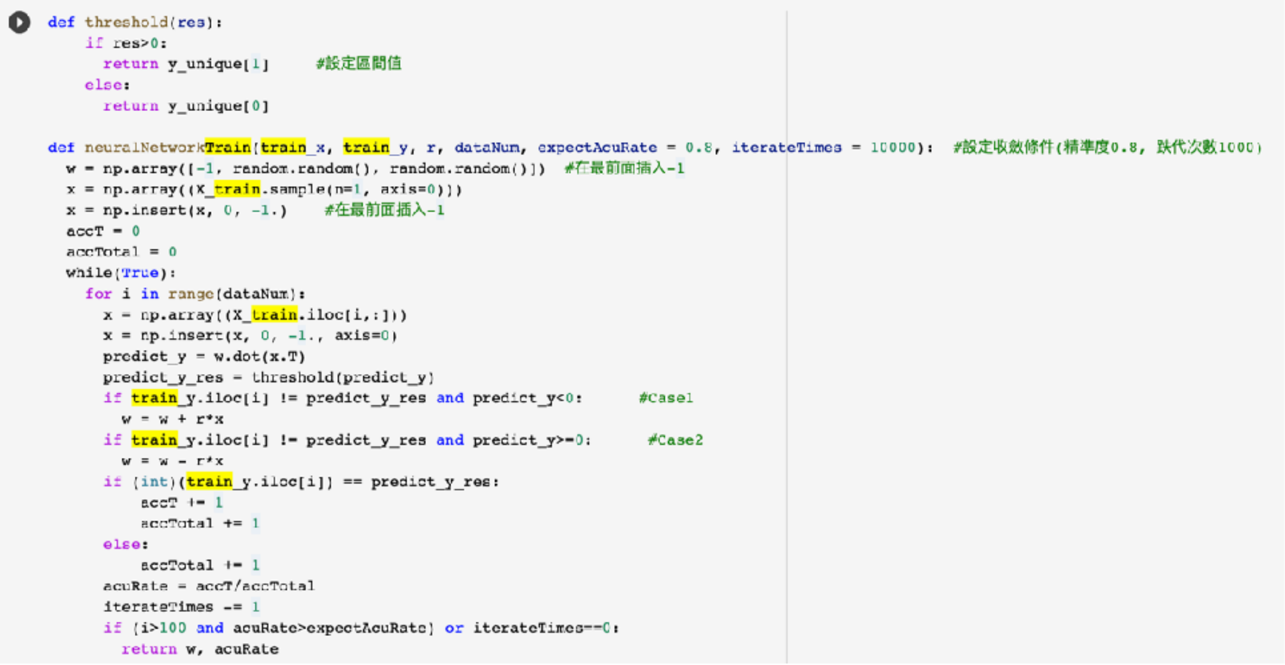
```
import numpy as np
import pandas as pd
import random
from sklearn.model_selection import train_test_split
dataArr = np.loadtxt('./drive/My Drive/DS/2ring.txt')
# dataArr = np.loadtxt('./ex.txt')
df = pd.DataFrame(dataArr)
dataShape = df.shape
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,0:dataShape[1]-1], df.iloc[:,dataShape[1]-1], test_size=0.25)
y_unique = y_train.unique()
y_unique.sort()
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
print(y_unique)
```

B, 程式執行簡介

請助教先上傳檔案到sample_data在更改路徑即可依序執行

X_train y_train是訓練資料

X_test y_test是測試資料



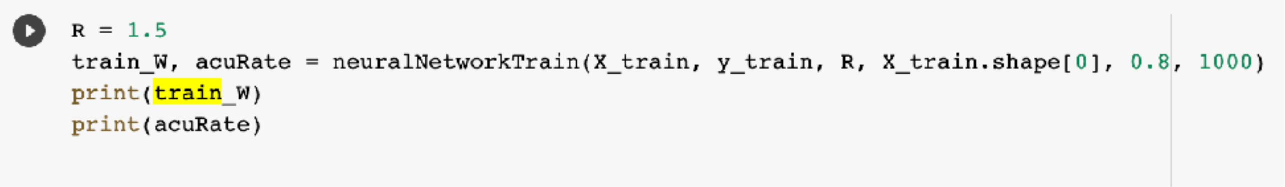
```
def threshold(res):
    if res>0:
      return y_unique[1]      #設定區間值
    else:
      return y_unique[0]

def neuralNetworkTrain(train_x, train_y, r, dataNum, expectAcuRate = 0.8, iterateTimes = 10000):  #設定收斂條件(精準度0.8, 跌代次數1000)
  w = np.array([-1, random.random(), random.random()])  #在最前面插入-1
  x = np.array((X_train.sample(n=1, axis=0)))
  x = np.insert(x, 0, -1.)     #在最前面插入-1
  accT = 0
  accTotal = 0
  while(True):
    for i in range(dataNum):
      x = np.array((X_train.iloc[i,:]))
      x = np.insert(x, 0, -1., axis=0)
      predict_y = w.dot(x.T)
      predict_y_res = threshold(predict_y)
      if train_y.iloc[i] != predict_y_res and predict_y<0:       #Case1
        w = w + r*x
      if train_y.iloc[i] != predict_y_res and predict_y>=0:       #Case2
        w = w - r*x
      if (int)(train_y.iloc[i]) == predict_y_res:
          accT += 1
          accTotal += 1
      else:
          accTotal += 1
      acuRate = accT/accTotal
      iterateTimes -= 1
      if (i>100 and acuRate>expectAcuRate) or iterateTimes==0:
        return w, acuRate
```

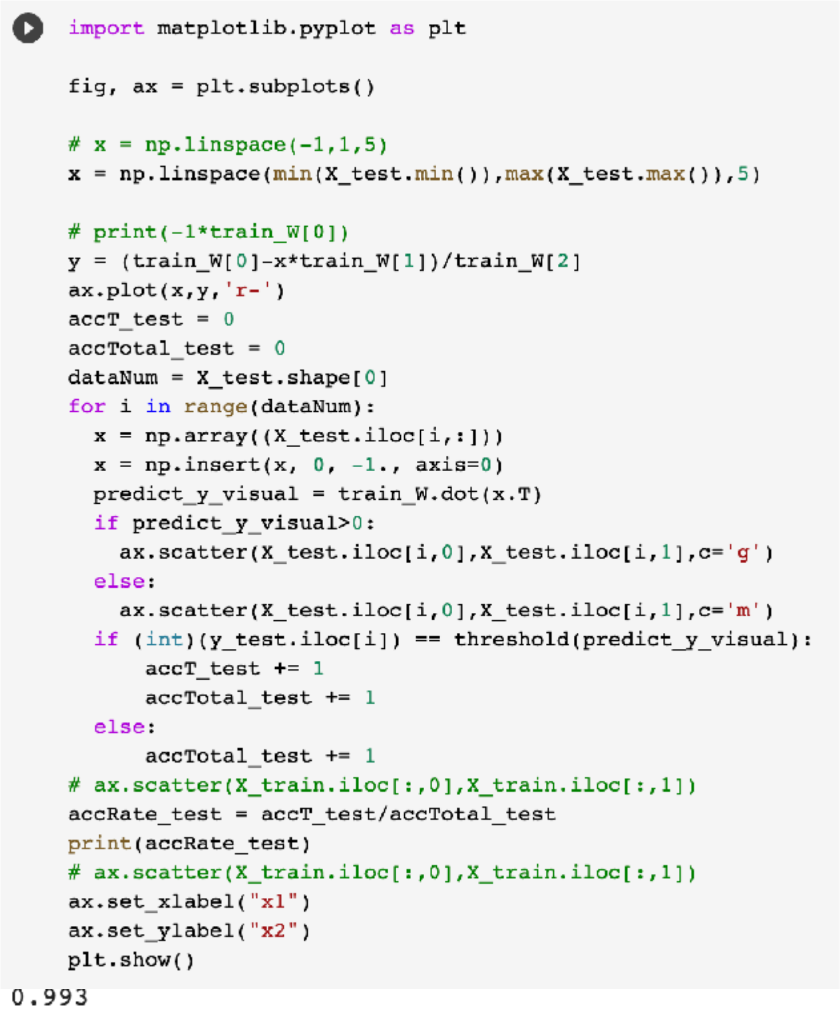
R 是學習率

0.8 是準確率 (收斂條件)

1000 是訓練次數 (收斂條件)



```
R = 1.5
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```



```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

# x = np.linspace(-1,1,5)
x = np.linspace(min(X_test.min()),max(X_test.max()),5)

# print(-1*train_W[0])
y = (train_W[0]-x*train_W[1])/train_W[2]
ax.plot(x,y,'r-')
accT_test = 0
accTotal_test = 0
dataNum = X_test.shape[0]
for i in range(dataNum):
  x = np.array((X_test.iloc[i,:]))
  x = np.insert(x, 0, -1., axis=0)
  predict_y_visual = train_W.dot(x.T)
  if predict_y_visual>0:
    ax.scatter(X_test.iloc[i,0],X_test.iloc[i,1],c='g')
  else:
    ax.scatter(X_test.iloc[i,0],X_test.iloc[i,1],c='m')
  if (int)(y_test.iloc[i]) == threshold(predict_y_visual):
      accT_test += 1
      accTotal_test += 1
  else:
      accTotal_test += 1
# ax.scatter(X_train.iloc[:,0],X_train.iloc[:,1])
accRate_test = accT_test/accTotal_test
print(accRate_test)
# ax.scatter(X_train.iloc[:,0],X_train.iloc[:,1])
ax.set_xlabel("x1")
ax.set_ylabel("x2")
plt.show()
```

```
0.993
```

上面這個cell執行完

會顯示訓練出來的鍵結值 train_W

以及準確率

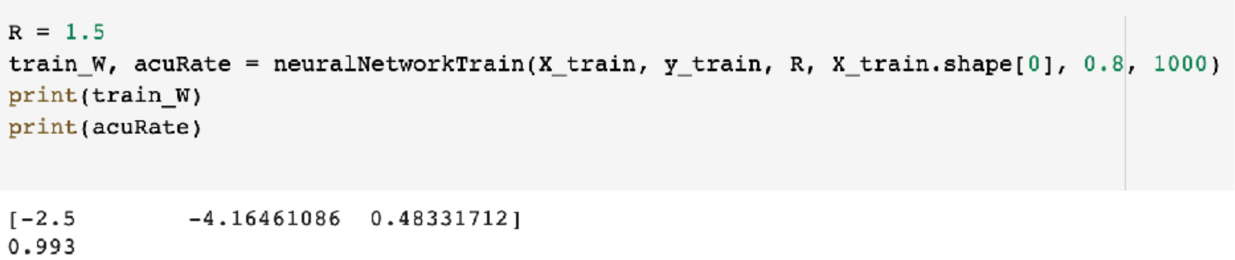
再來執行上面這個cell 會出現測試準確率及測試的結果圖

C, 實驗結果

每個實驗結果均會附上學習率R 鍵結值W 訓練準確率 測試準確率 收斂條件

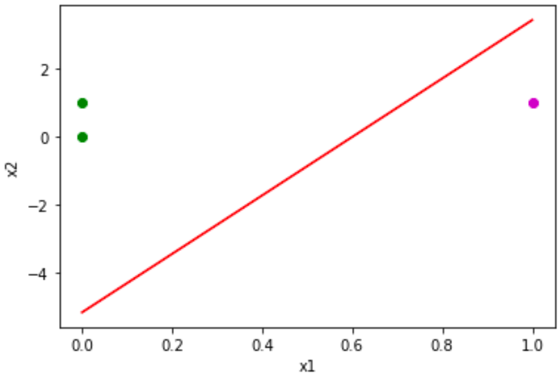
左邊的圖是測試結果, 右邊是訓練結果

(1)perceptron1

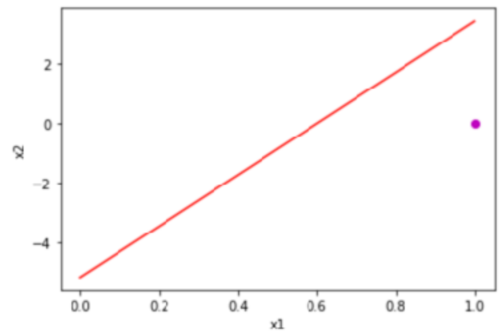


```
R = 1.5
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

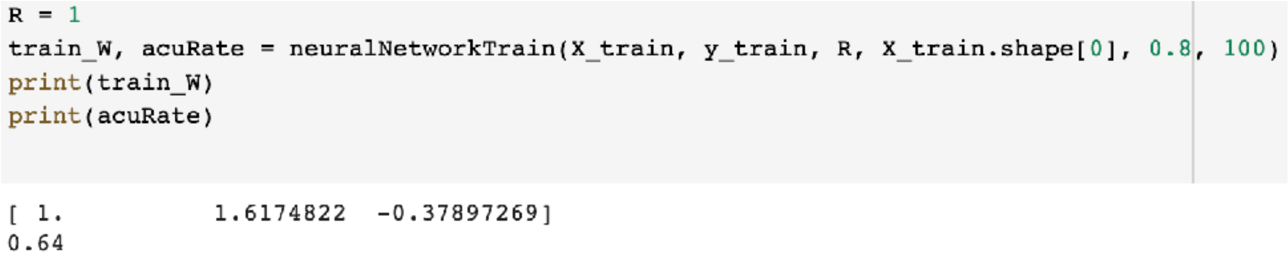
```
[-2.5        -4.16461086  0.48331712]
0.993
```



```
1.0
```

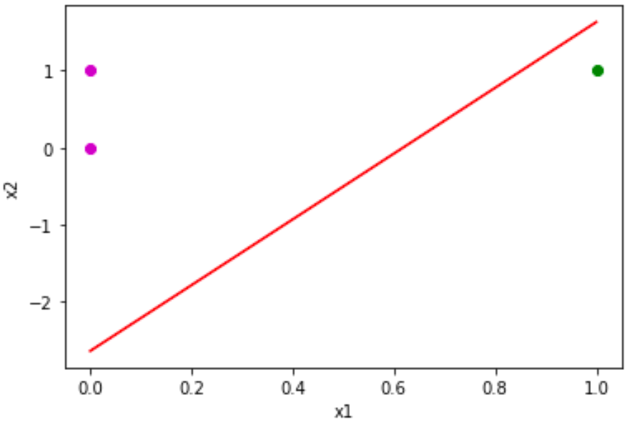


(2)perceptron2

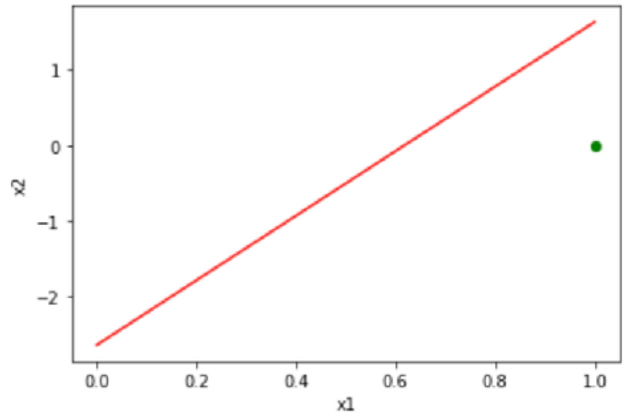


```
R = 1
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 100)
print(train_W)
print(acuRate)
```

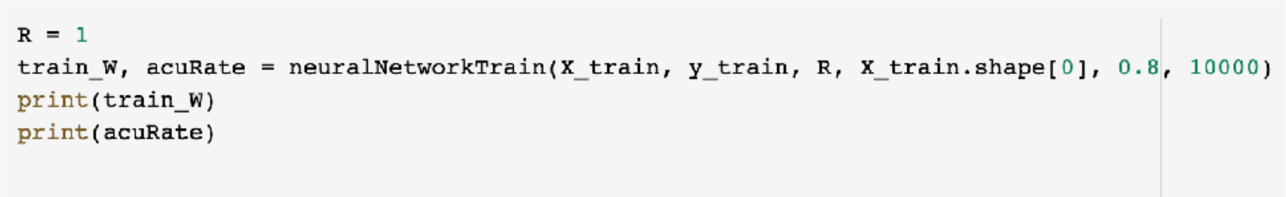
```
[ 1.          1.6174822  -0.37897269]
0.64
```



```
0.0
```

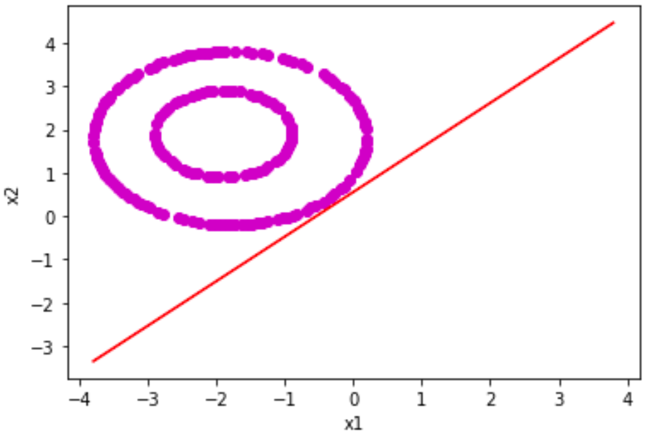


(3)2Ccircle1

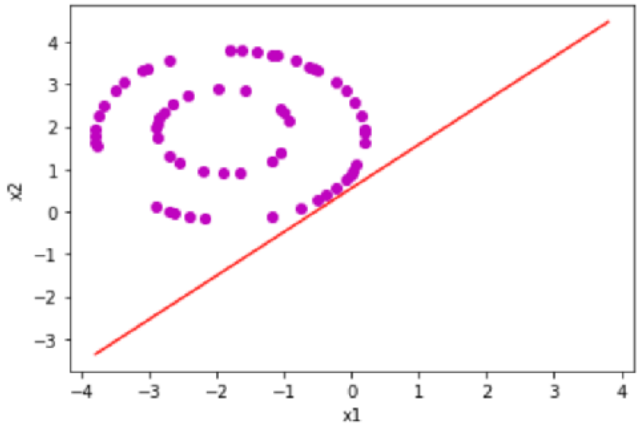


```
R = 1
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 10000)
print(train_W)
print(acuRate)
```

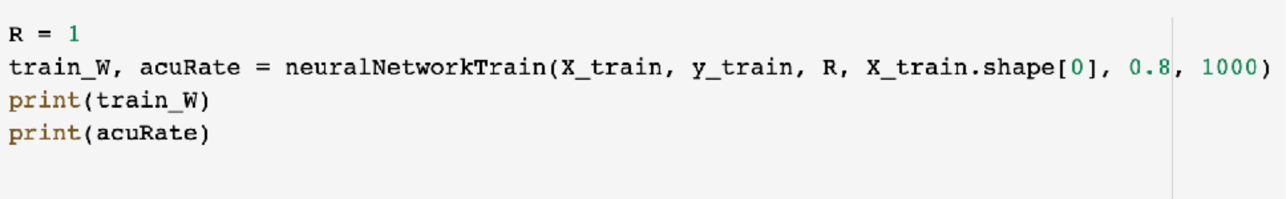
```
[-1.          1.85610976 -1.80403662]
0.5332
```



```
0.31666666666666665
```



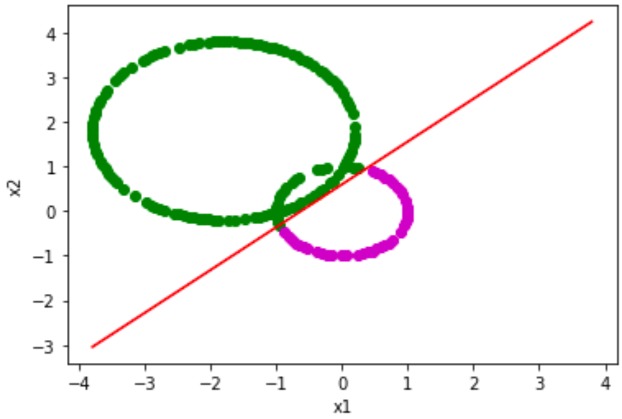
(4)2Circle1

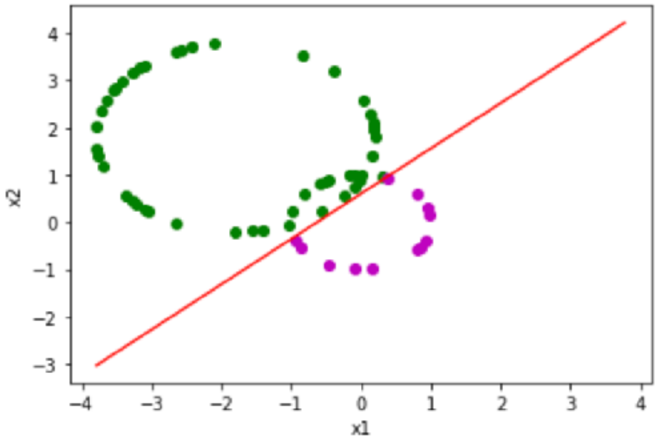


```
R = 1
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

```
[ 1.         -1.60171475  1.67146391]
0.8235294117647058
```

```
0.85
```

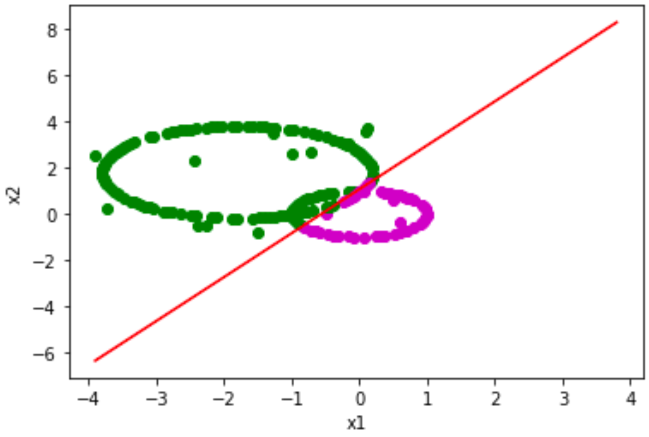




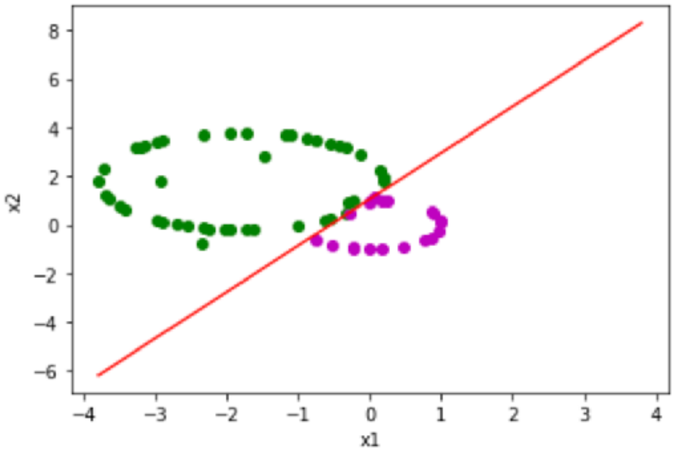
(5)2 Circle2

```
R = 2
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

```
[ 3.         -5.39496891  2.82985468]
0.696
```



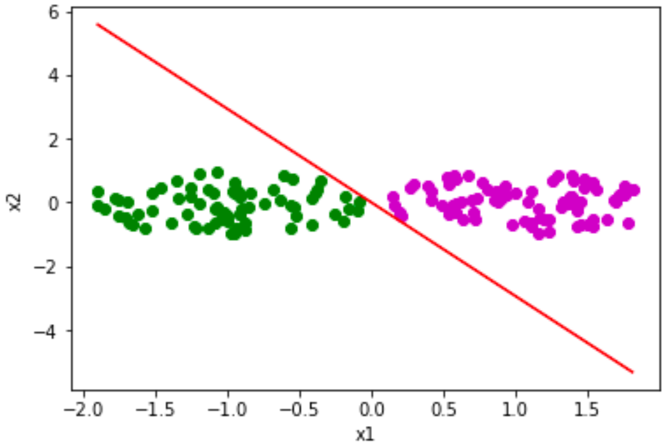
```
0.8307692307692308
```



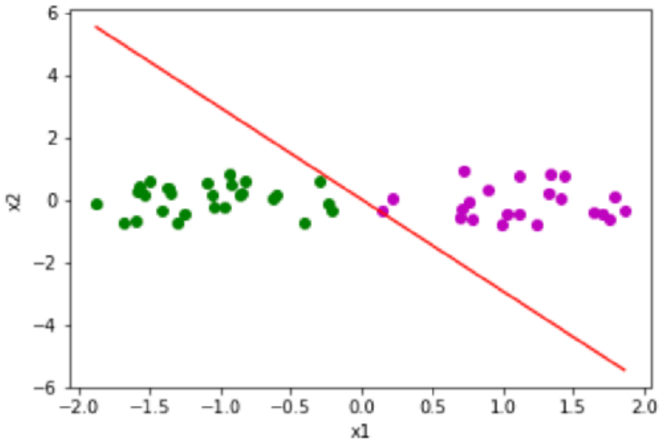
(6)2CloseS

```
R = 1
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

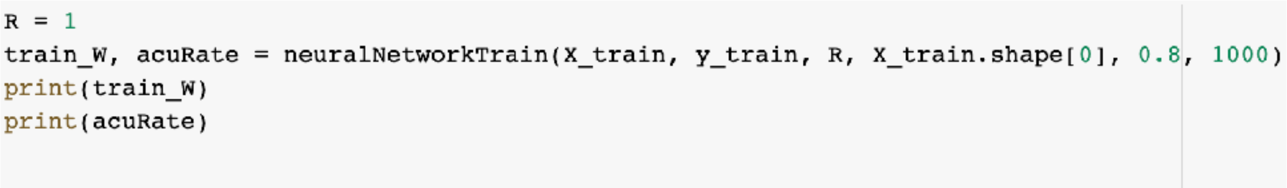
```
[ 0.         -1.80798239 -0.61552292]
0.9705882352941176
```



```
1.0
```

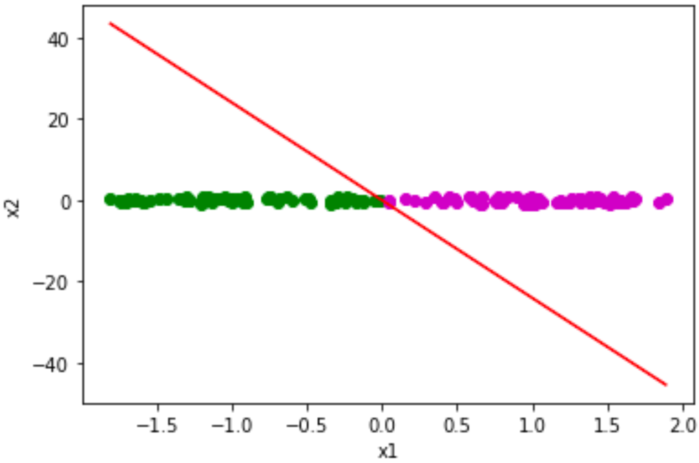


(7)2CloseS2

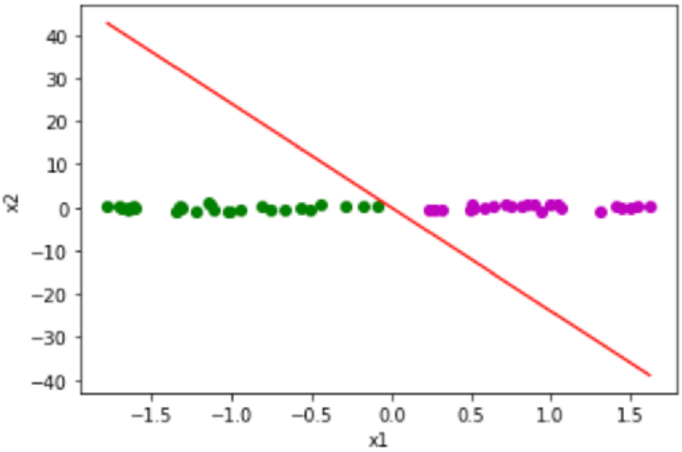


```
R = 1
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

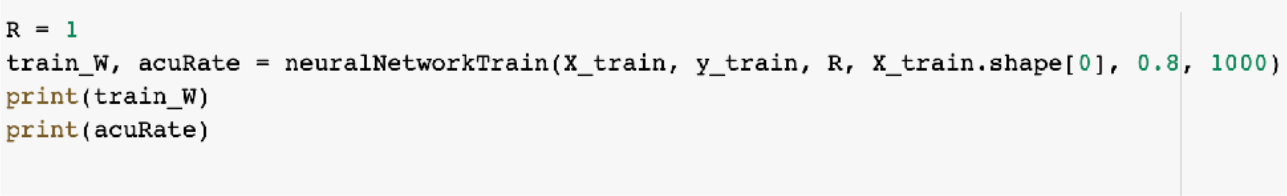
```
[ 0.         -2.63887346 -0.11000539]
0.9313725490196079
```



```
1.0
```

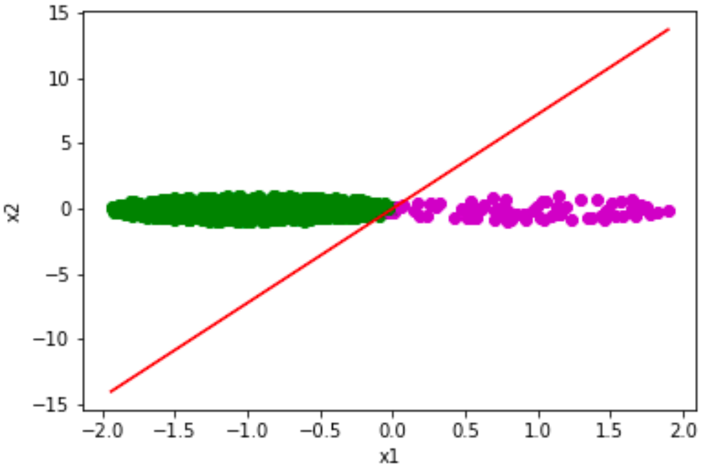


(8)2CloseS3

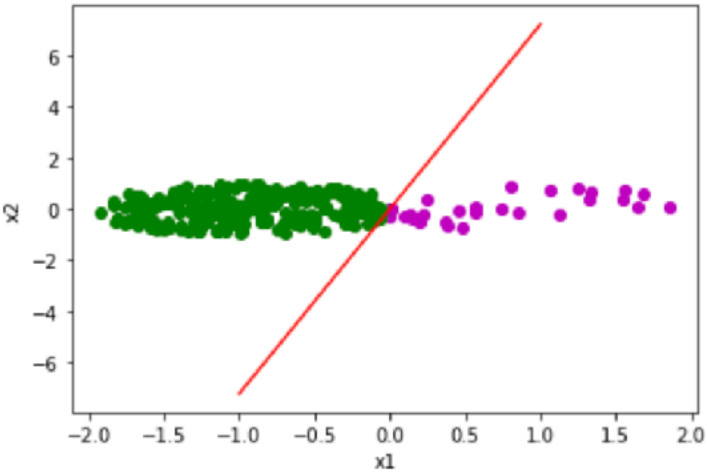


```
R = 1
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

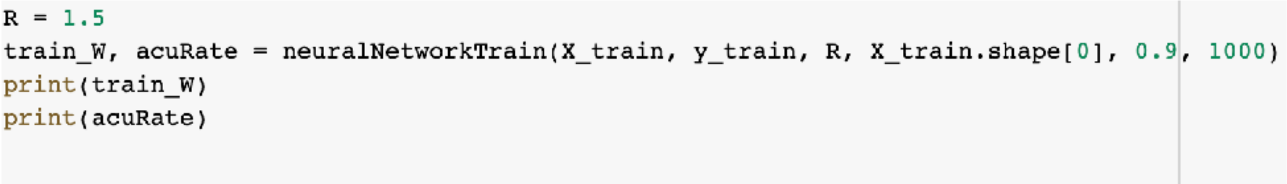
```
[ 0.         -1.8323606   0.25370207]
0.9509803921568627
```



```
0.988
```



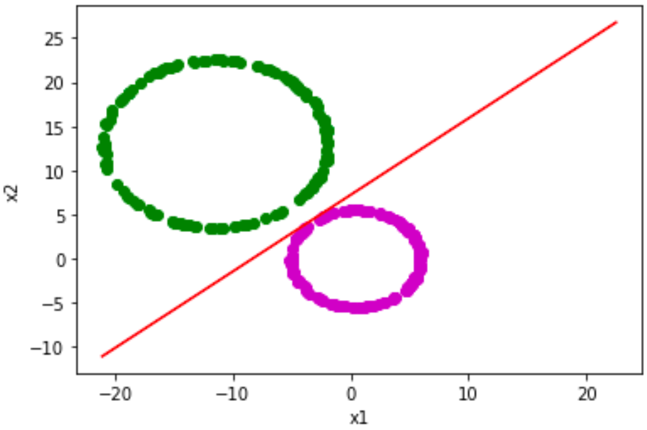
(9)2cring

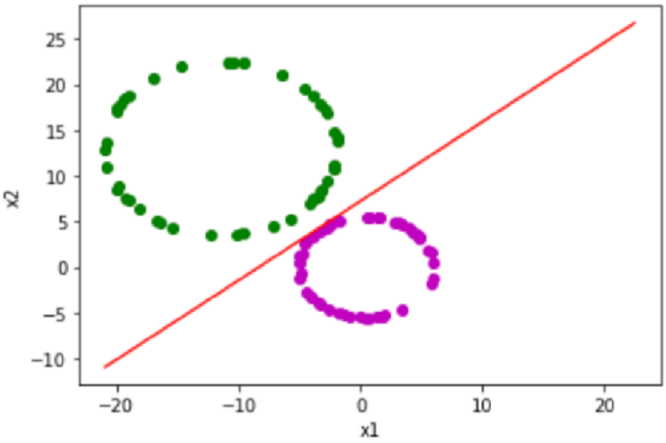


```
R = 1.5
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.9, 1000)
print(train_W)
print(acuRate)
```

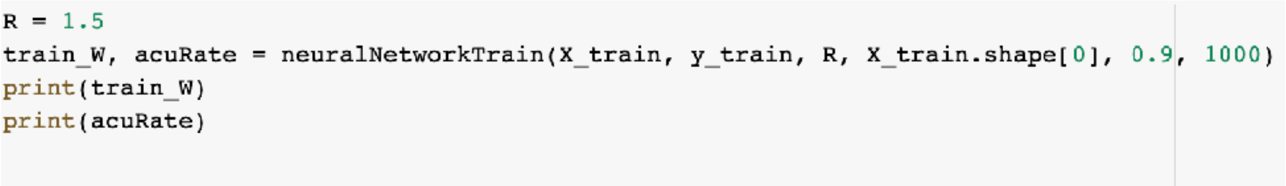
```
[41.          -4.91592467   5.66858529]
0.9001996007984032
```

```
1.0
```





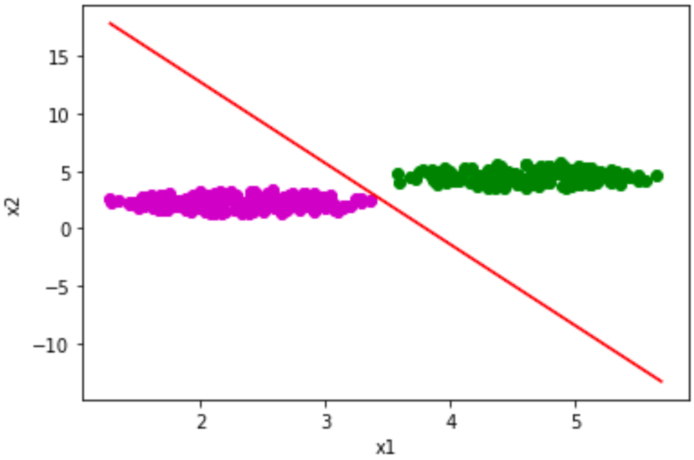
(10)2CS

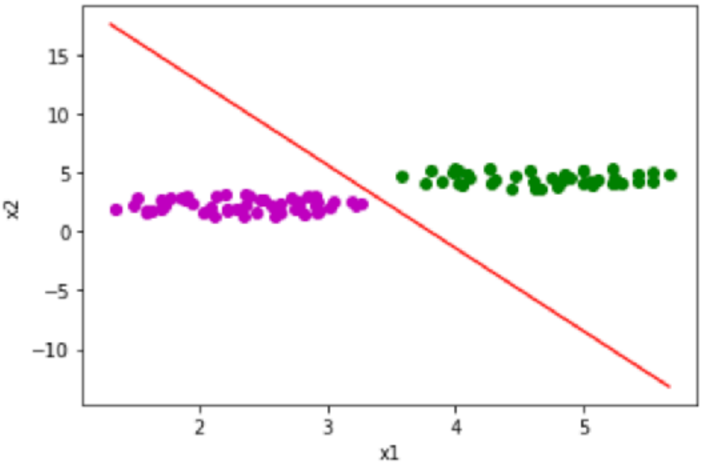


```
R = 1.5
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.9, 1000)
print(train_W)
print(acuRate)
```

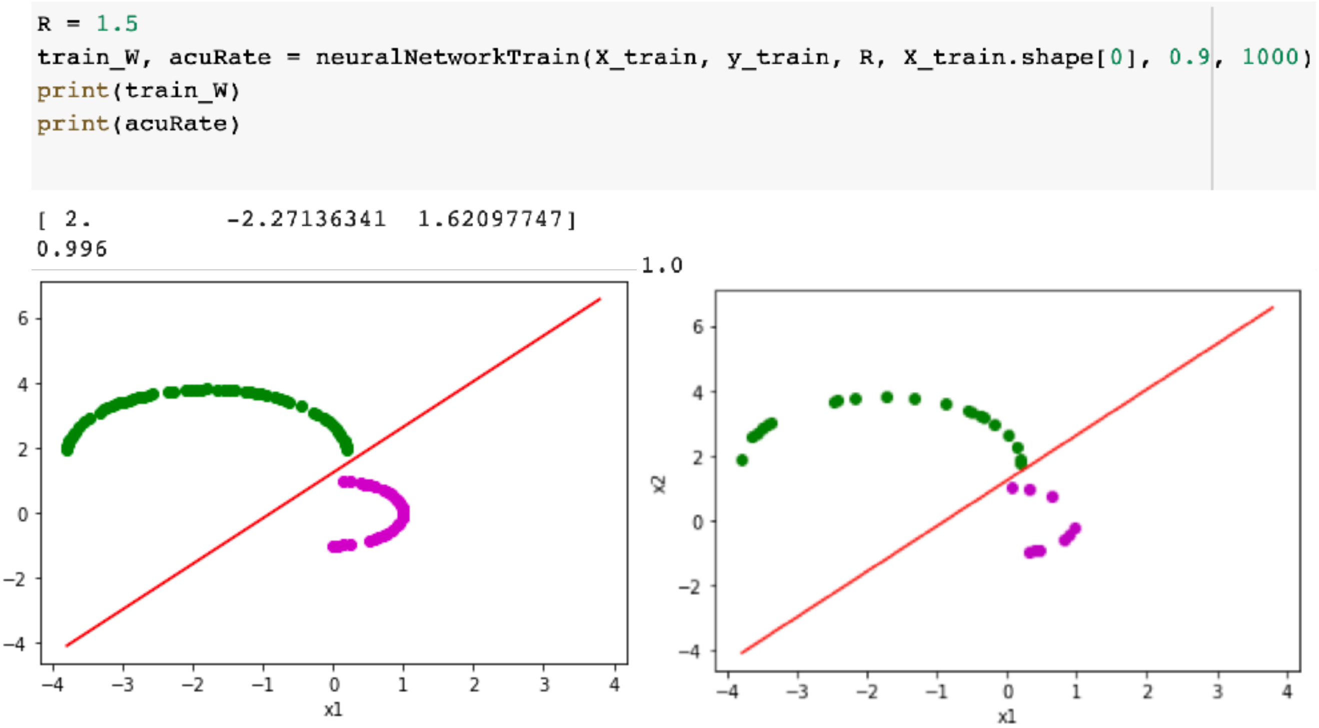
```
[18.5          4.86470114   0.68926564]
0.9002320185614849
```

```
1.0
```





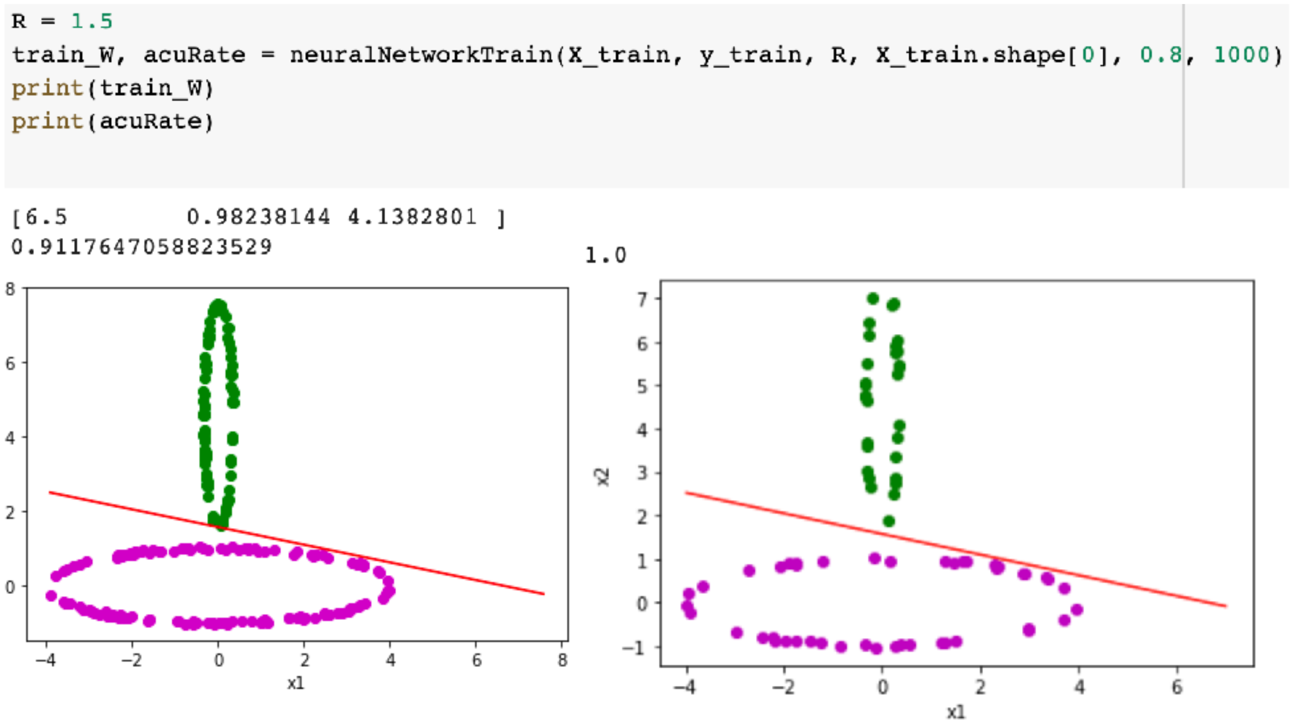
(11)2Hcircle1



```
R = 1.5
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.9, 1000)
print(train_W)
print(acuRate)
```

```
[ 2.         -2.27136341  1.62097747]
0.996
1.0
```

(12)2ring



```
R = 1.5
train_W, acuRate = neuralNetworkTrain(X_train, y_train, R, X_train.shape[0], 0.8, 1000)
print(train_W)
print(acuRate)
```

```
[6.5        0.98238144 4.1382801 ]
0.9117647058823529
1.0
```

D, 實驗結果分析及討論

我目前實作的是感知機的範例, 在做2Ccircle1的時候會有困難, 因為他是非線性的，這可能要用 二層感知機 或是先 將資料正規化 再做分類, 經過多次選擇學習率的經驗, 認為資料之間的差距和學習率的大小有關, 也認為如果同一批資料經過太多次的訓練其實並不會有太大的進步, 因此適當的學習次數也是很重要的。