碩管一甲　M09218001　周彥廷

高等類神經網路作業：0205 課輔練習

1.實驗調整初始化權重方法 例如我是用 uniform 分佈初始化 w,改成 normal 分佈會有什麼影響?？

2.實驗調整激勵函數 或是不要加激勵函數 會有什麼影響？

3.實驗調整學習率(lr)會有什麼影響？

1.導入套件

```
# jax 可以想成有微分功能的 numpy
import jax
import jax.numpy as jnp
from jax import random
import numpy as np

# 進度條
from tqdm import tqdm_notebook as tqdm

# 機器學習套件庫 sklearn
from sklearn.metrics import accuracy_score
from sklearn.datasets import make_blobs

# 繪圖用
import matplotlib.pyplot as plt

# 整理&操作表格數據用
import pandas as pd

# 畫出散點圖用
def plot_data(class_zero,class_one):
    plt.scatter(class_zero.x1,class_zero.x2,label='class_zero',color='red')
    plt.scatter(class_one.x1,class_one.x2,label='class_one',color='blue')
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.legend()

# 畫出決策表面用
def plot_decision_surface(X,predict_fn):
```

```
    min1, max1 = X[:, 0].min()-1, X[:, 0].max()+1
    min2, max2 = X[:, 1].min()-1, X[:, 1].max()+1
    x1grid = np.arange(min1, max1, 0.05)
    x2grid = np.arange(min2, max2, 0.05)
    xx, yy = np.meshgrid(x1grid, x2grid)
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    grid = np.hstack((r1,r2))
    yhat = predict_fn(grid)
    zz = yhat.reshape(xx.shape)
    plt.contourf(xx,yy,zz,cmap='Paired')
    plt.contour(xx,yy,zz,color='black',lw=0.5)
```

2.製作數據

```
X, y = make_blobs(n_samples=100,centers=2, n_features=2,random_state=0) #random_state = 0 確保實驗結果依樣
df = pd.DataFrame()
df['x1'] = X[:,0]
df['x2'] = X[:,1]
df['y'] = y
df.head()
```

3.定義感知器模型參數

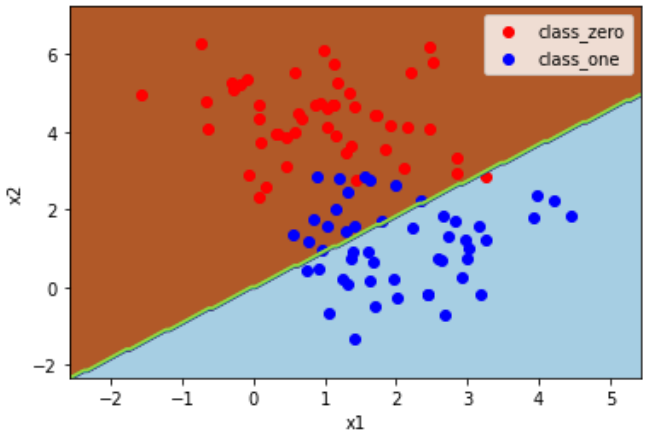
Q1.實驗調整初始化權重方法 例如我是用 uniform 分佈初始化 w,改成 normal 分佈會有什麼影響?

```
key = random.PRNGKey(0)   #random seed
params = {
   'w': random.uniform(key,minval=0.0,maxval=1.0,shape=(2,)),
   'b': 0.
}

key = random.PRNGKey(0)
params_normal = {
   'w': random.normal(key,shape=(2,)), #使用 normal 分佈
   'b': 0.
}
params_normal
```

訓練前結果：0.47→0.17

```
plot_decision_surface(df[['x1','x2']].values,lambda x:(forward(params,x)>=0.5).astype(int))
#params,x 改參數
plot_data(df[df.y == 0],df[df.y == 1])
plt.show()
```



4.訓練感知器模型(梯度下降法)

```
def J(params):
  y_pred = forward(params,X) # 預測值
  loss = jnp.mean(jnp.square(y_pred - y)) # 預測值跟真實值之間的 mse loss
  return loss
loss = J(params)
print('loss:',loss)
```

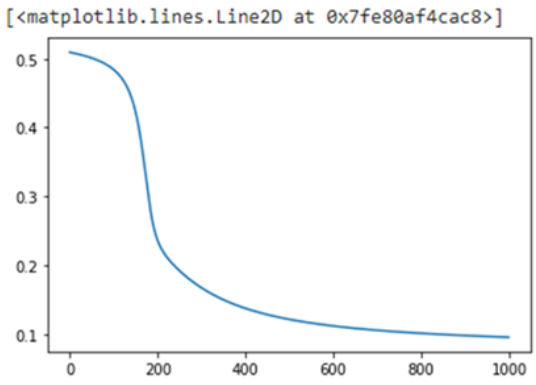
loss: 0.50943154→0.6486839

#這裡 α 代表 lr(learning rate)學習率,也是可以改的大家可以改看看

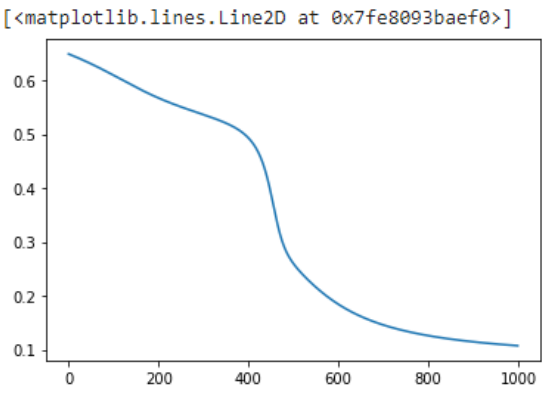
舊參數 {'w': DeviceArray([-0.784766 , 0.8564448], dtype=float32), 'b': 0.0}

新參數: {'b': DeviceArray(-0.2745567, dtype=float32), 'w': DeviceArray([1.0169114 , -0.27104917], dtype=float32)}

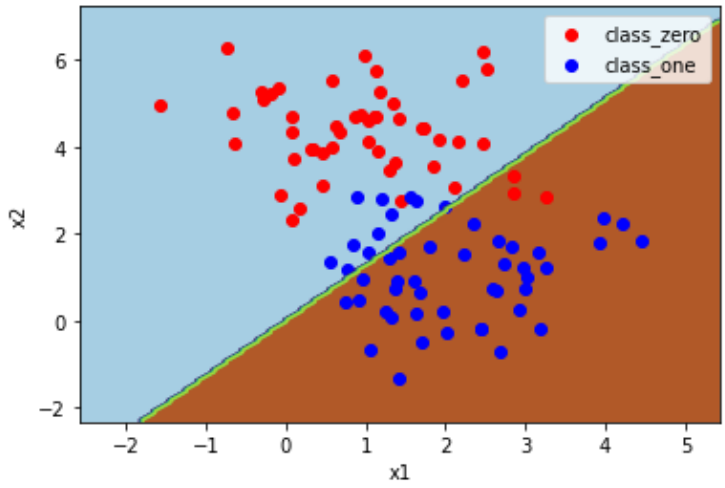
#改變前



#改變後



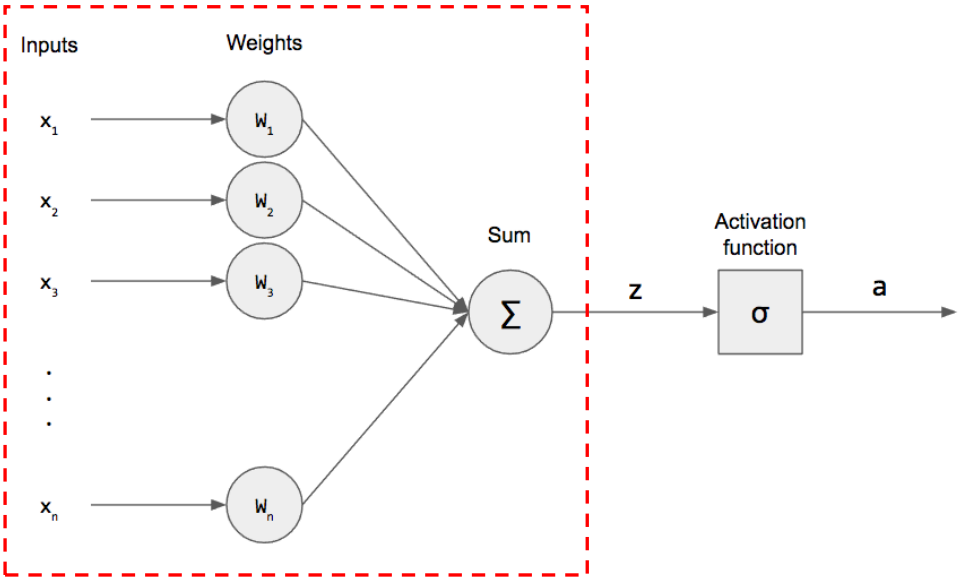
#評估感知器模型(訓練後): 0.87→0.86



Q2.實驗調整激勵函數或是不要加激勵函數 會有什麼影響？

若不加激勵函數，即為基本的 Perceptron，僅能產生簡單線性組合結果，無法進行非線性函數轉換。

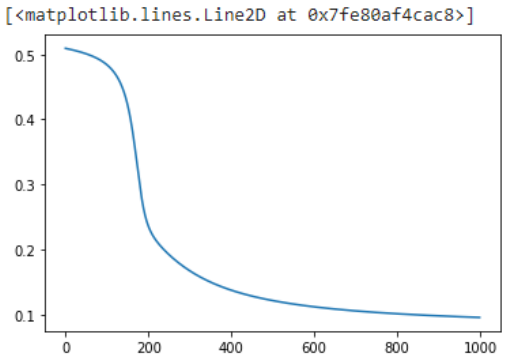
EX：不加激勵函數如下圖 output 只會得到 Z 值($\sum X_1W_1+X_2W_2+\ldots X_nW_n$)

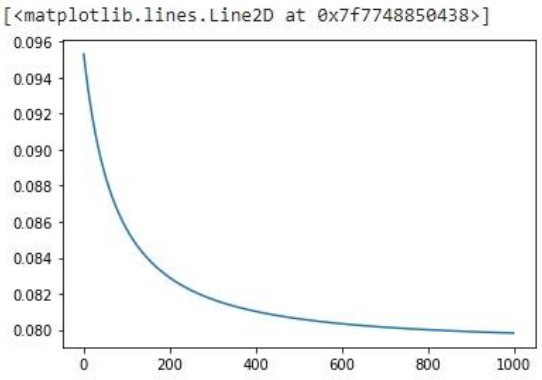


Q3.實驗調整學習率(lr)會有什麼影響？

```
def update(params, grads,lr = 0.1):
# lr 代表學習率 為 0.01(0.87),   0.05(0.87),   0.1(0.88)
 new_params = jax.tree_multimap(lambda p, g: p - lr * g, params, grads)
  return new_params
```

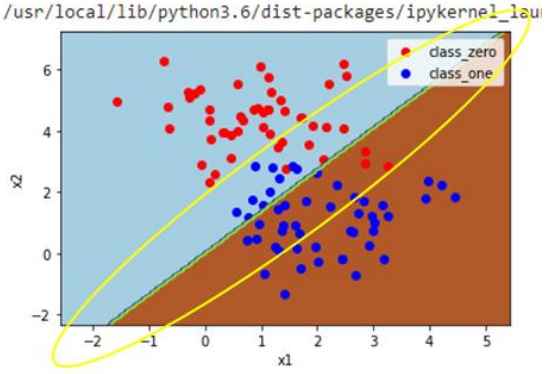
#觀察 loss 下降曲線：lr = 0.01　　lr = 0.1

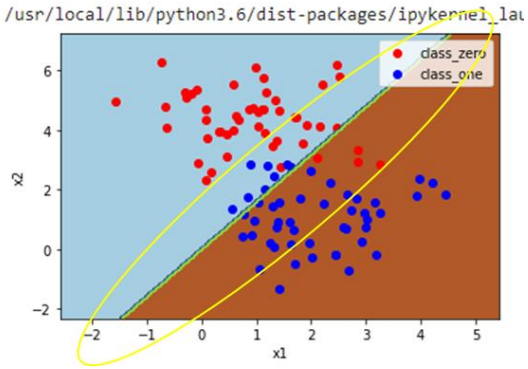




#可視化預測結果：

(lr)=0.01, accuracy = 0.87　　(lr)=0.01, accuracy = 0.88





結論：學習率調整為 0.01(0.87)，0.05(0.87)與 0.1(0.88)，顯示三種學習率準確率差不多，準確率影響原因由決策邊界(Decision boundary)處(黃色區間)之藍色正確分類數目與紅色正確分類數目比例決定。