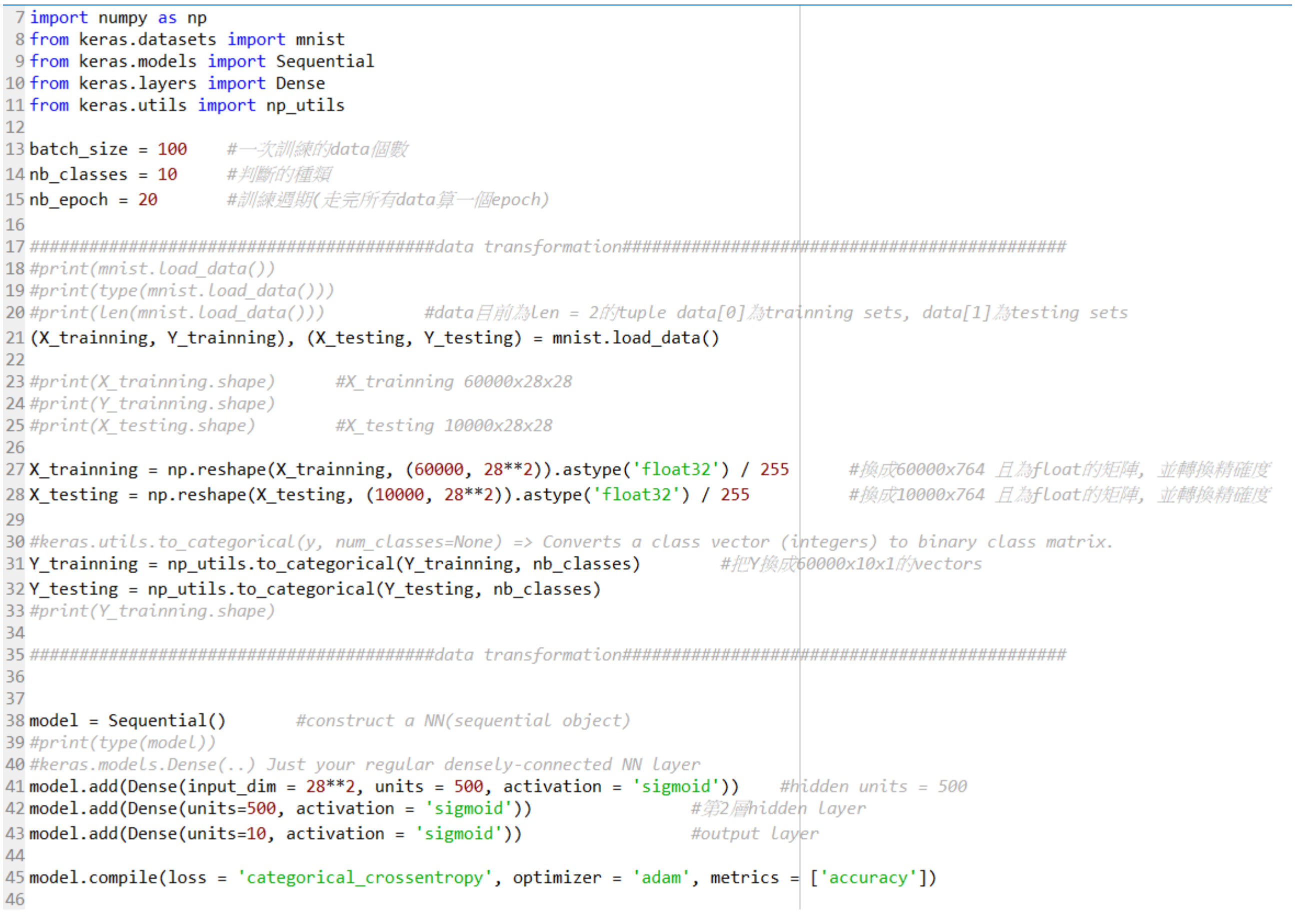
機器學習 HW6

姓名:林士恩

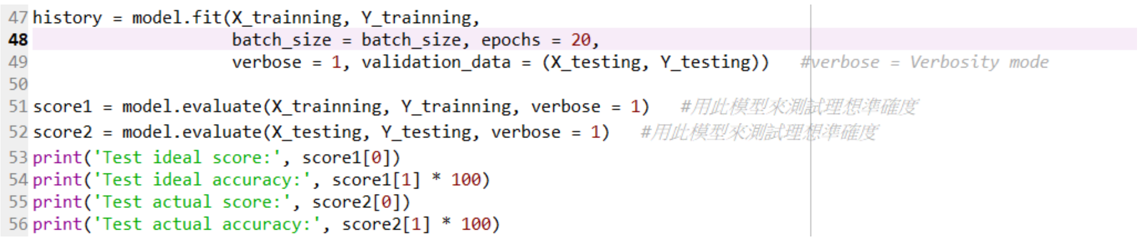
學號:B043011031

系級:108 電機甲

一、原始程式碼:



```python
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils

batch_size = 100    #一次訓練的data個數
nb_classes = 10     #判斷的種類
nb_epoch = 20       #訓練週期(走完所有data算一個epoch)

#########################################data transformation#############################################
#print(mnist.load_data())
#print(type(mnist.load_data()))
#print(len(mnist.load_data()))          #data目前為len = 2的tuple data[0]為trainning sets, data[1]為testing sets
(X_trainning, Y_trainning), (X_testing, Y_testing) = mnist.load_data()

#print(X_trainning.shape)      #X_trainning 60000x28x28
#print(Y_trainning.shape)
#print(X_testing.shape)        #X_testing 10000x28x28

X_trainning = np.reshape(X_trainning, (60000, 28**2)).astype('float32') / 255      #換成60000x764 且為float的矩陣, 並轉換精確度
X_testing = np.reshape(X_testing, (10000, 28**2)).astype('float32') / 255          #換成10000x764 且為float的矩陣, 並轉換精確度

#keras.utils.to_categorical(y, num_classes=None) => Converts a class vector (integers) to binary class matrix.
Y_trainning = np_utils.to_categorical(Y_trainning, nb_classes)        #把Y換成60000x10x1的vectors
Y_testing = np_utils.to_categorical(Y_testing, nb_classes)
#print(Y_trainning.shape)

#########################################data transformation#############################################


model = Sequential()       #construct a NN(sequential object)
#print(type(model))
#keras.models.Dense(..) Just your regular densely-connected NN layer
model.add(Dense(input_dim = 28**2, units = 500, activation = 'sigmoid'))    #hidden units = 500
model.add(Dense(units=500, activation = 'sigmoid'))                #第2層hidden layer
model.add(Dense(units=10, activation = 'sigmoid'))                 #output layer

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

```



```python
history = model.fit(X_trainning, Y_trainning,
                    batch_size = batch_size, epochs = 20,
                    verbose = 1, validation_data = (X_testing, Y_testing))    #verbose = Verbosity mode

score1 = model.evaluate(X_trainning, Y_trainning, verbose = 1)    #用此模型來測試理想準確度
score2 = model.evaluate(X_testing, Y_testing, verbose = 1)    #用此模型來測試理想準確度
print('Test ideal score:', score1[0])
print('Test ideal accuracy:', score1[1] * 100)
print('Test actual score:', score2[0])
print('Test actual accuracy:', score2[1] * 100)
```

Console:

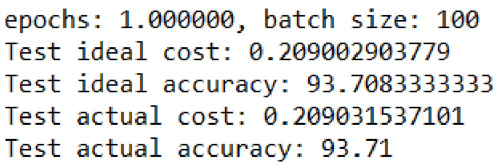
```
Test ideal score: 0.0108949495773
Test ideal accuracy: 99.6066666667
Test actual score: 0.109480199174
Test actual accuracy: 97.62
```

(當 epoch = 20, input units = 500, activation function = sigmoid)

注:softmax 只適合當成最後一層 layer 的激活方程式，使輸入的數值更容易能區分，判斷結果。

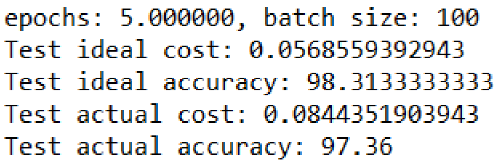
二、問題討論:

1.調整參數對於訓練模型的影響:



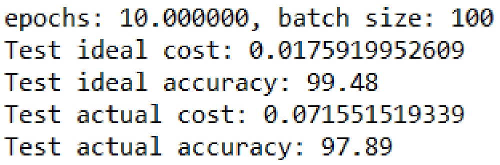
```
epochs: 1.000000, batch size: 100
Test ideal cost: 0.209002903779
Test ideal accuracy: 93.7083333333
Test actual cost: 0.209031537101
Test actual accuracy: 93.71
```

(activation func. 皆為 sigmoid)



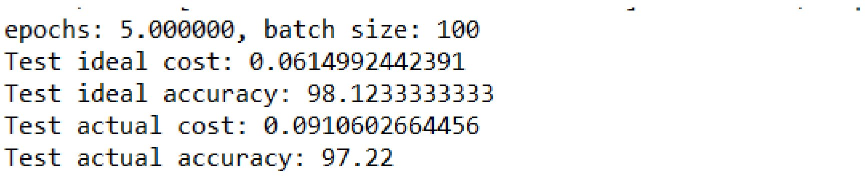
```
epochs: 5.000000, batch size: 100
Test ideal cost: 0.0568559392943
Test ideal accuracy: 98.3133333333
Test actual cost: 0.0844351903943
Test actual accuracy: 97.36
```

(activation func. 皆為 sigmoid)



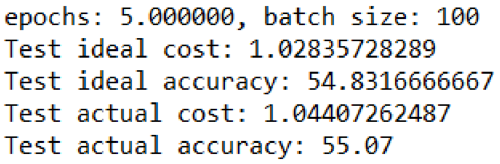
```
epochs: 10.000000, batch size: 100
Test ideal cost: 0.0175919952609
Test ideal accuracy: 99.48
Test actual cost: 0.071551519339
Test actual accuracy: 97.89
```

(activation func. 皆為 sigmoid)



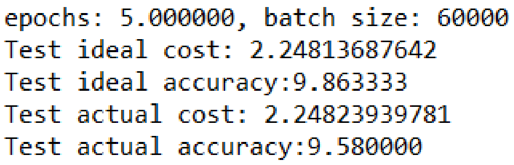
```
epochs: 5.000000, batch size: 100
Test ideal cost: 0.0614992442391
Test ideal accuracy: 98.1233333333
Test actual cost: 0.0910602664456
Test actual accuracy: 97.22
```

(output layer 利用 softmax function 來激活)



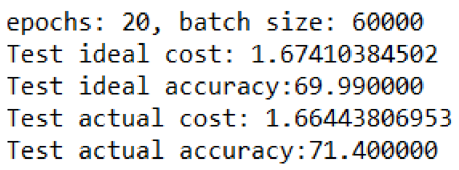
```
epochs: 5.000000, batch size: 100
Test ideal cost: 1.02835728289
Test ideal accuracy: 54.8316666667
Test actual cost: 1.04407262487
Test actual accuracy: 55.07
```

(所有 hypothesis 皆為 softmax function, 可以發現到正確率只有 55%)



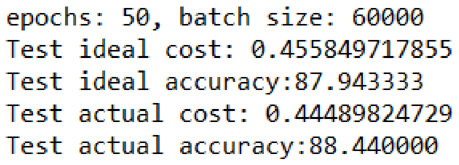
```
epochs: 5.000000, batch size: 60000
Test ideal cost: 2.24813687642
Test ideal accuracy:9.863333
Test actual cost: 2.24823939781
Test actual accuracy:9.580000
```

(full batch 之下, epoch = 5)



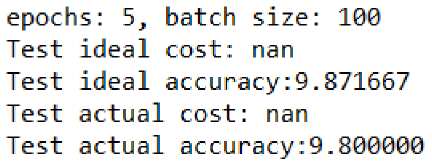
```
epochs: 20, batch size: 60000
Test ideal cost: 1.67410384502
Test ideal accuracy:69.990000
Test actual cost: 1.66443806953
Test actual accuracy:71.400000
```

(full batch 之下, epoch = 20)



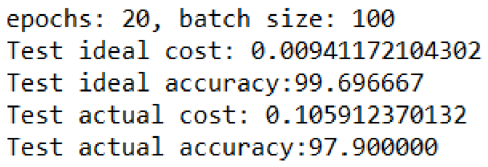
```
epochs: 50, batch size: 60000
Test ideal cost: 0.455849717855
Test ideal accuracy:87.943333
Test actual cost: 0.44489824729
Test actual accuracy:88.440000
```

(full batch 之下, epoch = 50, 可以發現正確率拉回到 88.5%)



```
epochs: 5, batch size: 100
Test ideal cost: nan
Test ideal accuracy:9.871667
Test actual cost: nan
Test actual accuracy:9.800000
```

(output layer 利用 relu 來激活)



```
epochs: 20, batch size: 100
Test ideal cost: 0.00941172104302
Test ideal accuracy:99.696667
Test actual cost: 0.105912370132
Test actual accuracy:97.900000
```

(其中 2 層的 hidden units 都為 750, 可以發現已 overfitting)

由上列多個結果可以觀察到:

Softmax function:只適合放在 output layer，因 softmax 的運用再於把結果間的差距擴大，讓最後的答案更容易判斷。若在中間使用到 softmax func 時，會導致本來互相 linearly independent 的資料間變成 linerly dependent，導致之後的運算會產生問題。

Batch Size:會影響程式運行的速度，如果 batch size 越大，則跑完一個 epoch 也就越快，若 batch size 越小，則程式跑完一個 epoch 就越慢。但是不一定 batch size 越大就越好，如果 batch size 太大，且資料數量也多，可能會導致 gradient descent 的精確度下降，因跑完一個 epoch 所需的迭代次數(gradient descent)次數不夠多，所以要增加 epoch 數量才可以把精確度拉回來，若 batch size 太小，雖然精確度會較高，但是跑 1 epoch 的時間會很久，

導致整個訓練過程時間太久，效率不高。

Epoch:epoch 在這即代表跑完所有資料的次數，若次數太少的話，會因為 gradient 次數不夠多，沒辦法接近 optimal，使正確率下降；若 epoch 次數太多的話，會導致過於接近 optimal，使整個模型 overfitting。

Hidden Units: 代表著訓練數學方程式的 feature 數量(解析度)，可以想成一張照片的解析度問題，1024x760 一定比 500x300 的畫面更好，所以 hidden unit 就可以想成是其中的小方格，數量越多就代表著切得越精細，準確度也會越高。若 hidden Units 太大的話會導致 overfitting