1、2

In [1]:
```
import numpy as np
from keras.utils import np_utils
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
```

In [2]:
```
(X_train, y_train), (X_test, y_test)= tf.keras.datasets.fashion_mnist.load_data()
```

In [3]:
```
# hidden_layer_sizes=(5,3)表示隱藏層有兩層第一層為五個神經元，第二層為三個神經元
#設定分類器:最佳化參數的演算法，alpha值
from sklearn.neural_network import MLPClassifier
nsamples, nx, ny = X_train.shape
d2_train_dataset = X_train.reshape((nsamples,nx*ny))
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(5,3), random_state=1)

clf.fit(d2_train_dataset, y_train)
```

Out[3]:
```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(5, 3), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=1, shuffle=True, solver='lbfgs',
              tol=0.0001, validation_fraction=0.1, verbose=False,
```

In [4]:
```
nsamples, nx, ny = X_test.shape
d2_Xtest_dataset = X_test.reshape((nsamples,nx*ny))
clf.score(d2_Xtest_dataset,y_test)
```

Out[4]:
```
0.0989
```

In [5]:
```
from sklearn.neural_network import MLPClassifier
nsamples, nx, ny = X_train.shape
d2_train_dataset = X_train.reshape((nsamples,nx*ny))
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(5,5), random_state=1)

clf.fit(d2_train_dataset, y_train)
```

Out[5]:
```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(5, 5), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=1, shuffle=True, solver='lbfgs',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

In [6]:
```
nsamples, nx, ny = X_test.shape
d2_Xtest_dataset = X_test.reshape((nsamples,nx*ny))
clf.score(d2_Xtest_dataset,y_test)
```

Out[6]:
```
0.1
```

3.

In [13]:
```
X = np.random.random_sample((100, 8)) # 輸入是 100 筆 8 維的資料
y = np_utils.to_categorical(np.random.randint(3, size=100))
```

In [14]:
```
model.fit(X, y, epochs=20, batch_size=10, verbose=2, validation_split=0.2)
```

```
Epoch 1/20
8/8 - 25s - loss: 1.0950 - accuracy: 0.4000 - val_loss: 1.1070 - val_accuracy: 0.4500
Epoch 2/20
8/8 - 0s - loss: 1.0888 - accuracy: 0.4250 - val_loss: 1.1063 - val_accuracy: 0.4000
Epoch 3/20
8/8 - 0s - loss: 1.0844 - accuracy: 0.4500 - val_loss: 1.1052 - val_accuracy: 0.4000
Epoch 4/20
8/8 - 0s - loss: 1.0817 - accuracy: 0.4500 - val_loss: 1.1056 - val_accuracy: 0.4000
Epoch 5/20
8/8 - 0s - loss: 1.0774 - accuracy: 0.4500 - val_loss: 1.1055 - val_accuracy: 0.3000
Epoch 6/20
8/8 - 0s - loss: 1.0735 - accuracy: 0.4875 - val_loss: 1.1052 - val_accuracy: 0.4000
Epoch 7/20
8/8 - 0s - loss: 1.0719 - accuracy: 0.4875 - val_loss: 1.1059 - val_accuracy: 0.4500
Epoch 8/20
8/8 - 0s - loss: 1.0675 - accuracy: 0.4750 - val_loss: 1.1056 - val_accuracy: 0.4500
Epoch 9/20
8/8 - 0s - loss: 1.0641 - accuracy: 0.4625 - val_loss: 1.1056 - val_accuracy: 0.4000
Epoch 10/20
8/8 - 0s - loss: 1.0615 - accuracy: 0.4750 - val_loss: 1.1060 - val_accuracy: 0.4000
Epoch 11/20
8/8 - 0s - loss: 1.0585 - accuracy: 0.4875 - val_loss: 1.1056 - val_accuracy: 0.4500
Epoch 12/20
8/8 - 0s - loss: 1.0560 - accuracy: 0.5000 - val_loss: 1.1059 - val_accuracy: 0.4500
```

In [15]:
```
model = Sequential()
#dd Input layer, 隱藏層(hidden layer) 有 256個輸出變數
model.add(Dense(units=256, input_dim=784, kernel_initializer='normal', activation='relu'))
# Add output layer
model.add(Dense(units=10, kernel_initializer='normal', activation='softmax'))

# 編譯: 選擇損失函數、優化方法及成效衡量方式
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# 將 training 的 label 進行 one-hot encoding
y_TrainOneHot = np_utils.to_categorical(y_train)
y_TestOneHot = np_utils.to_categorical(y_test)

# 將 training 的 input 資料轉為2維
X_train_2D = X_train.reshape(60000, 28*28).astype('float32')
X_test_2D = X_test.reshape(10000, 28*28).astype('float32')

x_Train_norm = X_train_2D/255
x_Test_norm = X_test_2D/255

# 進行訓練, 訓練過程會存在 train_history 變數中
train_history = model.fit(x=x_Train_norm, y=y_TrainOneHot, validation_split=0.2, epochs=16, batch_size=16, verbose=16)

# 顯示訓練成果(分數)
scores = model.evaluate(x_Test_norm, y_TestOneHot)
print()
print("\t[Info] Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))
```

```
Epoch 1/16
```

```
# 進行訓練, 訓練過程會存在 train_history 變數中
train_history = model.fit(x=x_Train_norm, y=y_TrainOneHot, validation_split=0.2, epochs=16, batch_size=16, verbose=16)

# 顯示訓練成果(分數)
scores = model.evaluate(x_Test_norm, y_TestOneHot)
print()
print("\t[Info] Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))
```

```
Epoch 1/16
Epoch 2/16
Epoch 3/16
Epoch 4/16
Epoch 5/16
Epoch 6/16
Epoch 7/16
Epoch 8/16
Epoch 9/16
Epoch 10/16
Epoch 11/16
Epoch 12/16
Epoch 13/16
Epoch 14/16
Epoch 15/16
Epoch 16/16
313/313 [==============================] - 2s 5ms/step - loss: 0.3704 - accuracy: 0.8873

        [Info] Accuracy of testing data = 88.7%
```

嘗試用少數神經元和多數神經元，還有不同種的方式，如上圖的 onehotencoding，想得知前處理過和前處理較少的資料分別搭配多神經元和少神經元出來的準確度是否相差很大或是億點點而已，結果得出以 onehotencoding 配合多神經元的方法準確率大於原來較少的神經元模型，不管是 MLP 或是 NN 皆如此

5.

```
clf.score(d2_Xtest_dataset,y_test)
```

```
Out[4]: 0.0989
```

```
In [5]:
from sklearn.neural_network import MLPClassifier
nsamples, nx, ny = X_train.shape
d2_train_dataset = X_train.reshape((nsamples,nx*ny))
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(5,5), random_state=1)

clf.fit(d2_train_dataset, y_train)
```

```
Out[5]: MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(5, 5), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=1, shuffle=True, solver='lbfgs',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
In [6]:
nsamples, nx, ny = X_test.shape
d2_Xtest_dataset = X_test.reshape((nsamples,nx*ny))
clf.score(d2_Xtest_dataset,y_test)
```

```
Out[6]: 0.1
```

```
In [7]:
(X_train, y_train), (X_test, y_test)= tf.keras.datasets.fashion_mnist.load_data()
```

```
# 顯示訓練成果(分數)
scores = model.evaluate(x_Test_norm, y_TestOneHot)
print()
print("\t[Info] Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))

Epoch 1/16
Epoch 2/16
Epoch 3/16
Epoch 4/16
Epoch 5/16
Epoch 6/16
Epoch 7/16
Epoch 8/16
Epoch 9/16
Epoch 10/16
Epoch 11/16
Epoch 12/16
Epoch 13/16
Epoch 14/16
Epoch 15/16
Epoch 16/16
313/313 [==============================] - 2s 5ms/step - loss: 0.3704 - accuracy: 0.8873

        [Info] Accuracy of testing data = 88.7%
```