This notebook aims to show result of multilayer perceptron classification

Dataset contains 1000 clothing images \of size 28 pixels × 28 pixels I aim to detect type of clothings

import libraries

In [1]:

```
import pandas as pd
import numpy as np
import math
from sklearn.metrics import confusion_matrix
```

Question 2 :

read files

In [2]:

```
imagesdf = pd.read_csv('hw03_images.csv',header=None)
labeldf = pd.read_csv('hw03_labels.csv',header=None)
```

In [3]:

```
initial_v = pd.read_csv('initial_V.csv',header=None)
initial_w = pd.read_csv('initial_W.csv',header=None)
```

Question 3 :

Divide data into 2 part which are test set and train set

In [4]:

```
train_x = imagesdf.iloc[0:500]
test_x = imagesdf.iloc[-500:]
train_y = labeldf.iloc[0:500]
test_y = labeldf.iloc[-500:]
```

Question 4 :

In [5]:

```
def sigmoid(X):
    return 1/(1+np.exp(-X))
```

Parameters

In [6]:

```
eta = 0.005
epsilon = 1e-3
H = 20 #number of hidden nodes
max_iteration = 500 #max number of iteration
```

z : hidden node features

In [7]:

```
train_x.insert(loc = 0, value = 1,column=784)
```

In [8]:

```
z = sigmoid(train_x.dot(initial_w))
```

In [9]:

```
z.insert(loc = 0, value = 1,column=20)
```

In [10]:

```
z0 = sigmoid(z.dot(initial_v))
```

In [11]:

```
#####z0 is all close to 0.5
```

In [12]:

```
y_head = z0.copy() #initial y_head
```

In [13]:

```
objective_values = -sum(train_y * np.log(y_head) + (1 - train_y) *
np.log(1 - y_head))
```

In [15]:

```
from random import shuffle
random = [[i] for i in range(500)]
shuffle(random)
```

In [16]:

```
W = initial_w.copy()
v = initial_v.copy()
```

In [ ]:

```
iteration = 1
while(1):
    for i in random :
        i = i[0]
        # calculate hidden nodes
        current_X = train_x[i:]


        z.iloc[:,i:] = sigmoid(current_X.iloc[:,1:].dot(W))
        # calculate output node
        current_z = z[i:]
        y_head[i:] = sigmoid(current_z.dot(v))

        delta_v = eta * (train_y.loc[:i,:]- y_head.loc[:i,:] ) * current_z
        delta_W = eta * (train_y.loc[1,:i] - y_head.loc[:i,:]) * current_X.iloc[1,:i].dot((v.iloc[2:(H + 1),1].transpose() )* z.loc[i, 1:H] * (1 - z[i, 1:H]))

        v = v + delta_v
        W = W + delta_W


    z = sigmoid(train_x.insert(loc = 0, value = 1,column=len(train_x)).dot(W))
    y_predicted = sigmoid(z.insert(loc = 0, value = 1,column=len(train_x)).dot(v))
    objective_values = objective_values.append(-sum(train_y * np.log(y_predicted) + (1 - y_head) * np.log(1 - y_predicted)))

    if (abs(objective_values[iteration + 1] - objective_values[iteration]) < epsilon | iteration >= max_iteration) :
        break


    iteration = iteration + 1
```