In [1]:

```
# Kaushal Banthia
# 19CS10039
# Question 10
```

In [ ]:

```
from keras.datasets import mnist
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn.metrics as metrics
```

In [2]:

```
(train_X, train_y), (test_X, test_y) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
ist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

In [3]:

```
print('X_train: ' + str(train_X.shape))
print('Y_train: ' + str(train_y.shape))
print('X_test:  '  + str(test_X.shape))
print('Y_test:  '  + str(test_y.shape))
```

```
X_train: (60000, 28, 28)
Y_train: (60000,)
X_test:  (10000, 28, 28)
Y_test:  (10000,)
```

In [4]:

```
# Normalisation
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')

train_X = train_X/255.0
test_X = test_X/255.0

# Reshaping
train_X = train_X.reshape(len(train_X),-1)
test_X = test_X.reshape(len(test_X),-1)
```

In [5]:

```
NUM_CLASSES = 10
```

In [6]:

```
def train(X_train, y_train, reg=1):
    y_train = one_hot(y_train)
    right = np.zeros((X_train.shape[1], y_train.shape[1]))
    left = np.zeros((X_train.shape[1], X_train.shape[1]))
    for i in range(X_train.shape[0]):
        if i % 1000 == 0:
            print(i)
        right += np.outer(X_train[i], np.transpose(y_train[i]))
        left += np.outer(X_train[i], np.transpose(X_train[i]))
    left = left + reg*np.identity(X_train.shape[1])
    left = np.linalg.inv(left)
    return np.dot(left, right)
```

In [7]:

```
def one_hot(labels_train):
    result = np.zeros((labels_train.shape[0], NUM_CLASSES))
    for i in range(labels_train.shape[0]):
        result[i][labels_train[i]] = 1;
    return result
```

In [8]:

```
def predict(model, X):
    results = np.zeros(X.shape[0])
    matrix = []
    for i in range(X.shape[0]):
        matrix.append(np.dot(np.transpose(model), X[i]))
        results[i] = np.argmax(np.dot(np.transpose(model), X[i]))

    return results, matrix
```

In [15]:

```
X_train = train_X[:,:,np.newaxis]
X_test = test_X[:,:,np.newaxis]
labels_train = train_y
labels_test = test_y

model = train(X_train, train_y)
y_train = one_hot(train_y)
y_test = one_hot(labels_test)

pred_labels_train, matrix_train = predict(model, X_train)
pred_labels_test, matrix_test= predict(model, X_test)


print("Train accuracy: {0}".format(metrics.accuracy_score(labels_train, pred_labels_
print("Test accuracy: {0}".format(metrics.accuracy_score(labels_test, pred_labels_te
```

```
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
25000
26000
27000
28000
29000
30000
31000
32000
33000
```

```
34000
35000
36000
37000
38000
39000
40000
41000
42000
43000
44000
45000
46000
47000
48000
49000
50000
51000
52000
53000
54000
55000
56000
57000
58000
59000
Train accuracy: 0.8519
Test accuracy: 0.8534
```

In [28]:

```
print("10x10 confusion matrix for the train dataset")
cf_train = metrics.confusion_matrix(labels_train, pred_labels_train)
print(cf_train)

error_train = 0
for i in range(10):
  error_train += cf_train[i][i]
error_train /= 60000
error_train = (1-error_train)*100
print("Error rate on the training data = ", error_train, "%")
```

```
10x10 confusion matrix for the train dataset
[[5664    8   20   18   26   43   73    3   62    6]
 [   1 6515   36   17   10   29   15   13  101    5]
 [  91  258 4799  150  102   12  237   85  203   21]
 [  42  142  182 5207   29   91   57  109  140  132]
 [   9  100   54    8 5125   50   48   23   82  343]
 [ 143   70   31  520   82 3788  196   38  400  153]
 [ 105   66   64    2   58   84 5491    0   46    2]
 [  52  184   43   58  151    8    3 5396   20  350]
 [  82  529   61  222  122  233   51   20 4356  175]
 [  66   59   26  113  356    8    4  491   53 4773]]
Error rate on the training data =  14.81 %
```

In [30]:

```
print("10x10 confusion matrix for the test dataset")
cf_test = metrics.confusion_matrix(labels_test, pred_labels_test)
print(cf_test)

error_test = 0
for i in range(10):
  error_test += cf_test[i][i]
error_test /= 10000
error_test = (1-error_test)*100
print("Error rate on the test data = ", error_test, "%")
```

```
10x10 confusion matrix for the test dataset
[[ 942    0    2    2    1    7   15    2    7    2]
```

```
 [   0 1107    2    2    1    1    5    2   15    0]
 [  17   55  809   28   16    0   42   21   39    5]
 [   4   15   26  887    2   14    9   21   22   10]
 [   0   23    6    2  873    5   10    2   13   48]
 [  20   18    2   84   19  625   22   13   67   22]
 [  17    9   10    0   21   20  872    0    9    0]
 [   5   38   18    8   20    0    2  876    3   58]
 [  17   55    9   32   27   41   15   12  743   23]
 [  18   10    2   15   72    1    1   77   13  800]]
Error rate on the test data =  14.659999999999995 %
```