MNIST classiffication of alphabets (1-10) classes using Incremental Extreme Machine learning

In [94]:

```
import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.datasets import mnist
from keras.layers import Dense
from keras.optimizers import Adam
import random
import keras
import keras.utils
from keras.utils.np_utils import to_categorical
from keras import utils as np_utils
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
import time
```

In [95]:

```
(X_train, y_train), (X_test, y_test) = mnist.load_data() #-----------------loading mnist da
```

In [96]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape[0])
```

```
(60000, 28, 28)
(10000, 28, 28)
60000
```

In [97]:

```
y_train = to_categorical(y_train, 10)#-------------------converting values of mnist with 10
y_test = to_categorical(y_test, 10)
```

In [98]:

```
#Each image has Intensity from 0 to 255
X_train = X_train
X_test = X_test
```

In [ ]:

```

```

In [99]:

```
                                                    #we must change the shape of the images to
                                                        #for multipication 1*784
num_pixels = 784
X_train = X_train.reshape(X_train.shape[0],
                          num_pixels)
X_test = X_test.reshape(X_test.shape[0],
                          num_pixels)
print(X_train.shape)
```

```
(60000, 784)
```

In [100]:

```
print(y_train.shape)
predicted_output=np.zeros([1,10])
print(predicted_output)
acc=0
```

```
(60000, 10)
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

In [101]:

```
print(y_train.shape)
predicted_output=np.zeros([1,10])
print(predicted_output)
acc=0
```

```
(60000, 10)
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

In [102]:

```
start = time.time()
l=0
l_max=100 #----------------------------------settting number of neurons
beta=0
error=0
for i in range(l_max):









    if(acc<90):
        weights = np.random.rand(784,i)*100#-----------------------------creating random w
        weights_transpose = np.transpose(weights)
        weights.shape
        h_new = np.dot(X_train,weights)#-------------------------finding value of h
        h_inv = np.linalg.pinv(h_new)#-----------------------------finding pseudo inverse o

        beta = np.dot(h_inv, y_train)#------------------finding beta
    #print(beta.shape)
    #print(h_new.shape)
        predicted_output = np.dot(h_new,beta)#--------------------predicted output
        #error = y_train - predicted_output
        #print(predicted_output)


    #print(predicted_output.shape)

    k=0
    j=0
    for k in range(60000):
        for j in range(10):
            max_item=max(predicted_output[k][:])#-----------------------------changing the
            if(predicted_output[k][j]==max_item):
                predicted_output[k][j]=1

            else:
                predicted_output[k][j]=0

   # print(predicted_output[i][:])
   # print(predicted_output.shape)
        #print(y_train)
    acc = accuracy_score(predicted_output,y_train)#----------------------------FInding acc
    print("Accuracy number", i, "is", + acc*100, "%")
end = time.time()
print("Time elapsed",end - start)
    #print(error)

    #print(h_inv.shape)
    #print(h_new.shape)
    #print(beta)
    #print(weights.shape)
#print(error[1][9])
```

```
Accuracy number 0 is 9.871666666666666 %
Accuracy number 1 is 9.871666666666666 %
Accuracy number 2 is 20.68 %
```

```
Accuracy number 3 is 29.21166666666667 %
Accuracy number 4 is 24.078333333333333 %
Accuracy number 5 is 38.96 %
Accuracy number 6 is 36.21666666666667 %
Accuracy number 7 is 34.43333333333333 %
Accuracy number 8 is 43.53666666666667 %
Accuracy number 9 is 48.513333333333335 %
Accuracy number 10 is 48.141666666666666 %
Accuracy number 11 is 52.06833333333334 %
Accuracy number 12 is 53.803333333333335 %
Accuracy number 13 is 54.50833333333334 %
Accuracy number 14 is 54.635 %
Accuracy number 15 is 53.381666666666675 %
Accuracy number 16 is 59.93333333333334 %
Accuracy number 17 is 58.17 %
Accuracy number 18 is 56.40666666666667 %
Accuracy number 19 is 58.60666666666666 %
Accuracy number 20 is 61.083333333333336 %
Accuracy number 21 is 65.11333333333333 %
Accuracy number 22 is 63.73833333333333 %
Accuracy number 23 is 65.19 %
Accuracy number 24 is 63.915 %
Accuracy number 25 is 67.295 %
Accuracy number 26 is 70.12666666666667 %
Accuracy number 27 is 68.155 %
Accuracy number 28 is 69.65166666666667 %
Accuracy number 29 is 71.21166666666666 %
Accuracy number 30 is 71.83833333333334 %
Accuracy number 31 is 70.88666666666667 %
Accuracy number 32 is 71.26166666666667 %
Accuracy number 33 is 72.34666666666666 %
Accuracy number 34 is 71.285 %
Accuracy number 35 is 70.60499999999999 %
Accuracy number 36 is 72.84666666666666 %
Accuracy number 37 is 73.315 %
Accuracy number 38 is 73.93666666666667 %
Accuracy number 39 is 73.10833333333333 %
Accuracy number 40 is 73.815 %
Accuracy number 41 is 74.22999999999999 %
Accuracy number 42 is 75.94833333333332 %
Accuracy number 43 is 75.79666666666667 %
Accuracy number 44 is 76.175 %
Accuracy number 45 is 75.38666666666667 %
Accuracy number 46 is 75.34166666666667 %
Accuracy number 47 is 75.925 %
Accuracy number 48 is 75.07166666666667 %
Accuracy number 49 is 77.265 %
Accuracy number 50 is 76.60666666666667 %
Accuracy number 51 is 76.35666666666665 %
Accuracy number 52 is 77.55666666666666 %
Accuracy number 53 is 76.83333333333333 %
Accuracy number 54 is 78.22 %
Accuracy number 55 is 78.28 %
Accuracy number 56 is 77.92833333333333 %
Accuracy number 57 is 77.79166666666667 %
Accuracy number 58 is 77.75 %
Accuracy number 59 is 78.74833333333333 %
Accuracy number 60 is 78.96666666666667 %
Accuracy number 61 is 78.89666666666668 %
Accuracy number 62 is 78.51666666666667 %
Accuracy number 63 is 78.84333333333333 %
```

```
Accuracy number 64 is 79.42166666666667 %
Accuracy number 65 is 80.05833333333334 %
Accuracy number 66 is 78.72666666666667 %
Accuracy number 67 is 79.68499999999999 %
Accuracy number 68 is 79.38833333333334 %
Accuracy number 69 is 79.29666666666667 %
Accuracy number 70 is 79.92166666666667 %
Accuracy number 71 is 80.24833333333333 %
Accuracy number 72 is 80.89666666666666 %
Accuracy number 73 is 80.76 %
Accuracy number 74 is 80.46 %
Accuracy number 75 is 80.21166666666667 %
Accuracy number 76 is 80.42666666666666 %
Accuracy number 77 is 79.98666666666666 %
Accuracy number 78 is 80.80666666666667 %
Accuracy number 79 is 81.12333333333333 %
Accuracy number 80 is 80.74333333333334 %
Accuracy number 81 is 81.545 %
Accuracy number 82 is 81.19333333333333 %
Accuracy number 83 is 81.23666666666666 %
Accuracy number 84 is 81.37166666666667 %
Accuracy number 85 is 81.39333333333333 %
Accuracy number 86 is 81.20333333333333 %
Accuracy number 87 is 81.525 %
Accuracy number 88 is 81.285 %
Accuracy number 89 is 81.83833333333334 %
Accuracy number 90 is 81.63333333333334 %
Accuracy number 91 is 81.88333333333333 %
Accuracy number 92 is 81.96333333333334 %
Accuracy number 93 is 81.58333333333333 %
Accuracy number 94 is 82.05333333333333 %
Accuracy number 95 is 81.76666666666667 %
Accuracy number 96 is 82.465 %
Accuracy number 97 is 82.08666666666666 %
Accuracy number 98 is 81.74666666666667 %
Accuracy number 99 is 82.35333333333334 %
Time elapsed 369.9198372364044
```

In [ ]:

```
```

In [77]:

```
#print(error[1])
#print(predicted_output[1])
#print(y_train.size)
#print(beta)
```

In [78]:

```
#h_new = np.dot(X_train,weights)
#h_new.shape
```

In [79]:

```
#h_reshaped = np.reshape(h, (l, 784))
#h_inv = np.linalg.pinv(h_new)
#print(h_inv.shape)
```

In [80]:

```
#beta = np.dot(h_inv, y_train)
```

In [81]:

```
#print(beta)
```

In [82]:

```
#predicted_output= np.dot(h_new, beta) #-------------------------------------------------
#print(predicted_output.shape)
```

In [83]:

```
#[1][:])
```

In [84]:

```
#acc = accuracy_score(error,y_train)#----------------------------FInding accuracy
#print("Accuracy is", + acc*100, "%")
```

In [ ]:

```

```

In [ ]:

```

```