Double-click (or enter) to edit

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_

Enter your authorization code:
··········
Mounted at /content/drive
```

loading the data files: mnist_data into data which is X and mnist_labels into labels which is y

```
data=pd.read_csv("/content/drive/My Drive/mnist_hw2_ml/mnist_data.txt",sep=" ",heac
labels=pd.read_csv("/content/drive/My Drive/mnist_hw2_ml/mnist_labels.txt",header=N
```

inserting bias term as 1 initially, as last column into data

```
data.insert(loc=784, column=784,value=1,allow_duplicates=True)
```

spiting the "data" and "labels" using sklearn split method in 50:50 ratio and storing for training and
X_test,y_test

```
X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.5)
```

creating and initialising weights dataframe with 0's of size 10x785

```
weights=pd.DataFrame(index=np.arange(10),columns=np.arange(785))
weights.fillna(value=0.0, inplace = True)
```

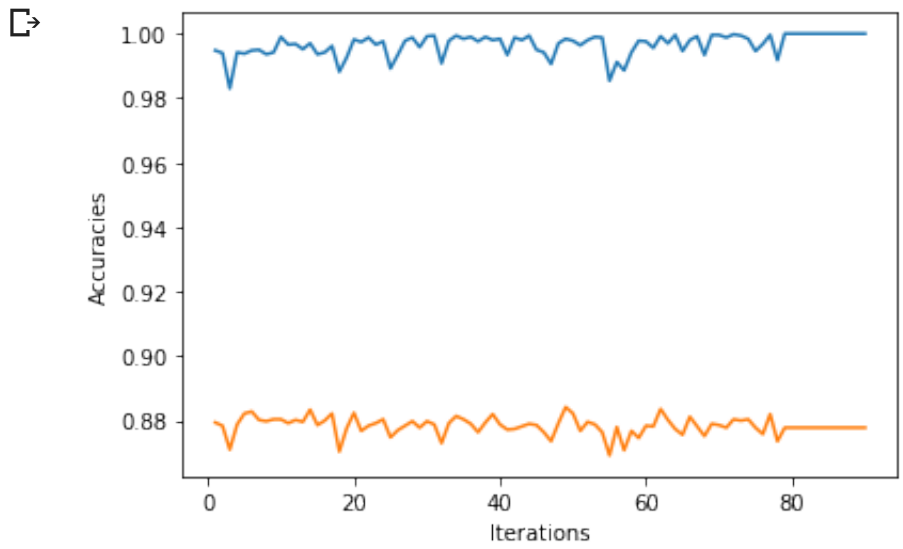
the perceptron algorithm

```
iterations =[] #lists for storing iterations, accuracy on train and test, which lat
accuracy_on_train=[]
accuracy_on_test=[]
#epochs 120- traning our weights on same data 120 times to increase the
for num_iterations in range(1,120):
  iterations.append(num_iterations)
  print(num_iterations)
  for i in X_train.index:
    k=np.argmax(weights.dot(X_train.loc[i,:]))
    if k != y_train.loc[i,0]:
      weights.loc[k,:] = weights.loc[k,:] - X_train.loc[i,:]
      weights.loc[y_train.loc[i,0],:]=weights.loc[y_train.loc[i,0],:]+X_train.loc[i
```

```
  y_predicted=pd.DataFrame(index=list(y_test.index),columns=np.arange(1))
  y_predicted.fillna(value=0, inplace = True)

  y_train_predicted=pd.DataFrame(index=list(y_train.index),columns=np.arange(1))
  y_train_predicted.fillna(value=0, inplace = True)

  for idx in X_train.index:
    y_train_predicted.loc[idx,0]=np.argmax(weights.dot(X_train.loc[idx,:]))
  train_accuracy=accuracy_score(y_train,y_train_predicted)
  accuracy_on_train.append(train_accuracy)
  print(train_accuracy)


  for idx in X_test.index:
    y_predicted.loc[idx,0]=np.argmax(weights.dot(X_test.loc[idx,:]))
  test_accuracy=accuracy_score(y_test,y_predicted)
  accuracy_on_test.append(test_accuracy)
  print(test_accuracy)
```

Plotting Accuracies on train and test sets for different epochs

```
plt.plot(iterations[:90], accuracy_on_train[:90], iterations[:90], accuracy_on_test
plt.xlabel("Iterations")
plt.ylabel("Accuracies")
#plt.text(65,0.025,"on_train")
plt.show()
```



The upper curve is for train and lower is obtained on test data

Conclusion: The accuracy of the classifier on the training set is: 0.9996 The accuracy of the classif

At 77 iterations the train accuracy is 0.9996 and accuracy on test is 0.882 if i increase the epoch th it is decreasing. So using converging.

ref:https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.htm