In [8]:
```
import init
import mnistdata as mylib
import numpy as np
from matplotlib import pyplot
import matplotlib as mpl
import matplotlib.pyplot as plt
import math as math
import preprocessing as pp
import knnfeatures as kf
import datetime


def show(image,image2='None'):
    fig = pyplot.figure()
    ax = fig.add_subplot(1,1,1)
    imgplot = ax.imshow(image,cmap=mpl.cm.Greys)
    imgplot.set_interpolation('nearest')
    ax.xaxis.set_ticks_position('top')
    ax.yaxis.set_ticks_position('left')
    if(image2!='None'):
        ax = fig.add_subplot(2,2,1)
```

jupyter knn Last Checkpoint: Last Tuesday at 12:39 AM (unsaved changes)
File Edit View Insert Cell Kernel Help Python 3

```
def euclideanDistance(x1, x2):
    dis = sum(pow(x1-x2,2))
    return math.sqrt(dis)

def oneNormDistance(x1, x2):
    dis = sum(abs(x1-x2))
    return dis

def infNormDistance(x1,x2):
    dis = np.zeros(x1.size)
    for i in range(x1.size):
        dis[i] = max(x1[i],x2[i])
    return(sum(dis))

def mahalonobisDist(x1,x2,invS):
    diff = np.matrix(x1-x2)
    print(diff)
    dis = (diff*invS)*diff.T
    return dis

def getFeatureVector(Image):
    #pre-processed Image
    ppImage = pp.boundarysquare(pp.thresholding(Image,0))
```

jupyter knn Last Checkpoint: Last Tuesday at 12:39 AM (unsaved changes)
File Edit View Insert Cell Kernel Help Python 3

```
    #feature 2,3: Number of Left/Right pixels
    lrmass = kf.averagelrmass(ppImage)
    featureVector.append(lrmass[0])
    featureVector.append(lrmass[1])
    #feature 4,5: Number of Top/Bottom pixels
    tbmass = kf.averagetbmass(ppImage)
    featureVector.append(tbmass[0])
    featureVector.append(tbmass[1])
    #feature 6: avgHorizontalStroke
    featureVector.append(kf.avgHorizontalStroke(ppImage))
    #feature 7: avgVerticalStroke
    featureVector.append(kf.avgVerticalStroke(ppImage))
    featureVector.append(kf.transitions(ppImage))
    featureVector.append(kf.topBoundaryTouch(ppImage))
    featureVector.append(kf.bottomBoundaryTouch(ppImage))
    featureVector.append(kf.leftBoundaryTouch(ppImage))
    featureVector.append(kf.rightBoundaryTouch(ppImage))
    featureVector.append(kf.aspectRatio(ppImage))
    featureVector.append(kf.avgDistanceFromImageCenter(ppImage))
    featureVector.append(kf.ySymmetric(ppImage))
    #featureVector.append(kf.xSymmetric(ppImage))

    #featureVector.append(kf.lrRatio(ppImage))
    #featureVector.append(kf.tbRatio(ppImage))
```

jupyter knn Last Checkpoint: Last Tuesday at 12:39 AM (unsaved changes)
File Edit View Insert Cell Kernel Help Python 3

```
    while((i<images.shape[0])and(i<N)):
        dataFeatures.append(getFeatureVector(images[i]))
        i+=1
    return np.array(dataFeatures)


def getNeighbors(X, Z, xt, k, dist=euclideanDistance):
    distances = []
    for x1 in X:
        distance = dist(x1,xt)
        distances.append(distance)
    sortedDistances = distances[:]
    sortedDistances.sort()
    neighbors = []
    for d in sortedDistances:
        if(len(neighbors)<k):
            for index in range(len(distances)):
                if(d == distances[index] and len(neighbors)<k):
                    neighbor = (X[index],Z[index],d)
                    neighbors.append(neighbor)
    return neighbors

def getResponse(neighbors, c=10):
    response = -1
```

jupyter knn Last Checkpoint: Last Tuesday at 12:39 AM (unsaved changes)
File Edit View Insert Cell Kernel Help Python 3

```
def getAccuracy(YT, ZT):
    success = 0
    size = len(YT)
    for i in range(size):
        if(YT[i] == ZT[i]):
            success+=1
    accuracy = (success/float(size))*100
    print(success)
    return accuracy

def predict(X, Z, XT, k):
    Y=[]
    i = 0
    for xt in XT:
```

jupyter knn Last Checkpoint: Last Tuesday at 12:39 AM (unsaved changes)
File Edit View Insert Cell Kernel Help Python 3

```
print("start : "+ str(datetime.datetime.now()))
Ntr = 60000
Nte = 10000

lbl_tr, img_tr = mylib.read("training","../data")
dataFeatures_train = getAllFeatureVectors(img_tr,Ntr)

lbl_te, img_te = mylib.read("testing","../data")
dataFeatures_test = getAllFeatureVectors(img_te,Nte)

k = 4
YT = predict(dataFeatures_train[0:Ntr], lbl_tr[0:Ntr], dataFeatures_test[0:Nte], k)
accuracy = getAccuracy(YT, lbl_te[0:Nte])
#print(np.array(YT))
#print(lbl_te[0:Nte])
print('Accuracy: ' + repr(accuracy))
print("end : "+ str(datetime.datetime.now()))
```

```
start : 2016-12-29 00:00:58.827461
7739
Accuracy: 77.39
end : 2016-12-29 00:49:31.586612
```

In [ ]: