ECE175 hw4

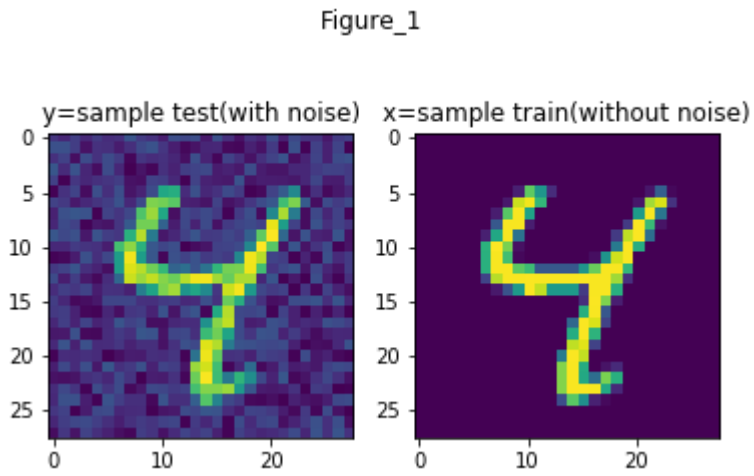
In [1]:
```
from scipy.io import loadmat
import numpy as np
data=loadmat('data.mat')
imageTest=data['imageTestNew']
imageTrain=data['imageTrain']
labelTest=data['labelTestNew']
labelTrain=data['labelTrain']
```

In [3]:
```
# importing matplotlib modules
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

# Read Images
sampletest = mpimg.imread('sampletest.png')
sampletrain = mpimg.imread('sampletrain.png')

# Output Images
fig, axs = plt.subplots(1,2)
fig.suptitle('Figure_1')
axs[0].set_title('y=sample test(with noise)')
axs[0].imshow(sampletest)
axs[1].set_title('x=sample train(without noise)')
axs[1].imshow(sampletrain)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x23d3e3f6b08>
```



Part 1

In [4]:
```
import numpy as np
def reshape_image(input):
    '''
    reshape the image matrix to n dimensional vector
    '''
    size=input.shape
    height,width=size[0],size[1]
    output=np.reshape(input,(height*width,1))
    return output
```

MLE of parameter 'a' for unnormalized image

In [5]:
```
#reshape sample train, x and sample test, y
x=reshape_image(sampletrain)
y=reshape_image(sampletest)
#obtain the transpose of x
x_t=np.transpose(x)
#obtain scale a
a=float(np.dot(x_t,y)/np.dot(x_t,x))
print("The MLE of scale parameter 'a' is",a)
```

```
The MLE of scale parameter 'a' is 0.6796183586120605
```

Part 2

In [6]:
```
import numpy as np
def least_square_distance(test,train,a=1):
    '''
    calculate the least square distance between testing image and training image
    '''
    test=reshape_image(test)
    train=reshape_image(train)
    dif=test-a*train
    distance=float(np.dot(dif.transpose(),dif))
    return distance
```

In [7]:
```
def obtain_predicted_label(imageTest,imageTrain):
    num_test=imageTest.shape[2]
    num_train=imageTrain.shape[2]
    predicted_test_label=np.zeros([num_test,1])
    for i in range(num_test):
        distance_list=np.zeros([num_train,1])
        test=reshape_image(imageTest[:,:,i])
        for j in range(num_train):
            train=reshape_image(imageTrain[:,:,j])
            distance=least_square_distance(test,train,a)
            distance_list[j]=distance
        arg_min=np.argmin(distance_list)
        predicted_test_label[i]=int(labelTrain[arg_min])
    return predicted_test_label
```

In [65]:
```
def obtain_stats(labelTest,predicted_test_label):
    stats=np.zeros([10,4])
    for i in range(10):
        stats[i][0]=i
        stats[i][1]=list(labelTest).count(i)

    for j in range(500):
        if labelTest[j,0]!=predicted_test_label[j,0]:
            index=labelTest[j,0]
            stats[index][2]=stats[index][2]+1
    stats[:,3]=np.round(stats[:,2]/stats[:,1],3)
    return stats
```

In [10]:
```
predicted_test_label_1=obtain_predicted_label(imageTest,imageTrain)
```

In [66]:
```
stats_1=obtain_stats(labelTest,predicted_test_label_1)
```

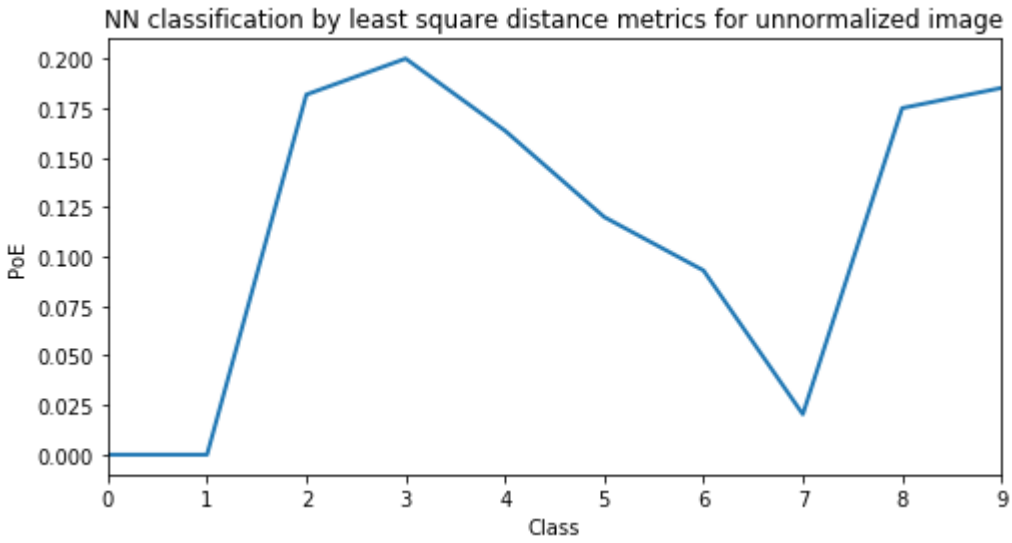
predicted test label by using least square distance metrics

In [125]:
```
from tabulate import tabulate
print(tabulate(stats_1,headers=["Class","Total Samples","Errors","Error Rate"],t
```

Class	Total Samples	Errors	Error Rate
0	42	0	0
1	67	0	0
2	55	10	0.182
3	45	9	0.2
4	55	9	0.164
5	50	6	0.12
6	43	4	0.093
7	49	1	0.02
8	40	7	0.175
9	54	10	0.185

In [113]:
```
plt.figure(figsize=(8,4))
plt.xlim(0,9)
plt.title('NN classification by least square distance metrics for unnormalized ir
plt.ylabel('PoE')
plt.xlabel('Class')
plt.plot(stats_1[:,0],stats_1[:,2]/stats_1[:,1],linewidth=2.0)
```

Out[113]:
```
[<matplotlib.lines.Line2D at 0x23d3fa2fec8>]
```



In [116]:
```
#total error rate
total_error_rate=sum(list(stats_1[:,2]))/500
print("the total error rate is",total_error_rate)
```

```
the total error rate is 0.112
```

Part 3.

In [117]:
```
def normalize(input):
    minimum=np.min(input)
    maximum=np.max(input)
    interval=maximum-minimum
    output=(input-minimum)/interval
    return output
```

MLE of parameter 'a' after normalization

In [118]:
```
x=reshape_image(sampletrain)
x=normalize(x)
y=reshape_image(sampletest)
y=normalize(y)
x_t=np.transpose(x)
a=float(np.dot(x_t,y)/np.dot(x_t,x))
print("The MLE of scale parameter 'a' is",a)
```

```
The MLE of scale parameter 'a' is 0.9218227863311768
```

In [119]:
```
def obtain_predicted_label_normalized(imageTest,imageTrain):
    num_test=imageTest.shape[2]
    num_train=imageTrain.shape[2]
    predicted_test_label=np.zeros([num_test,1])
    for i in range(num_test):
        distance_list=np.zeros([num_train,1])
        test=reshape_image(imageTest[:,:,i])
        test=normalize(test)
        for j in range(num_train):
            train=reshape_image(imageTrain[:,:,j])
            train=normalize(train)
            distance=least_square_distance(test,train)
            distance_list[j]=distance
        arg_min=np.argmin(distance_list)
        predicted_test_label[i]=int(labelTrain[arg_min])
    return predicted_test_label
```

In [78]:
```
predicted_test_label_2=obtain_predicted_label_normalized(imageTest,imageTrain)
```

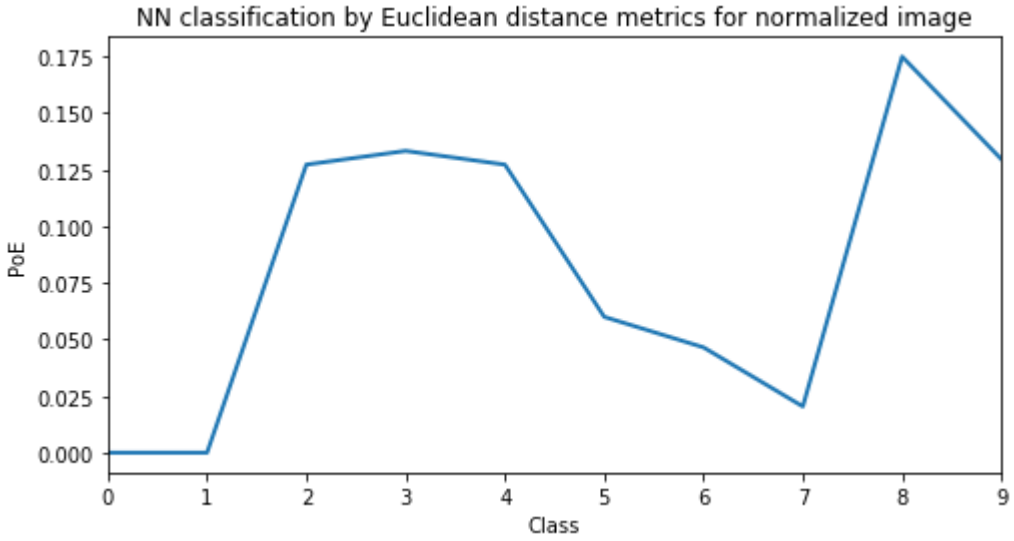
In [120]:
```
stats_2=obtain_stats(labelTest,predicted_test_label_2)
```

In [108]:
```
print(tabulate(stats_2,headers=["Class","Total Samples","Errors","Error Rate"],t
```

Class	Total Samples	Errors	Error Rate
0	42	0	0
1	67	0	0
2	55	7	0.127
3	45	6	0.133
4	55	7	0.127
5	50	3	0.06
6	43	2	0.047
7	49	1	0.02
8	40	7	0.175
9	54	7	0.13

In [121]:
```
plt.figure(figsize=(8,4))
plt.xlim(0,9)
plt.title('NN classification by Euclidean distance metrics for normalized image'
plt.ylabel('PoE')
plt.xlabel('Class')
plt.plot(stats_2[:,0],stats_2[:,2]/stats_2[:,1],linewidth=2.0)
```

Out[121]:
```
[<matplotlib.lines.Line2D at 0x23d3fb70c88>]
```



In [122]:
```
#total error rate
total_error_rate=sum(list(stats_2[:,2]))/500
print("the total error rate is",total_error_rate)
```

```
the total error rate is 0.08
```

In [ ]:
```

```