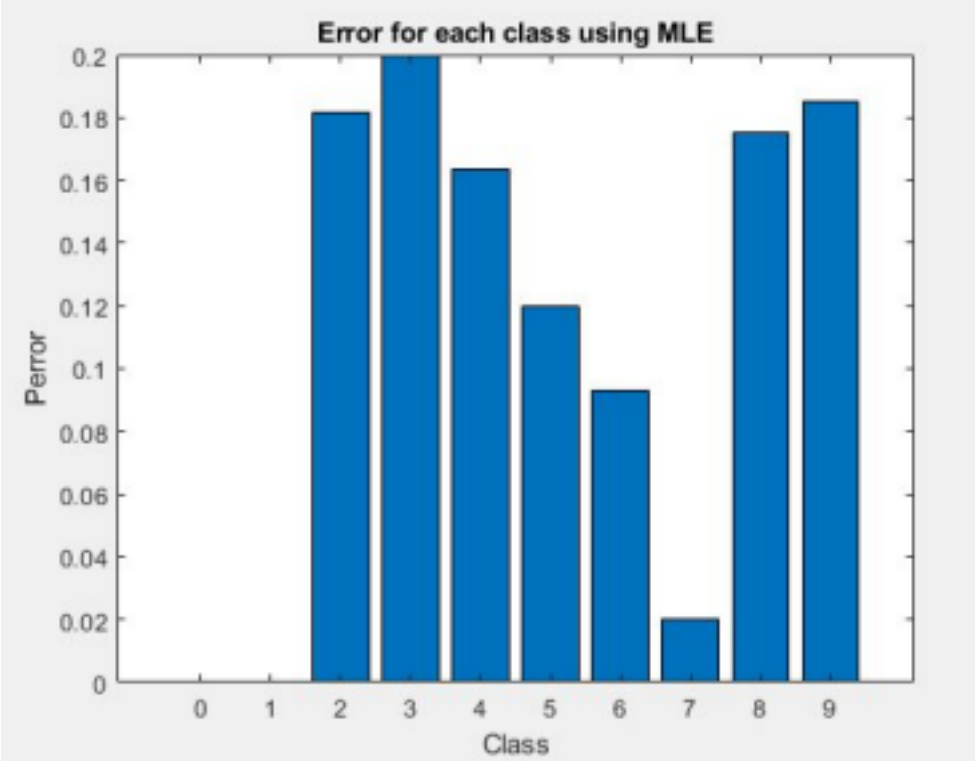
I used the code below to get a* using the least squares solving formula.

```
test = double(imread('sampletest.png'));
train = double(imread('sampletrain.png'));
y=test(find(train~=0));
gamma=train(find(train~=0));
a=(inv(gamma.'*gamma))*gamma.'*y;
```

The total error rate I got was 0.1120 and the errors for each class are: 0
0
0.181818181818182
0.200000000000000
0.163636363636364
0.120000000000000
0.0930232558139535
0.0204081632653061
0.175000000000000
0.185185185185185
With the top being for class 0 and below that class 1 and so on until class 9.



The total error rate I got was 0.2120 and the errors for each class are:

0.0952380952380952
0
0.327272727272727
0.333333333333333
0.181818181818182
0.260000000000000
0.255813953488372
0.0816326530612245
0.425000000000000
0.259259259259259

With the top being for class 0 and below that class 1 and so on until class 9. The error rate without using the MLE least squares and a* are much higher than using the least squares distance. Other than class 1 each one of the classes for no MLE has a higher error rate than the least squares model. Class 1 for both have 0 error which probably shows that the NN algorithm for classifying class 1 is very robust to gaussian noise. However class 8 has the highest error difference between euclidean distance and least squares classifiers, suggesting that class 8 images are the least robust to gaussian noise when it comes to NN classification.