Project1:Density Estimation and Classification

Parameters estimated

▼ **Digit 0**

(No.1) Mean of feature1 for digit0 : 44.2661431122

(No.2) Variance of feature1 for digit0 : 117.247069137

(No.3) Mean of feature2 for digit0 : 87.4577371002

(No.4) Variance of feature2 for digit0 : 103.340126397

▼ **Digit 1**

(No.5) Mean of feature1 for digit1 : 19.396907398

(No.6) Variance of feature1 for digit1 : 31.4728832398

(No.7) Mean of feature2 for digit1 : 61.4018410989

(No.8) Variance of feature2 for digit1 : 83.1622162002

Accuracy

trainset for digit 0 : 0.9173469387755102

trainset for digit 1 : 0.9233480176211454

Observation and Analysis

I think as we are using just those two specific features for more digits the high accuracy would be significantly less.

Given that the two features in no way correlate with the shape/geometry of the digits ,what makes this work. I suspect it is the fact that the "surface area" and "perimeter" of the two digits chosen are significantly different. There is a larger surface area increases the mean brightness and a longer perimeter increases the standard deviation because more variation occurs along the edges of the digits. Comparing the numbers for mean brightness and mean variance for both digits support this hypothesis. A more empirical (trial and error) approach would just calculate the four parameter set for more digits and just see how well separated they are; in the case of digits 0 and 1 the means of all both features are separated by more than one standard deviation in all cases so accurate separation based on these features was easily calculatable with reasonable effort.

Code Block

```
import numpy
import scipy.io
import math
import geneNewData
def main():
    myID='6477'
    geneNewData.geneData(myID)
    Numpyfile0 = scipy.io.loadmat('digit0_stu_train'+myID+'.mat')
    Numpyfile1 = scipy.io.loadmat('digit1_stu_train'+myID+'.mat')
    Numpyfile2 = scipy.io.loadmat('digit0_testset'+'.mat')
    Numpyfile3 = scipy.io.loadmat('digit1_testset'+'.mat')
    train0 = Numpyfile0.get('target_img')
    train1 = Numpyfile1.get('target_img')
    test0 = Numpyfile2.get('target_img')
    test1 = Numpyfile3.get('target_img')
    a0=[]
    m0=[]
    a1=[]
    m1=[]
    t0a=[]
    t0s=[]
    t1a=[]
    t1s=[]
    res0=[]
    res1=[]
    ac0=0
    at1=0
    ac1=0
    at0=0

    print([len(train0),len(train1),len(test0),len(test1)])
    print('Your trainset and testset are generated successfully!')

    for c in train0:
        a0.append(numpy.average(c))
        m0.append(numpy.std(c))
    for e in train1:
        a1.append(numpy.average(e))
        m1.append(numpy.std(e))
    for g in test0:
        t0a.append(numpy.average(g))
        t0s.append(numpy.std(g))
    for i in test1:
        t1a.append(numpy.average(i))
        t1s.append(numpy.std(i))

    xa0 = numpy.mean(a0)
    va0=numpy.var(a0)
    xm0 = numpy.mean(m0)
    vm0=numpy.var(m0)
    xa1 = numpy.mean(a1)
    va1=numpy.var(a1)
    xm1 = numpy.mean(m1)
    vm1=numpy.var(m1)

    print('(No.1) Mean of feature1 for digit0','(No.2) Variance of
feature1 for digit0','(No.3) Mean of feature2 for digit0','(No.4)
Variance of feature2 for digit0')
    print(xa0,va0,xm0,vm0)
    print('(No.5) Mean of feature1 for digit1','(No.6) Variance of
feature1 for digit1','(No.7) Mean of feature2 for digit1','(No.8) Variance
of feature2 for digit1')
    print(xa1,va1,xm1,vm1)


    for (m,n) in zip(t0a,t0s):
        p0 = (((1/(math.sqrt(2*(22/7)*va0)))*math.exp(-((m-
xa0)**2)/(2*va0)))*((1/(math.sqrt(2*(22/7)*vm0)))*math.exp((-((n-
xm0)**2)/(2*vm0)))))
        p1 = (((1/(math.sqrt(2*(22/7)*va1)))*math.exp(-((m-
xa1)**2)/(2*va1)))*((1/(math.sqrt(2*(22/7)*vm1)))*math.exp((-((n-
xm1)**2)/(2*vm1)))))
        if(p0 > p1):
            res0.append('0')
    for (s,r) in zip(t1a,t1s):
        ps0 = 0.5*(((1/(math.sqrt(2*(22/7)*va0)))*math.exp(-((s-
xa0)**2)/(2*va0)))*((1/(math.sqrt(2*(22/7)*vm0)))*math.exp((-((r-
xm0)**2)/(2*vm0)))))

        ps1 = 0.5*(((1/(math.sqrt(2*(22/7)*va1)))*math.exp(-((s-
xa1)**2)/(2*va1)))*((1/(math.sqrt(2*(22/7)*vm1)))*math.exp((-((r-
xm1)**2)/(2*vm1)))))
        if(ps1 > ps0):
            res1.append('1')

    print('Accuracy of digit0')
    print(len(res0)/len(t0a))
    print('Accuracy of digit1')
    print(len(res1)/len(t1a))

    pass
if __name__ == '__main__':
    main()
```