Bayes Classifier

Code:

```
import glob, os
from PIL import Image
import numpy as np
import PIL
from sklearn.ensemble import RandomForestClassifier as rfc
from matplotlib import pyplot

features = []

# loading training images from folder and converting them to matrix (1x1024) after downscaling to 32x32

for file in os.listdir("TrainCharacters/1"):
    try:
        img = np.asarray(Image.open("TrainCharacters/1/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()
        features.append(img)
    except Exception:
        pass
for file in os.listdir("TrainCharacters/2"):
    try:
        img = np.asarray(Image.open("TrainCharacters/2/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()
        features.append(img)
    except Exception:
        pass
for file in os.listdir("TrainCharacters/2"):
    try:
        img = np.asarray(Image.open("TrainCharacters/3/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()
        features.append(img)
    except Exception:
        pass

target = []
for x in range(200):
    target.append(1)
for x in range(200):
    target.append(2)
for x in range(200):
    target.append(3)
```

```
#normalising feature vector
features = np.matrix(features)/255

#claculation of mean vectors from maximum likelihood
mu1, mu2, mu3 = np.matrix(np.sum(features[0:200], axis = 0)/200), np.matrix(np.sum(features[200:400],
axis = 0)/200), np.matrix(np.sum(features[400:600], axis = 0)/200)

# calculation of covariance matrices from maximum likelihood
cov1, cov2, cov3 = np.zeros((1024,1024)), np.zeros((1024,1024)), np.zeros((1024,1024))

for x in range(200):
   cov1 = cov1 + (features[x] - mu1).transpose()*(features[x] - mu1)
for x in range(200,400):
   cov2 = cov2 + (features[x] - mu2).transpose()*(features[x] - mu2)
for x in range(400,600):
   cov3 = cov3 + (features[x] - mu3).transpose()*(features[x] - mu3)

cov1 = cov1/200 + 0.6*np.identity(1024)
cov2 = cov2/200 + 0.6*np.identity(1024)
cov3 = cov3/200 + 0.6*np.identity(1024)

# beforehand calculation of inverse and determinant of covariance matrices of all classes
i1, i2, i3 = np.linalg.inv(cov1), np.linalg.inv(cov2), np.linalg.inv(cov3)
det1 ,det2, det3= np.linalg.det(cov1), np.linalg.det(cov2), np.linalg.det(cov3)

# probability function gives value of p(w(i)/mu, x, cov)
def prob(mean, x, i, dcr):
   a = np.sqrt(dcr)*np.pi**40
   b = float(np.exp(-0.5*(x-mean)*(i)*(x-mean).transpose()))
   return float(b/a)

score = [0,0,0]

class1 = np.zeros((100,1))
class2 = np.zeros((100,1))

# calculation of accuracy of classification model on testing data
k = 0

for file in os.listdir("TestCharacters/1"):
   g = file.split('.')
   if g[1] == 'jpg':
      img = np.asmatrix(Image.open("TestCharacters/1/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
      p1, p2, p3 = prob(mu1, img, i1, det1), prob(mu2, img, i2, det2), prob(mu3, img, i3, det3)
      p = [p1,p2,p3]
      if np.argmax(p)==0:
         score[0] = score[0] + 1
```

```
        else:
            class1[k] = class1[k] + 1
        k = k+1
k = 0
for file in os.listdir("TestCharacters/2"):
    g = file.split('.')
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/2/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i1, det1), prob(mu2, img, i2, det2), prob(mu3, img, i3, det3)
        p = [p1,p2,p3]
        if np.argmax(p)==1:
            score[1] = score[1] + 1
        else:
            class2[k] = class2[k] + 1
        k = k+1
k = 0
for file in os.listdir("TestCharacters/3"):
    g = file.split('.')
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/3/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i1, det1), prob(mu2, img, i2, det2), prob(mu3, img, i3, det3)
        p = [p1,p2,p3]
        if np.argmax(p)==2:
            score[2] = score[2] + 1
print("Part 1")
print('Class1: ' + str(score[0]) + '%' + '\nClass2: ' + str(score[1]) + '%' + '\nClass3: ' + str(score[2]) + '%')

score = [0,0,0]

feat = features.transpose()
cov0 = np.identity(1024)
count = 0

# building covariance matrix by pooling all the data together
for i in range(1024):
    cov0[i][count] = np.var(feat[i]) + 0.6
    count = count + 1

#calculating determinant and inverse
det0 = np.linalg.det(cov0)
i0 = np.linalg.inv(cov0)

# calculation of accuracy of classification model on testing data
k = 0
for file in os.listdir("TestCharacters/1"):
    g = file.split('.')
```

```
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/1/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i0, det0), prob(mu2, img, i0, det0), prob(mu3, img, i0, det0)
        p = [p1,p2,p3]
        if np.argmax(p)==0:
            score[0] = score[0] + 1
        else:
            class1[k] = class1[k] + 1
        k = k+1
k = 0
for file in os.listdir("TestCharacters/2"):
    g = file.split('.')
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/2/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i0, det0), prob(mu2, img, i0, det0), prob(mu3, img, i0, det0)
        p = [p1,p2,p3]
        if np.argmax(p)==1:
            score[1] = score[1] + 1
        else:
            class2[k] = class2[k] + 1
        k = k+1
for file in os.listdir("TestCharacters/3"):
    g = file.split('.')
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/3/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i0, det0), prob(mu2, img, i0, det0), prob(mu3, img, i0, det0)
        p = [p1,p2,p3]
        if np.argmax(p)==2:
            score[2] = score[2] + 1
print("Part 2")
print('Class1: ' + str(score[0]) + '%' + '\nClass2: ' + str(score[1]) + '%' + '\nClass3: ' + str(score[2]) + '%')

score = [0,0,0]

# using identity matrix as covariance matrix
cov01 = np.identity(1024)
det01 = np.linalg.det(cov01)
i01 = np.linalg.inv(cov01)

# calculation of accuracy of classification model on testing data
k = 0
for file in os.listdir("TestCharacters/1"):
    g = file.split('.')
    if g[1] == 'jpg':
```

```
        img = np.asmatrix(Image.open("TestCharacters/1/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i01, det01), prob(mu2, img, i01, det01), prob(mu3, img, i01, det01)
        p = [p1,p2,p3]
        if np.argmax(p)==0:
            score[0] = score[0] + 1
        else:
            class1[k] = class1[k] + 1
        k = k+1
k = 0
for file in os.listdir("TestCharacters/2"):
    g = file.split('.')
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/2/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i01, det01), prob(mu2, img, i01, det01), prob(mu3, img, i01, det01)
        p = [p1,p2,p3]
        if np.argmax(p)==1:
            score[1] = score[1] + 1
        else:
            class2[k] = class2[k] + 1
        k = k+1
for file in os.listdir("TestCharacters/3"):
    g = file.split('.')
    if g[1] == 'jpg':
        img = np.asmatrix(Image.open("TestCharacters/3/" + file).convert('L').resize((32,32),
Image.ANTIALIAS)).flatten()/255
        p1, p2, p3 = prob(mu1, img, i01, det01), prob(mu2, img, i01, det01), prob(mu3, img, i01, det01)
        p = [p1,p2,p3]
        if np.argmax(p)==2:
            score[2] = score[2] + 1
print("Part 3")
print('Class1: ' + str(score[0]) + '%' + '\nClass2: ' + str(score[1]) + '%' + '\nClass3: ' + str(score[2]) + '%')

for i in range(100):
    if class1[i] == 3:
        img = np.asmatrix(Image.open("TestCharacters/1/" + str(201 + i) + '.jpg').convert('L').resize((32,32),
Image.ANTIALIAS))
        pyplot.imshow(img, pyplot.cm.gray)
        pyplot.show()
for i in range(100):
    if class2[i] == 3:
        img = np.asmatrix(Image.open("TestCharacters/1/" + str(201 + i) + '.jpg').convert('L').resize((32,32),
Image.ANTIALIAS))
        pyplot.imshow(img, pyplot.cm.gray)
        pyplot.show()
```

Results:

Classifier 1	Classifier 2	Classifier 3
Class1 : 0.85	Class1 : 0.86	Class1 : 0.87
Class2 : 0.93	Class2 : 0.86	Class2 : 0.86
Class3 : 1	Class3 : 1	Class3 : 1
Average : 0.9266	Average : 0.9066	Average : 0.91

Misclassified Samples (true class : 1, misclassified as class 2)