7th Homework

Thomas Saltos

Exercise 1

We have that $P(\omega_1) = P(\omega_2)$. In order to determine the R_1 and R_2 we should solve the following equation:

$$P(\omega_1)P(x|\omega_1) = P(\omega_2)P(x|\omega_2) \Rightarrow \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} = \frac{1}{\sqrt{2\pi}\sqrt{5}}e^{-\frac{x^2}{10}} \Rightarrow -\frac{x^2}{2} = -\frac{1}{2}\ln 5 - \frac{x^2}{10} \Rightarrow \frac{4x^2}{5} = \ln 5$$

$$\Rightarrow x^2 = \frac{5}{4}\ln 5$$

This defines the regions R_1 and R_2:

$$R_1: \{x: -\sqrt{\frac{5}{4}\ln 5} < x < \sqrt{\frac{5}{4}\ln 5}\}$$

$$R_2: \{x: x < -\sqrt{\frac{5}{4}\ln 5} \cup x > \sqrt{\frac{5}{4}\ln 5}\}$$

Exercise 2

a) According to the Bayes classifier the pdf that models the data in class ω_i is:

$$\mathrm{p}(x|\omega_i) = \frac{1}{(2\pi)^{l/2}|\Sigma_i|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right)$$

Let $g_i(x) = ln\big(p(x|\omega_i)P(\omega_i)\big) = -\frac{1}{2}x^T\Sigma_i^{-1}x - \frac{1}{2}\mu_i\Sigma_i^{-1}\mu_i + c_i + lnP(\omega_i)$
with $c_i = -\frac{1}{2}ln2\pi - \frac{1}{2}ln|\Sigma_i|$

We have two equiprobable classes ω_1 and ω_2. We define $g(x) = g_1(x) - g_2(x)$ and we know that $\Sigma = \sigma^2 I$ and $l = 2$ (2-dimentions) and $c_1 = c_2$. Thus

$$g(x) = -\frac{1}{2}x^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}x - \frac{1}{2}\mu_1{}^T\Sigma^{-1}\mu_1 + \frac{1}{2}x^T\Sigma^{-1}x - \mu_2^T\Sigma^{-1}x - \mu_2^T\Sigma^{-1}\mu_2 + \frac{1}{2}\mu_2{}^T\Sigma^{-1}\mu_2$$

$$= \mu_1^T\Sigma^{-1}x - \mu_2^T\Sigma^{-1}x - \tfrac{1}{2}\mu_1{}^T\Sigma^{-1}\mu_1 + \tfrac{1}{2}\mu_2{}^T\Sigma^{-1}\mu_2 \quad (1)$$

Σ is a diagonal matrix and $\Sigma^{-1} = \begin{pmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{pmatrix}$

Now we have that:

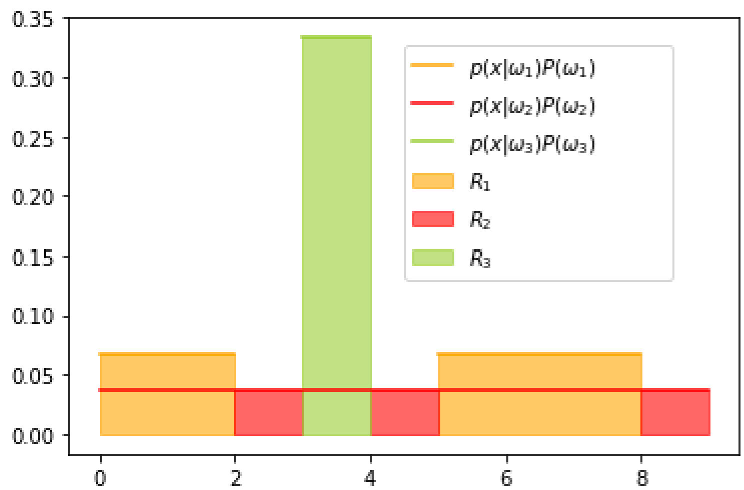
$$g(x) = \frac{1}{\sigma^2}\mu_{11}x_{11} + \frac{1}{\sigma^2}\mu_{12}x_{12} - \frac{1}{\sigma^2}\mu_{21}x_{21} - \frac{1}{\sigma^2}\mu_{22}x_{22} + \frac{1}{2\sigma^2}(\mu_{11}^2 + \mu_{12}^2) - \frac{1}{2\sigma^2}(\mu_{21}^2 + \mu_{22}^2)$$
$$= \frac{1}{\sigma^2}(\mu_1 - \mu_2)^T x - \frac{1}{2\sigma^2}||\mu_1||^2 + \frac{1}{2\sigma^2}||\mu_2||^2 \Rightarrow$$
$$g(x) = \frac{1}{\sigma^2}\left((\mu_1 - \mu_2)^T x - \frac{1}{2}||\mu_1||^2 + \frac{1}{2}||\mu_2||^2\right)$$

b) In the case that $\Sigma \neq \sigma^2 I$ the decision regions will be elliptical, and border will not be orthogonal as in case (a).

Exercise 3

a)

i.



The decision region for the three classes are:

$$R_1 = \{0,2\} \cup \{5,8\}$$
$$R_2 = \{2,3\} \cup \{4,5\} \cup \{8,9\}$$
$$R_3 = \{3,4\}$$

ii. The error of the Bayes Classifier is:

$$P_e = \frac{1}{3}\Bigg(\int_{R_1}(p(x|\omega_2) + p(x|\omega_3)dx + \int_{R_2}(p(x|\omega_1) + p(x|\omega_3)dx$$
$$+ \int_{R_3}(p(x|\omega_1) + p(x|\omega_2)dx\Bigg)$$

$$= \frac{1}{3}\left(\int_0^2 \big(p(x|\omega_2) + p(x|\omega_3)\big)dx + \int_5^8 \big(p(x|\omega_2) + p(x|\omega_3)\big)dx\right.$$
$$+ \int_2^3 (p(x|\omega_1) + p(x|\omega_3)dx + \int_4^5 (p(x|\omega_1) + p(x|\omega_3)dx$$
$$\left. + \int_8^9 (p(x|\omega_1) + p(x|\omega_3)dx + \int_3^4 (p(x|\omega_1) + p(x|\omega_2)dx\right)$$
$$= \frac{1}{3}\left(\frac{1}{9} \times \big((2-0) + (8-5)\big) + 0 + 0 + 0 + \frac{1}{9}(4-3)\right) = \frac{2}{9} = 0.222$$

iii. The point $x' = 3.5$ is classified to ω_3 since it belongs to R_3.

b)

i. For this case the following scenario must be happened:

$$P(\omega_2|x = 3.5) > P(\omega_1|x = 3.5) \Rightarrow p(x = 3.5|\omega_2)P(\omega_2) > p(x = 3.5|\omega_1)P(\omega_1)$$

$p(x = 3.5|\omega_1) = 0$ and $P(\omega_2) > 0$. Thus, the inequality is always satisfied.

$$P(\omega_2|x = 3.5) > P(\omega_3|x = 3.5) \Rightarrow p(x = 3.5|\omega_2)P(\omega_2) > p(x = 3.5|\omega_3)P(\omega_3)$$
$$\Rightarrow P(\omega_2) > 9P(\omega_3)$$

ii. There is any combination of the a priori probabilities since $p(x = 3.5|\omega_1) = 0$.

Exercise 4

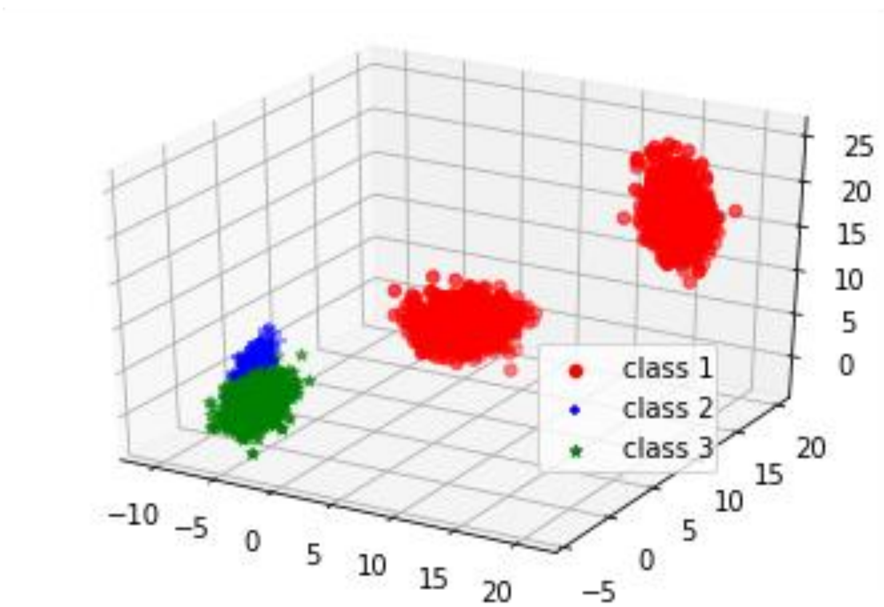
a)

```
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
from scipy.stats import multivariate_normal

training_set = sio.loadmat('Training_set.mat')
train_x = training_set['train_x']
train_y= training_set['train_y']

test_set = sio.loadmat('Test_set.mat')
test_x = test_set['test_x']
test_y = test_set['test_y']

class_1 = (train_y==1).reshape(len(train_y))
class_2 = (train_y==2).reshape(len(train_y))
class_3 = (train_y==3).reshape(len(train_y))

N = len(train_x)
```

```
N1 = np.count_nonzero(class_1)
N2 = np.count_nonzero(class_2)
N3 = np.count_nonzero(class_3)
P1 = N1/N
P2 = N2/N
P3 = N3/N
name1 = ["$P_{\omega_1}$","$P_{\omega_2}$","$P_{\omega_3}$"]
pd.DataFrame([P1,P2,P3],name1,[""])
```

P_{ω_1}	0.50
P_{ω_2}	0.25
P_{ω_3}	0.25

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
np.shape(train_y)
ax.scatter(train_x[class_1,1],train_x[class_1,2],train_x[class_1,3],c='r', marker='o',label='class 1')
ax.scatter(train_x[class_2,1],train_x[class_2,2],train_x[class_2,3],c='b', marker='+',label='class 2')
ax.scatter(train_x[class_3,1],train_x[class_3,2],train_x[class_3,3],c='g', marker='*',label='class 3')
plt.legend(bbox_to_anchor=(0.6, .2, .2, .2), loc=3,ncol=1, mode="expand", borderaxespad=0.)
plt.show()
```



i. Parametric approach
Class 1 is modeled as a mixture of two Gaussians since it consists of 2 clusters. Next we estimate the values of $p(x_i|\omega_i)P(\omega_i)$ for the samples x_i belonging to the test set. Finally, the classes 2 and 3 are modeled as Gaussians distributions.

```
from sklearn import mixture
class1 = mixture.GaussianMixture(n_components=2, covariance_type='full')
class1.fit(train_x[class_1,:])

class1_scores = np.exp(class1.score_samples(test_x))*P1
```

```python
from numpy import matlib
m2 = np.mean(train_x[class_2,:],0)
m3 = np.mean(train_x[class_3,:],0)
S2 = 1/N2*((train_x[class_2,:] - np.matlib.repmat(m2,N2,1)).T).dot((train_x[class_2,:] - np.matlib.repmat(m2,N2,1)))
S3 = 1/N3*((train_x[class_3,:] - np.matlib.repmat(m3,N3,1)).T).dot((train_x[class_3,:] - np.matlib.repmat(m3,N3,1)))
class2 = multivariate_normal(m2,S2)
class2_scores = class2.pdf(test_x)*P2
class3 = multivariate_normal(m3,S3)
class3_scores = class3.pdf(test_x)*P3
scores = np.array([class1_scores,class2_scores,class3_scores]).T

Btest_y = np.argmax(scores,axis=1).reshape(len(scores),1) + 1
Error = 1 - (np.sum(i==1 for i in Btest_y == test_y))/len(Btest_y)
print("Classification error rate - Parametric approach", Error)
```

```
Classification error rate - Parametric approach [0.012]
```

ii. Non-parametric approach

```python
from scipy.spatial import distance
from math import pi

#estimate pairwise distances betweeen test and training samples
dist = distance.cdist(test_x,train_x,'euclidean')

#estimate the volume of a 4d-hypersphere
#- dist corresponds to its radius
def vol(dist):
  return 0.5*(pi**2)*(dist**4)

k=5 #number of nearest neighbors
class1_scores = k/(N1*vol(np.sort(dist[:,class_1])[:,4]))*P1
class2_scores = k/(N2*vol(np.sort(dist[:,class_2])[:,4]))*P2
class3_scores = k/(N3*vol(np.sort(dist[:,class_3])[:,4]))*P3

scores = np.array([class1_scores,class2_scores,class3_scores]).T

Btest_y = np.argmax(scores,axis=1).reshape(len(scores),1) + 1

Error = 1 - (np.sum(i==1 for i in Btest_y == test_y))/len(Btest_y)
print("Classification error rate - Non-parametric (kNN density) approach", Error)
```

```
Classification error rate - Non-parametric (kNN density)
approach [0.0165]
```

b) Confusion matrix

- Parametric approach

```python
from sklearn.metrics import confusion_matrix
confusion_matrix(test_y, Btest_y)
```

```
array([[2000,    0,    0],
       [   0,  986,   14],
       [   0,   34,  966]], dtype=int64)
```

- Non-parametric approach

```python
confusion_matrix(test_y, Btest_y)
```

```
array([[2000,    0,    0],
       [   0,  991,    9],
       [   0,   57,  943]], dtype=int64)
```