Homework 7α

Giagkos Stylianos

f3352410

Exercise 1 Correct 1,2

2 Correct 4

3 Correct 3

4 Correct 4

5 Correct 2

6 Correct ~~2~~,3

7 Correct 1,4

8 Correct 2

9 Correct 1,2

10 Correct 2,4

11 Correct 1,3,4

12 Correct 3

13 Correct 2,3

14 Correct 1,4

15 Correct 1,4

16 Correct 1,2.

17 Correct 1

18 Correct 2,3

19 Correct 3,4

20 Correct 4

21 Correct 2,3

22 Correct 4,6,7

23 Correct 2

24 Correct 2,3,4

25 Correct 1,3,4

26 Correct 2, [illegible]

27 Correct 2,3,4

28 Correct 2

29 Correct 2,3,5

30 Correct 3

Exercise 31.

The decision surface of Bayesian for two classes is given by the equation $p(\omega_1|x) = p(\omega_2|x)$ ①

$p(x|\omega_1)p(\omega_1) = p(x|\omega_2)p(\omega_2)$ we also have $p(x|\omega_1) = p(x|\omega_2)$ since $p(\omega_1) = p(\omega_2)$

a) For normal distributions $N(\mu_1, \Sigma)$, $N(\mu_2, \Sigma)$, $\Sigma = \sigma^2 I$

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_1)^T(x-\mu_1)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_2)^T(x-\mu_2)\right) \Leftrightarrow$$

$$-\frac{1}{2\sigma^2}(x-\mu_1)^T(x-\mu_1) = -\frac{1}{2\sigma^2}(x-\mu_2)^T(x-\mu_2) \quad ②$$

$$\|x-\mu_1\|^2 = \|x-\mu_2\|^2$$

b) $\Sigma \neq \sigma^2 I$ ② becomes.

$$x^T\Sigma^{-1}x - 2\mu_1^T x + \mu_1^T\Sigma^{-1}\mu_1 = x^T\Sigma^{-1}x - 2\mu_2^T\Sigma^{-1}x + \mu_2^T\Sigma^{-1}\mu_2$$

$$\cancel{(\mu_1-\mu_2)^T\Sigma^{-1}x - 1/2\,\mu_1^{T-1}\mu_1 + 1/2\,\mu_2^T\Sigma^{-1}\mu_2 = 0.}$$

Cancelling common terms.

$$-2\mu_1^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}\mu_1 = -2\mu_2^T\Sigma^{-1}x + \mu_2^T\Sigma^{-1}\mu_2$$

Rearranging: $(\mu_1-\mu_2)^T\Sigma^{-1}x = \frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1 - \mu_2^T\Sigma^{-1}\mu_2)$.

That's hyperplane with orientation depending on Σ^{-1}

Geometric interpretation:

By transforming into linear the input space using $\Sigma^{-1/2}$ the problem is mapped into new space where Σ is identity

So the boundary is the bisector of the line segment connecting the means $\Sigma^{-1/2}\mu_1$ and $\Sigma^{-1/2}\mu_2$.

Exercise 32

Bayes classifier from HW7

In [2]:

```
import numpy as np
from scipy.stats import multivariate_normal
from scipy.io import loadmat

# Load data
data = loadmat("HW8.mat")
train_x = data["train_x"]
train_y = data["train_y"].flatten()
test_x = data["test_x"]
test_y = data["test_y"].flatten()

# (i) Estimate parameters
def estimate_parameters(train_x, train_y, class_label):
    class_data = train_x[train_y == class_label]
    n_class = len(class_data)
    mu = np.mean(class_data, axis=0)
    sigma = np.cov(class_data, rowvar=False)
    return mu, sigma, n_class

# Estimate for both classes
mu1, sigma1, n1 = estimate_parameters(train_x, train_y, class_label=1)
mu2, sigma2, n2 = estimate_parameters(train_x, train_y, class_label=2)

# Priors
P_w1 = n1 / len(train_y)
P_w2 = n2 / len(train_y)

# (ii) Classify test data
Btest_y = []
for x in test_x:
    # Compute likelihoods
    p_x_given_w1 = multivariate_normal.pdf(x, mean=mu1, cov=sigma1)
    p_x_given_w2 = multivariate_normal.pdf(x, mean=mu2, cov=sigma2)

    # Compute posteriors
    post_w1 = P_w1 * p_x_given_w1
    post_w2 = P_w2 * p_x_given_w2

    # Assign to the class with the highest posterior
    Btest_y.append(1 if post_w1 > post_w2 else 2)

Btest_y = np.array(Btest_y)

# (iii) Estimate error classification probability
correct_predictions = np.sum(Btest_y == test_y)
error_rate = 1 - (correct_predictions / len(test_y))

# Print results
```

```python
print(f"Bayes Classifier Results :")
print(f"Class Priors: P(w1) = {P_w1:.3f}, P(w2) = {P_w2:.3f}")
print(f"Means: mu1 = {mu1}, mu2 = {mu2}")
print(f"Covariances: sigma1 =\n{sigma1}\n, sigma2 =\n{sigma2}")
print(f"Classification Error Rate: {error_rate:.3f}")
```

```
Bayes Classifier Results :
Class Priors: P(w1) = 0.500, P(w2) = 0.500
Means: mu1 = [0.14549472 0.11840199], mu2 = [ 2.07024339 -1.89136529]
Covariances: sigma1 =
[[3.63737014 1.74128017]
 [1.74128017 4.22056748]]
, sigma2 =
[[4.71777486 2.6006903 ]
 [2.6006903  4.37763924]]
Classification Error Rate: 0.150
```

Exercise 32 (python code + text):

Consider a two-class, two-dimensional classification problem for which you can find attached two sets: one for training and one for testing (file *HW8.mat*). Each of these sets consists of pairs of the form $(y_i, \boldsymbol{x}_i)$, where y_i is the class label for vector $\boldsymbol{x}_i$. Let N_{train} and N_{test} denote the number of training and test sets, respectively. The data are given via the following arrays/matrices:

- ***train_x*** (a $N_{train} \times 2$ **matrix** that contains in its rows the training vectors $\boldsymbol{x}_i$)
- ***train_y*** (a N_{train}–dim. column **vector** containing the **class labels** (1 or 2) of the corresponding training vectors $\boldsymbol{x}_i$ included in ***train_x***).
- ***test_x*** (a $N_{test} \times 2$ **matrix** that contains in its rows the test vectors $\boldsymbol{x}_i$)
- ***test_y*** (a N_{test}–dim. column **vector** containing the **class labels** (1 or 2) of the corresponding test vectors $\boldsymbol{x}_i$ included in ***test_x***).

Assume that the two classes, ω_1 and ω_2 are modeled by normal distributions.

Recall that $\boldsymbol{x} = [x_1, x_2]^T$

(a) Adopt the **naïve Bayes classifier**.

i. Use the training set to estimate $P(\omega_1), P(\omega_2), p(x_1|\omega_1), p(x_2|\omega_1), p(x_1|\omega_2), p(x_2|\omega_2)$ (Each $p(\boldsymbol{x}|\omega_j)$ is written as $p(\boldsymbol{x}|\omega_j) = p(x_1|\omega_j) \cdot p(x_2|\omega_j)$. Use the ***ML* estimates** of the mean and variance for each one of the 1-dim. pdfs).

ii. Classify the points $\boldsymbol{x}_i = [x_{i1}, x_{i2}]^T$ of the test set, using the naïve Bayes classifier (Estimate $p(\boldsymbol{x}|\omega_j)$ with $p(x_{i1}|\omega_j) \cdot p(x_{i2}|\omega_j)$ and then apply the Bayes rule. Keep the class labels, to an a N_{test}–dim. column **vector**, called ***NBtest_y*** containing the **estimated class labels** (1 or 2) of the corresponding test vectors $\boldsymbol{x}_i$ included in ***test_x***)

iii. Estimate the error classification probability ((1) **compare** ***test_y*** and ***NBtest_y***, (2) **count** the positions where both of them have the same class label and (3) **divide** with the total number of test vectors).

(b) Depict graphically the training set, using different colors for points from different classes.

(c) Report the classification results obtained by (a) the Bayes classifier (see exercise 32 of Homework 7) and (b) the naïve Bayes classifier and comment on them. Under what conditions, the Bayes and the naïve Bayes classifiers would exhibit the same performance?

Hint: Use the attached Python code in file *HW8.ipynb* (also given in Homework 7).

```
In [4]: import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt
from scipy.stats import norm

# Load the data from the HW8.mat file
data = loadmat('HW8.mat')
train_x = data['train_x']
train_y = data['train_y'].flatten()
test_x = data['test_x']
test_y = data['test_y'].flatten()

# Separate the data by classes for training
train_x_class1 = train_x[train_y == 1]
train_x_class2 = train_x[train_y == 2]

# Number of training and test samples
N_train = train_x.shape[0]
N_test = test_x.shape[0]
```

(a) Naïve Bayes Classifier:

i. Estimate Class Probabilities and Conditional Densities:

```
In [6]: # (i) Estimate parameters for Naïve Bayes
def estimate_naive_parameters(train_x, train_y, class_label):
    class_data = train_x[train_y == class_label]
    mu = np.mean(class_data, axis=0)
    sigma2 = np.var(class_data, axis=0)  # Variance for each feature
    return mu, sigma2

# Estimate parameters for each class
mu1_naive, sigma2_1_naive = estimate_naive_parameters(train_x, train_y, class_label
mu2_naive, sigma2_2_naive = estimate_naive_parameters(train_x, train_y, class_label

# Priors
P_w1_naive = np.mean(train_y == 1)
P_w2_naive = np.mean(train_y == 2)
```

ii. Classify Test Points Using Naïve Bayes:

```
In [8]: # (ii) Classify test data using Naïve Bayes
NBtest_y = []
for x in test_x:
    # Compute likelihoods
    p_x1_w1 = norm.pdf(x[0], mu1_naive[0], np.sqrt(sigma2_1_naive[0]))
    p_x2_w1 = norm.pdf(x[1], mu1_naive[1], np.sqrt(sigma2_1_naive[1]))
    p_x_given_w1 = p_x1_w1 * p_x2_w1  # Naïve assumption: independence of features

    p_x1_w2 = norm.pdf(x[0], mu2_naive[0], np.sqrt(sigma2_2_naive[0]))
    p_x2_w2 = norm.pdf(x[1], mu2_naive[1], np.sqrt(sigma2_2_naive[1]))
    p_x_given_w2 = p_x1_w2 * p_x2_w2

    # Compute posteriors
```

```python
    post_w1_naive = P_w1_naive * p_x_given_w1
    post_w2_naive = P_w2_naive * p_x_given_w2

    # Assign to the class with the highest posterior
    NBtest_y.append(1 if post_w1_naive > post_w2_naive else 2)

NBtest_y = np.array(NBtest_y)
```

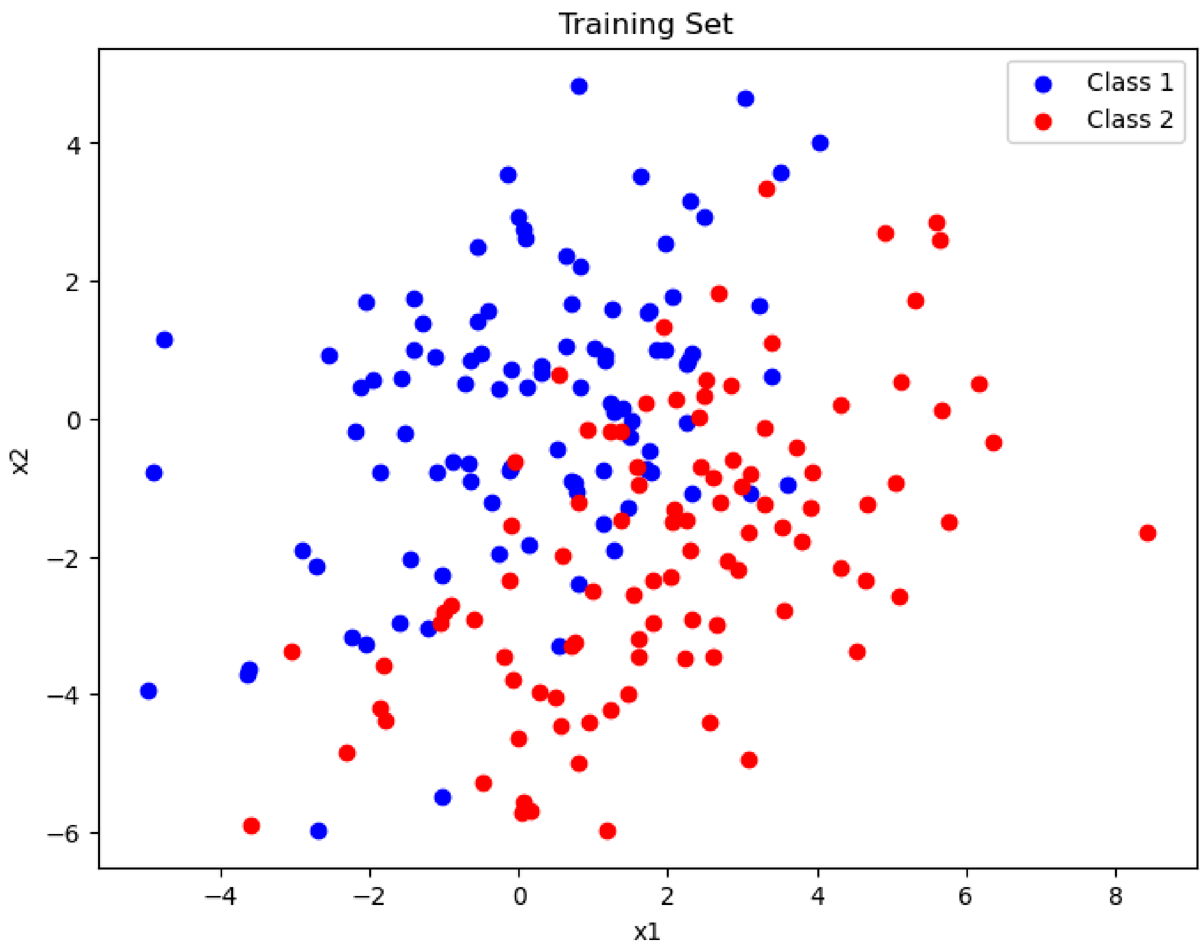
iii. Estimate classification error

In [10]:
```python
# (iii) Estimate error classification probability
correct_predictions_naive = np.sum(NBtest_y == test_y)
error_rate_naive = 1 - (correct_predictions_naive / len(test_y))
```

In [11]:
```python
# Print Naïve Bayes results
print(f"Naïve Bayes Class Priors: P(w1) = {P_w1_naive:.3f}, P(w2) = {P_w2_naive:.3f
print(f"Naïve Bayes Means: mu1 = {mu1_naive}, mu2 = {mu2_naive}")
print(f"Naïve Bayes Variances: sigma^2_1 = {sigma2_1_naive}, sigma^2_2 = {sigma2_2_
print(f"Naïve Bayes Classification Error Rate: {error_rate_naive:.3f}")
```

```
Naïve Bayes Class Priors: P(w1) = 0.500, P(w2) = 0.500
Naïve Bayes Means: mu1 = [0.14549472 0.11840199], mu2 = [ 2.07024339 -1.89136529]
Naïve Bayes Variances: sigma^2_1 = [3.60099644 4.17836181], sigma^2_2 = [4.67059711
4.33386285]
Naïve Bayes Classification Error Rate: 0.165
```

In [12]:
```python
# (b) Plot training data
plt.figure(figsize=(8, 6))
plt.scatter(train_x[train_y == 1][:, 0], train_x[train_y == 1][:, 1], color='blue',
plt.scatter(train_x[train_y == 2][:, 0], train_x[train_y == 2][:, 1], color='red',
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Training Set')
plt.legend()
plt.show()
```



In [13]:
```
# (c) Comparison
# Print Bayes results
print(f"Bayes Class Priors: P(w1) = {P_w1:.3f}, P(w2) = {P_w2:.3f}")
print(f"Bayes Means: mu1 = {mu1}, mu2 = {mu2}")
print(f"Bayes Covariances: sigma1 =\n{sigma1}, sigma2 =\n{sigma2}")
print(f"Bayes Classification Error Rate: {error_rate:.3f}")
print("------------------------------------------------")
# Print Naïve Bayes results
print(f"Naïve Bayes Class Priors: P(w1) = {P_w1_naive:.3f}, P(w2) = {P_w2_naive:.3f
print(f"Naïve Bayes Means: mu1 = {mu1_naive}, mu2 = {mu2_naive}")
print(f"Naïve Bayes Variances: sigma^2_1 = {sigma2_1_naive}, sigma^2_2 = {sigma2_2_
print(f"Naïve Bayes Classification Error Rate: {error_rate_naive:.3f}")
```

```
Bayes Class Priors: P(w1) = 0.500, P(w2) = 0.500
Bayes Means: mu1 = [0.14549472 0.11840199], mu2 = [ 2.07024339 -1.89136529]
Bayes Covariances: sigma1 =
[[3.63737014 1.74128017]
 [1.74128017 4.22056748]], sigma2 =
[[4.71777486 2.6006903 ]
 [2.6006903  4.37763924]]
Bayes Classification Error Rate: 0.150
-----------------------------------------------
Naïve Bayes Class Priors: P(w1) = 0.500, P(w2) = 0.500
Naïve Bayes Means: mu1 = [0.14549472 0.11840199], mu2 = [ 2.07024339 -1.89136529]
Naïve Bayes Variances: sigma^2_1 = [3.60099644 4.17836181], sigma^2_2 = [4.67059711
4.33386285]
Naïve Bayes Classification Error Rate: 0.165
```

Classification Results

Bayes Classifier:

- Class Priors: P(w1) = 0.500, P(w2) = 0.500
- Means: mu1 = [0.1455, 0.1184], mu2 = [2.0702, -1.8914]
- Covariances: sigma1 = [[3.6374, 1.7413], [1.7413, 4.2206]], sigma2 = [[4.7178, 2.6007], [2.6007, 4.3776]]
- Error Rate: 0.150

Naïve Bayes Classifier:

- Class Priors: P(w1) = 0.500, P(w2) = 0.500
- Means: mu1 = [0.1455, 0.1184], mu2 = [2.0702, -1.8914]
- Variances: sigma^2_1 = [3.6010, 4.1784], sigma^2_2 = [4.6706, 4.3339]
- Error Rate: 0.165

Comments:

1. **Performance:** Bayes classifier has a lower error rate due to accounting for feature correlations.
2. **Equal Performance Conditions:** When covariance matrices are diagonal or features are independent.
3. **Implications:**
   - Bayes: Best for correlated features and sufficient data.
   - Naïve Bayes: Simpler and effective for small datasets or when features are independent.

In [ ]: