# MAT 3373 Winter 2023. Assignment 4

Mohammad Alqahtani, Sebastian Doka and Yasmine Zoubdi

2023-03-11

**Total Marks: 20. (5 questions on 10 pages)**

## Question 1 [6 marks]

Consider the following generative model: there are two equally-probable classes: 'blue' and 'orange', and within each, the vector-valued variable $X = \left(X^1, X^2\right)$ has a multivariate Gaussian distribution, with the following parameters:

- blue class: $\mu_b = (0,0), \Sigma_b = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix}$
- orange class: $\mu_o = (0,0), \Sigma_o = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$

Consider the optimal Bayes classifier, i.e. the Bayes classifier corresponding to this true joint distribution of $(X, Y)$, where $Y$ is the class label.

*Note:* This Bayes classifier is the same as in QDA, except that we know the true parameters instead of having to estimate them.

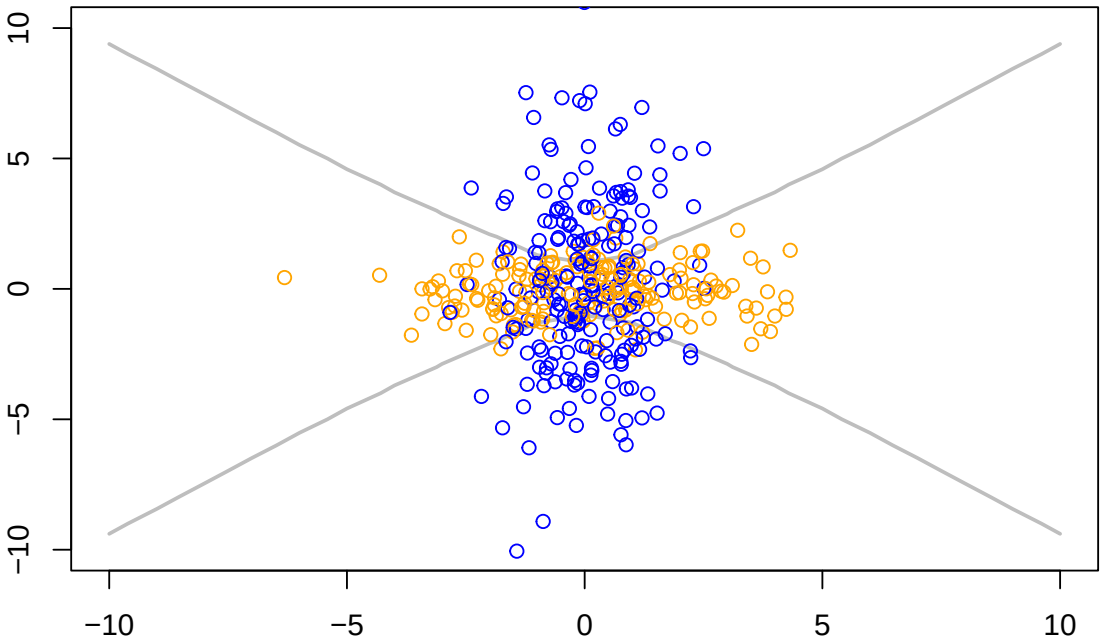
### A) Plot the decision boundary as in Assignment 3.

```
library(mnormt)
```

```
## Warning: package 'mnormt' was built under R version 4.1.3
```

```
library(MASS)


QDA2 = function(X, pi_m, mu_m, Sigma_m, mu_f, Sigma_f){
  # Inputs:
  # X = a dataframe with two columns (the predictors)
  # pi_m = a prior probability for class 'male'
  # All the other inputs are vectors of length 2, corresponding to 2 predictor variables
  # mu_m = a mean vector for class 'male'
  # Sigma_m = a covariance matrix for class 'male'
  # mu_f = a mean vector for class 'female'
  # Sigma_f = a covariance matrix for class 'female'
  #
```

```
  # Output: the posterior probability for class 'male'

  likelihood_male = dmnorm(X, mean=mu_m, varcov = Sigma_m)
  likelihood_female = dmnorm(X, mean=mu_f, varcov = Sigma_f)

  # unnormalized probabilities:
  unp_male = likelihood_male * pi_m
  unp_female = likelihood_female * (1 - pi_m)

  return((unp_male)/(unp_male + unp_female))
}

mu_b = c(0, 0)
sigma_b = matrix(c(1, 0, 0, 10), nrow = 2, ncol = 2, byrow = TRUE)
mu_o = c(0, 0)
sigma_o = matrix(c(4, 0, 0, 1), nrow = 2, ncol = 2, byrow = TRUE)
grid = expand.grid(x=-10:10, y=-10:10)

prob.grid = QDA2(grid, 0.5, mu_b, sigma_b, mu_o, sigma_o)

# plot the boundary
contour(x=-10:10, y=-10:10, z=matrix(prob.grid, nrow=21), levels=0.5,
        col="grey", drawlabels=FALSE, lwd=2)

# Finally, plot the training data points on the same figure:
n = 400
data = matrix(0, nrow = n, ncol = 2)
labels <- sample(c("blue", "orange"), size = n, replace = TRUE)

for (i in 1:n) {
  if (labels[i] == "blue") {
    data[i,] <- mvrnorm(n = 1, mu = mu_b, Sigma = sigma_b)
  } else {
    data[i,] <- mvrnorm(n = 1, mu = mu_o, Sigma = sigma_o)
  }
}


points(data[,1], data[,2], col=labels)
```




**B) Compute the exact decision boundary mathematically.**

We solve the equation : $P(Y = "Blue"/X = (x_1, x_2)) = P(Y = "Orange"/X = (x_1, x_2))$ (because the posterior probabilities are equal)

$=> \frac{P(Y="Blue"/X=(x_1,x_2))}{P(Y="Orange"/X=(x_1,x_2))} = 1$

$=> \frac{\frac{1}{2\pi}.det(\Sigma_b)^{-1/2}.e^{-\frac{1}{2}(X-\mu_b)^\top.(\Sigma_b)^{-1}.(X-\mu_b)}}{\frac{1}{2\pi}.det(\Sigma_o)^{-1/2}.e^{-\frac{1}{2}(X-\mu_o)^\top.(\Sigma_o)^{-1}.(X-\mu_o)}} = 1$

$=> log(\frac{det(\Sigma_b)^{-1/2}.e^{-\frac{1}{2}(X-\mu_b)^\top.(\Sigma_b)^{-1}.(X-\mu_b)}}{det(\Sigma_o)^{-1/2}.e^{-\frac{1}{2}(X-\mu_o)^\top.(\Sigma_o)^{-1}.(X-\mu_o)}}) = 0$

$=> -\frac{1}{2}.log(\frac{det(\Sigma_b)}{det(\Sigma_o)}) - \frac{1}{2}.(X-\mu_b)^\top.(\Sigma_b)^{-1}.(X-\mu_b) + \frac{1}{2}.(X-\mu_o)^\top.(\Sigma_o)^{-1}.(X-\mu_o) = 0$

$=> -log(\frac{10}{4}) - (X)^\top.(\Sigma_b)^{-1}.(X) + (X)^\top.(\Sigma_o)^{-1}.(X) = 0$

$=> -log(\frac{10}{4}) - x_1^2 - \frac{1}{10}x_2^2 + x_2^2 + \frac{1}{4}x_1^2 = 0$

$=> x_1^2(\frac{1}{4} - 1) + x_2^2(1 - \frac{1}{10}) = log(\frac{10}{4})$

since $log(\frac{10}{4}) \approx 1$ :

$=> x_2^2(\frac{9}{10}) - x_1^2(\frac{3}{4}) = 1$

and this is a hyperbola equation with $a = \sqrt{\frac{10}{9}}$ and $b = \sqrt{\frac{4}{3}}$ with centre (0,0).

**C) For generative models with Gausssian class-conditional distributions, the decision boundary is always quadratic.**

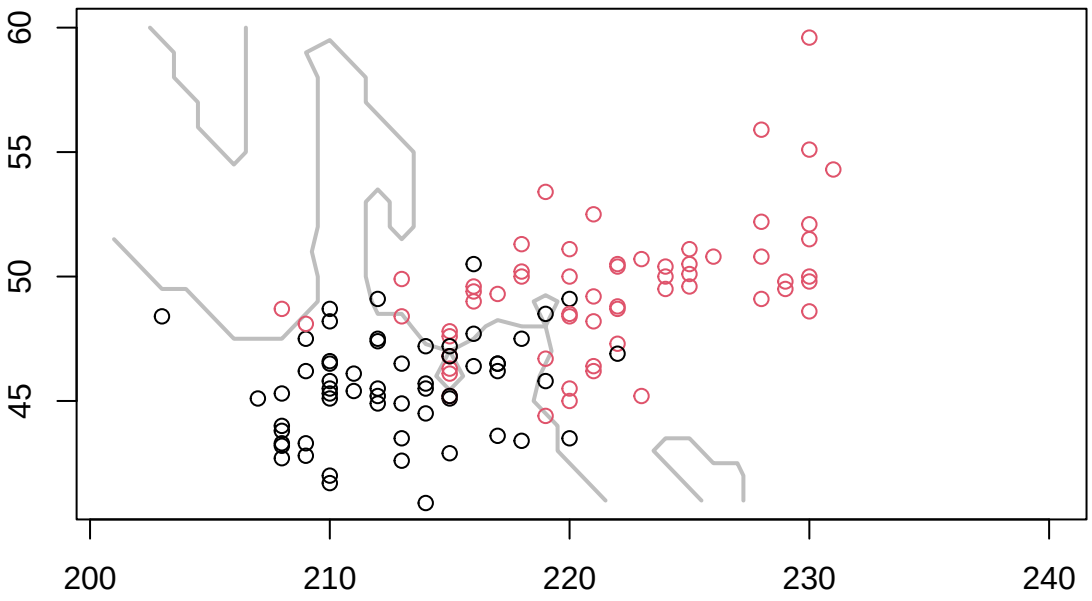
In fact, all conic sections are possible: circles, ellipses, parabolas and hyperbolas. We most often see parabolas, especially when the classes are well-separated. Give an example (mean vectors and covariance matrices for two classes) that leads to a circular or elliptical decision boundary. Justify your answer with *either* an exact calculation *or* a numerical computation of the approximate boundary.

we know that the general equation of any type of circle is represented by: $x_1^2 + x_2^2 + 2gx_1 + 2fx_2 + c = 0$, for all values of g, f and $c \neq g^2 + f^2$ in R.
if :

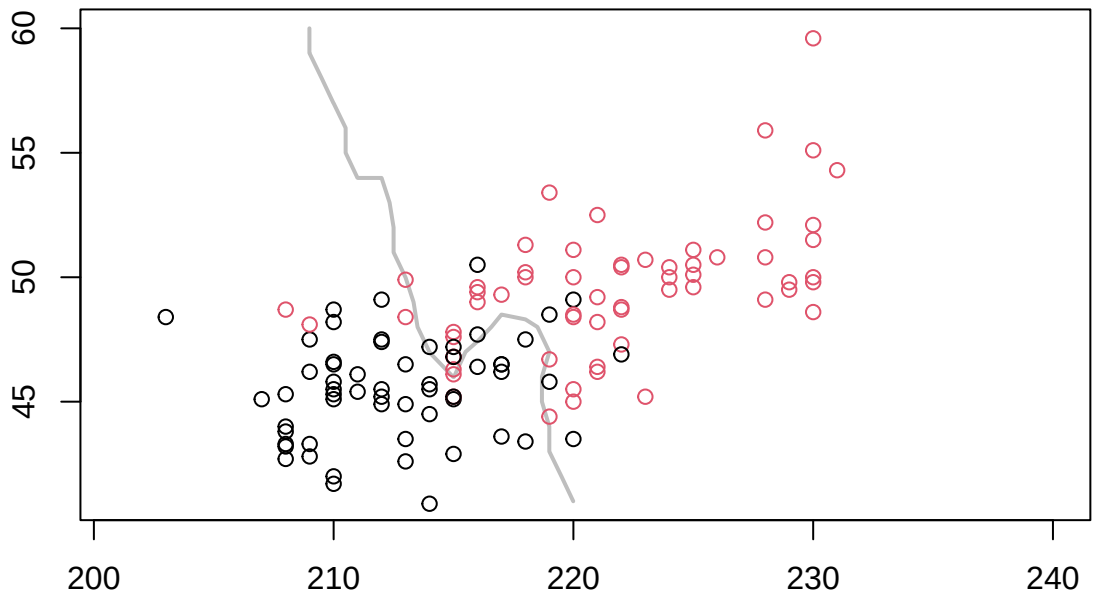
- blue class: $\mu_b = (-g, -f), \Sigma_b = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$
- orange class: $\mu_o = (0, 0), \Sigma_o = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$

in this case $c = -\frac{1}{2}.log(\frac{det(\Sigma_b)}{det(\Sigma_o)}) = -\frac{1}{2}.log(4)$
and we can choose g = f = 0,
the equation is then equal to $x_1^2 + x_2^2 = \frac{1}{2}.log(4)$

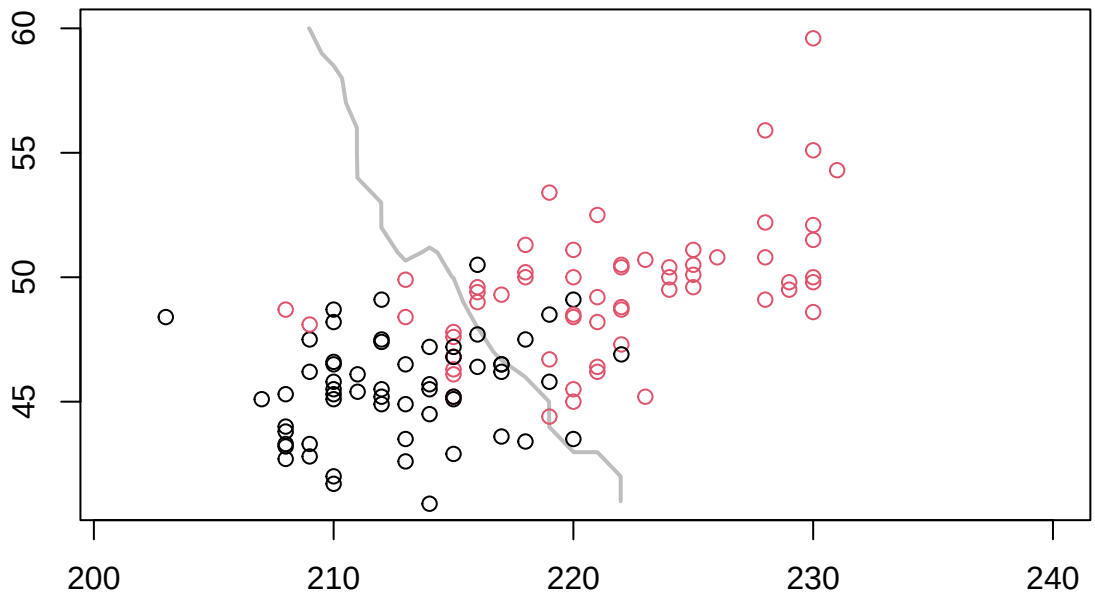
**Question 2 [3 marks]**



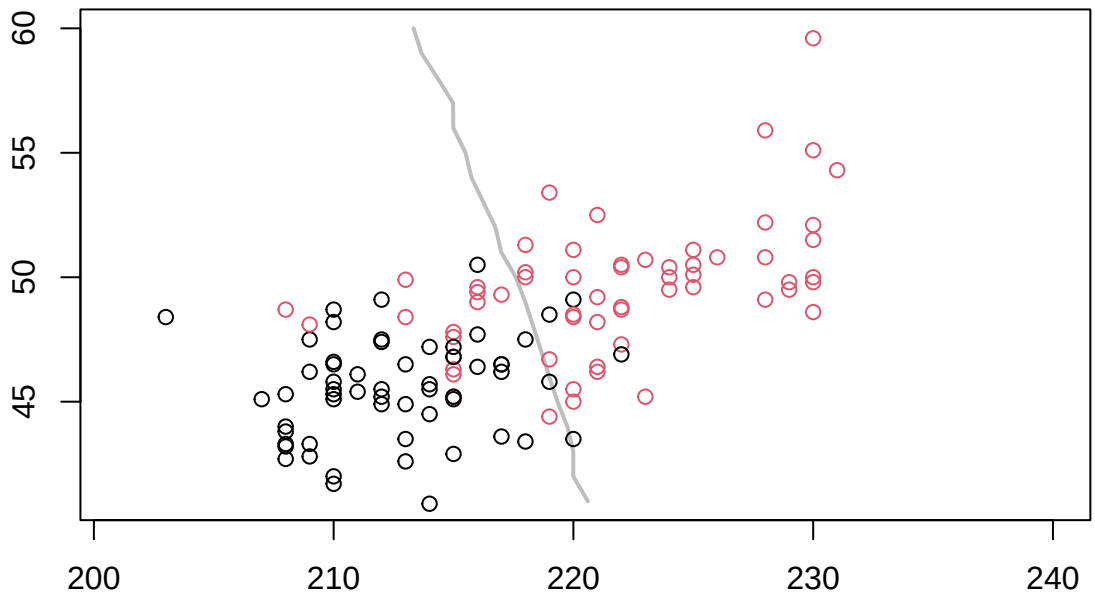

With k = 3:
With k = 10:




With k = 30:




With k = 100:




## Question 3 [7 marks]

```
N = 50
noise_sd = 0.2
set.seed(42)
xtrain = runif(N, min=0, max=2*pi) # sample data uniformly from [0,2pi]
ytrain = sin(xtrain) + rnorm(N, mean=0, sd=noise_sd)
xtest = runif(N, min=0, max=2*pi)
ytest = sin(xtest) + rnorm(N, mean=0, sd=noise_sd)
# plot(xtrain, ytrain)
```

```
knn.fit = knnreg(matrix(xtrain), ytrain, k=5)
ypred = predict(knn.fit, xtest)
# plot(ytest, ypred)
```

**A) Evaluate the Test MSE for 5-NN in the above example.**

```
mse = mean((ytest - ypred)^2)
mse
```

```
## [1] 0.06122751
```

**B) Now we add a second "predictor variable" that in fact is irrelevant to the prediction:**

```
x2train = runif(N, min=0, max=2*pi)
Xtrain = cbind(xtrain, x2train)

x2test = runif(N, min=0, max=2*pi)
Xtest = cbind(xtest, x2test)

# Re-do 5-NN using the matrices Xtrain and Xtest
knn.fit2 = knnreg(Xtrain, ytrain, k=5)
ypred2 = predict(knn.fit2, Xtest)

# Evaluate the Test MSE, and compare with (A).
mse2 = mean((ytest - ypred2)^2)
mse2
```

```
## [1] 0.1962908
```

**Why do you think this happened?** It is evidant that we get a larger MSE in B than in C. This is because the second predictor variable x2train and x2test only add noise to the model, they are irrelevant to the prediction of y. Therefore, we can conclude that the additional variable/noise make the model less accurate.

**C) Apply polynomial regression with your choice of the degree of the polynomial, and calculate the Test MSE.**

```
# Fit polynomial regression of degree 2
poly.fit2 = lm(ytrain ~ poly(xtrain, 2) + poly(x2train, 2))
ypred2 = predict(poly.fit2, data.frame(xtest, x2test))
test_mse2 = mean((ypred2 - ytest)^2)

# Fit polynomial regression of degree 3
poly.fit3 = lm(ytrain ~ poly(xtrain, 3) + poly(x2train, 3))
ypred3 = predict(poly.fit3, data.frame(xtest, x2test))
test_mse3 = mean((ypred3 - ytest)^2)

# Select the degree with the lowest Test MSE
if (test_mse2 < test_mse3) {
  poly.fit = poly.fit2
  ypred = ypred2
  test_mse = test_mse2
} else {
  poly.fit = poly.fit3
  ypred = ypred3
  test_mse = test_mse3
}

# Generate another test set and calculate Test MSE
N = 50
noise_sd = 0.2
```

```r
set.seed(43)

xtrain_new = runif(N, min=0, max=2*pi)
ytrain_new = sin(xtrain_new) + rnorm(N, mean=0, sd=noise_sd)

x2train_new = runif(N, min=0, max=2*pi)
Xtrain_new = cbind(xtrain_new, x2train_new)

xtest_new = runif(N, min=0, max=2*pi)
ytest_new = sin(xtest_new) + rnorm(N, mean=0, sd=noise_sd)
x2test_new = runif(N, min=0, max=2*pi)
Xtest_new = cbind(xtest_new, x2test_new)

ypred_new = predict(poly.fit, data.frame(xtest_new, x2test_new))
final_test_mse = mean((ypred_new - ytest_new)^2)
final_test_mse
```

```
## [1] 0.618595
```

## Question 4 [2 marks]

If $Y = f(X) + \epsilon$ and $f$ is a convex function, do you expect that the prediction given by KNN, for a particular value $X = x_0$, has a positive, negative, or zero bias, and why? *Justify your answer in one or (max) two sentences.*

The prediction has a positive bias because firstly looking at the graphs in question 3, we are looking at the minumum value of x0(in this case y=-1) of a convex function meaning all the true values of y will be higher than all the other x-values therefore the knn prediction value of x0(in this case y=1) will be an overestimate of the true value. Secondly, because the bias is the difference between the true value of y(=-1) and the prediction value of y(=1), the bias will be 1-(-1)= 2 again meaning it is a positive bias.

## Question 5 [2 marks]

The following plots were produced by applying KNN regression to a fictional dataset. The blue curve (top) shows Training error as a function of $K$, and the red curve (bottom) shows Validation error as a function of $K$. Is the algorithm overfitting, or underfitting, or neither, at $K = 5$? Justify your answer in one or (max) two sentences.

The algorithm is overfitting because firstly, if assuming that the model is good at K=10 then on the right side both the training and validation errors are high so it is underfitting and on the left side where we have K=5 the training error is low(=0.6) and much smaller than the validation error(=25) which is much higher so the algorithm is overfitting. Secondly, the difference between the training error and the validation error(=24.4) is larger compared to higher values of K.