1 Part I: Computer Exercises

We import some exclusive packages.

```
library(lattice)
library(ggplot2)
library(caret)
library(SimDesign)
library(MASS)
library(matlib)
library(Rfast)
```

1.1 General Theory

In this Home Assignment, we will use the Linear Discriminant Analysis (LDA) as probability model, as presented in Home Assignment 1, Task 2. This model will be compared to Fisher's LDA, which solves the same problem with a non-parametric approach. In short, Fisher's LDA derives the optimal direction of the discriminant, or the weights, by maximizing the distance in mean as well as minimizing the within-class variance. These objectives can be summarized as

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{\boldsymbol{w}^\intercal \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^\intercal \boldsymbol{S}_W \boldsymbol{w}}$$

where $\boldsymbol{S}_B = (\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)^\intercal$ is the between-class variance, and $\boldsymbol{S}_W = \sum_{i \in \mathcal{C}_1} (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_1)(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_1)^\intercal + \sum_{i \in \mathcal{C}_2} (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_2)^\intercal$ is the within-class variance. It can be shown that the maximum is obtained by letting $\boldsymbol{w} = \boldsymbol{S}_W^{-1}(\bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1)$. It should also be noted that, in the two-dimensional case, the Fisher's approach yields the same results as the "ordinary", Gaussian approach.

1.2 Task 1

1.2.1 Method

Firstly, the two sets of bivariate normal variables with corresponding labels, "1" and "2", were generated. Then, these were visualized through scatter plot where the first class was colored red, and the other one green.

1.2.2 Source Code

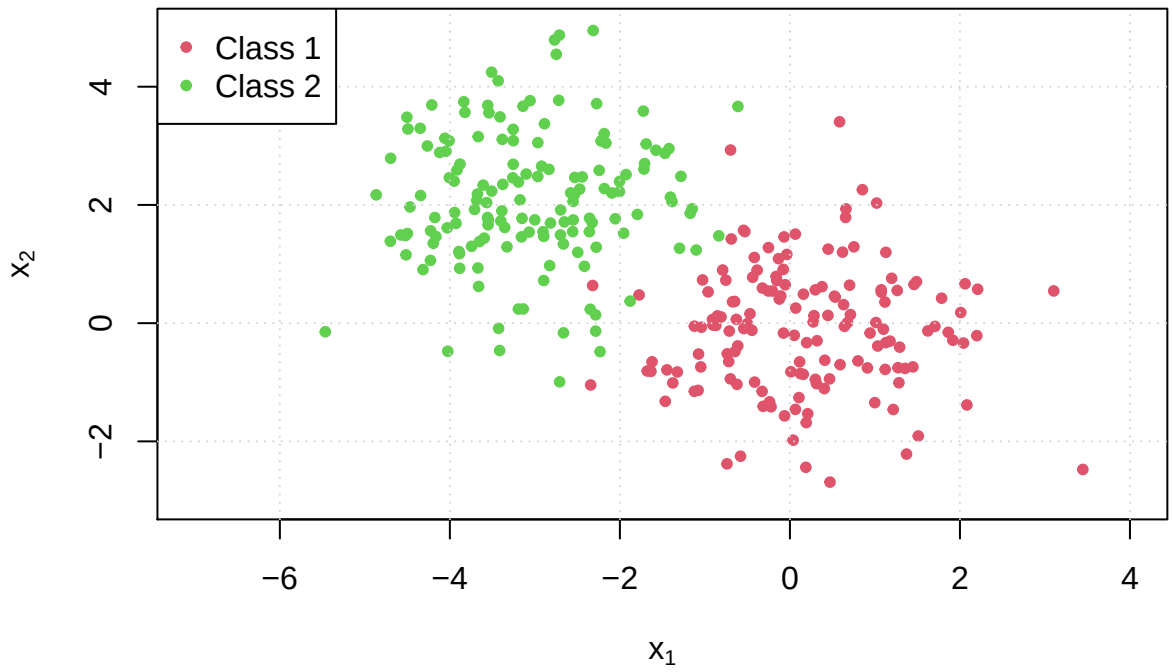
Generate two sets of bivariate normal numbers and visualize these with different colors.

```
set.seed(1337)
n1 = n2 = 150
mu1 = c(0, 0)
mu2 = c(-3, 2)

z1 = SimDesign::rmvnorm(n1, mu1)
lab1 = rep(1, n1)

z2 = SimDesign::rmvnorm(n2, mu2)
lab2 = rep(2, n2)
```

```r
plot(
    z1,
    col=2,
    xlim=c(-7, 4),
    ylim = c(-3, 5),
    xlab=expression('x'[1]),
    ylab=expression('x'[2]),
    pch=20
)
points(z2, col=3, pch=20)
grid()
legend(
    "topleft",
    legend=c("Class 1", "Class 2"),
    pch=20,
    col=c(2, 3),
    bg="white"
)
```



1.3 Task 2

1.3.1 Method

Consequently, the LDA model was fitted through MLE estimation, which simply corresponds to the sample mean and the pooled covariance matrix, since the simulated variables were uncorrelated. Thus, the decision boundary could be

expressed as

$$\hat{y} = (\bar{x}_2 - \bar{x}_1)^\intercal S_p^{-1} \boldsymbol{x} = \frac{1}{2}(\bar{x}_1 + \bar{x}_2)^\intercal S_p^{-1}(\bar{x}_2 - \bar{x}_1) = \hat{m},$$

where S_p is the pooled covariance matrix.

1.3.2 Source Code

Estimate the LDA parameters by MLE, i.e., computing sample mean and pooled covariance matrix, since our simulated sets of variables are uncorrelated. Then, we compute the corresponding decision boundary and plot it together with the simulated values.

```
mu1_s = colMeans(z1)
mu2_s = colMeans(z2)
S = pooled.cov(rbind(z1, z2), c(lab1, lab2)) # pooled covariance

MU_pos = mu1_s + mu2_s
dim(MU_pos) = c(2, 1)

MU_neg = mu2_s - mu1_s
dim(MU_neg) = c(2, 1)

c = 0.5 * t(MU_pos) %*% inv(S) %*% MU_neg
x = t(MU_neg) %*% inv(S)

k = x[1] / -x[2]
m = c / x[2]
```

```
plot(
    z1,
    col=2,
    xlim=c(-7, 4),
    ylim = c(-3, 5),
    xlab=expression('x'[1]),
    ylab=expression('x'[2]),
    pch=20
)
points(z2, col=3, pch=20)
abline(a=m, b=k)
grid()
legend(
    "topleft",
    legend=c("Class 1", "Class 2"),
    pch=20,
    col=c(2, 3),
    bg="white"
)
```

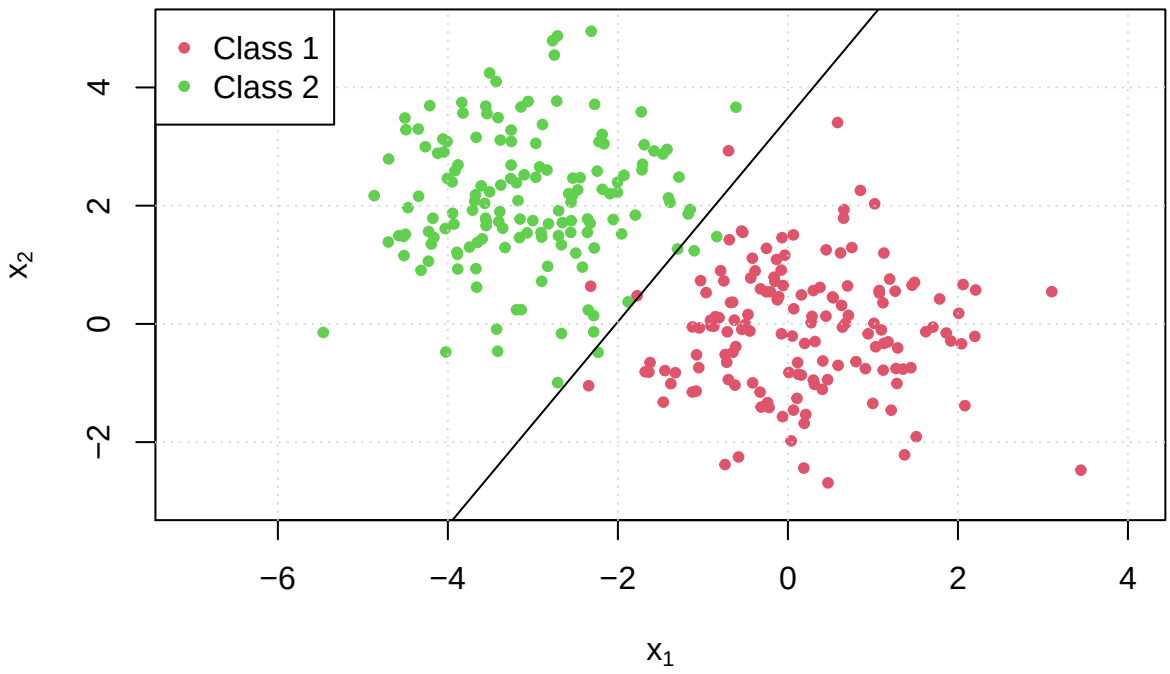


Figure 1: *Visualization of the two sets of simulated normal multivariate numbers, along with the MLE decision boundary.*

1.4 Task 3

1.4.1 Method

Further, a LDA was performed by the built-in LDA function. This function computes the direction of Fisher's projection, i.e., $\boldsymbol{w}$, as defined in Section 1.1. Then, the corresponding decision boundary was calculated and visualized side-by-side with the boundary based on MLE estimation.

1.4.2 Conclusion

As expected, the setting in the two-dimensional case, yields the exact same decision boundary for both the ordinary LDA and Fisher's method.

1.4.3 Source Code

Create a neat data frame with the simulated values and their corresponding labels.

```
# stack observations and labels to data frame
Z = rbind(z1, z2)
Y = as.factor(c(lab1, lab2))
dat = as.data.frame(cbind(Z, Y))
```

Fit LDA with built-in package in R.

```
# fit LDA model
mdl = lda(
    Y ~ V1 + V2,
    data=dat
)
```

Consequently, we compute the decision boundary based on the built-in LDA and compare it with the one in Task 2.

```
mu = mdl$means
w = mdl$scaling
mu_vec = colSums(mu)
dim(mu_vec) = c(2, 1)
c = 0.5 * t(w) %*% mu_vec
lda_k = w[1] / -w[2]
lda_m = c / w[2]
```

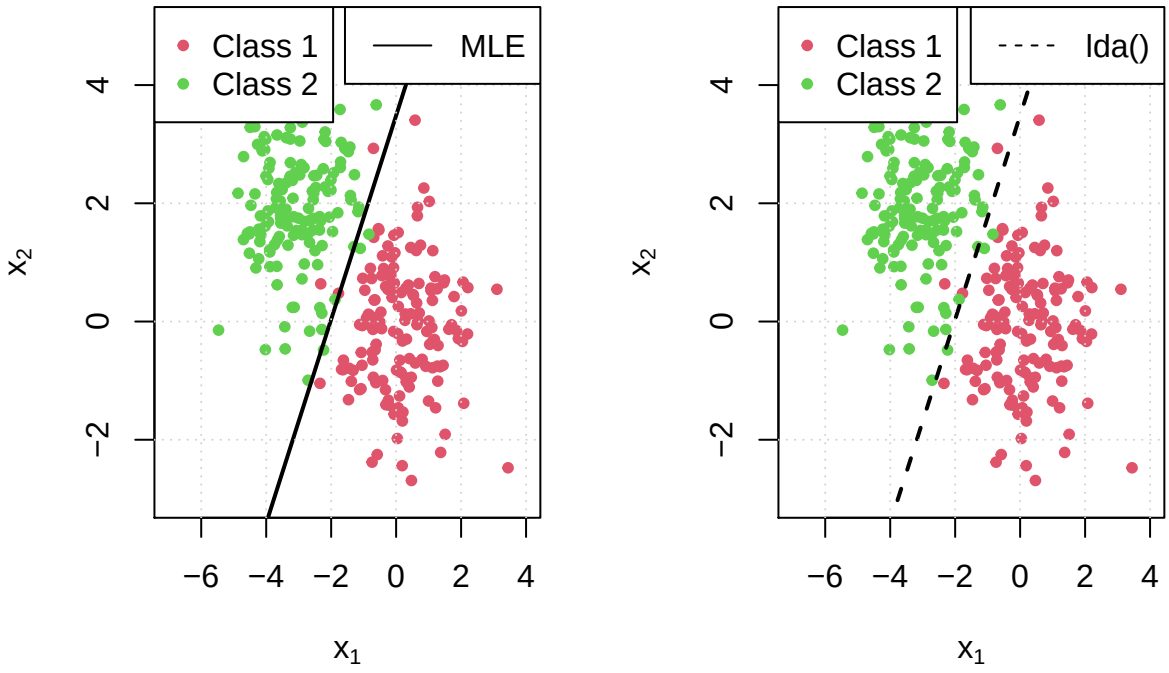


Figure 2: *Visualization of the two sets of simulated normal multivariate numbers, along with the MLE (l.h.s.) and built-in LDA (r.h.s.) decision boundary, respectively.*

The decision boundary for each method presented on the form $y = kx + m$, which should be identical.

```
paste("Task 2 - MLE decision boundary: y = ",
      round(k, 4), "x + ", round(m, 4), sep=""
)
```

```
## [1] "Task 2 - MLE decision boundary: y = 1.7245x + 3.4854"
```

```
paste("Task 3 - lda() decision boundary: y = ",
      round(lda_k, 4), "x + ", round(lda_m, 4), sep=""
)
```

```
## [1] "Task 3 - lda() decision boundary: y = 1.7245x + 3.4854"
```

1.5 Task 4

1.5.1 Method

We simply followed the nice and clear steps requested in the lab specification!

1.5.2 Conclusions

Keeping Figure 7 in mind, it becomes clear that the two methods results in the same separation of the classes. The multiclass LDA model yields two decision boundaries which can be interpreted as majority voting w.r.t. the discriminants, when distinguishing between the classes. The one-versus-one approach, on the other hand, yields three decision boundaries, which combined together results in the same predictions as the LDA (see Figure 7). The combination of the three decision boundaries could also be viewed, in this matter, as majority voting. Since we assume linearly separable classes, these results are not unexpected. Last but not least, it is exciting that the multiclass LDA is equivalent to the one-versus-one approach under these presumptions.

1.5.3 Source Code

Initially, we generate a new set of observations with label "3".

```
# generate another set of observations
set.seed(1337)
n3 = 150
mu3 = c(-1, -3)

z3 = SimDesign::rmvnorm(n3, mu3)
lab3 = rep(3, n3)
```

1.5.3.1 Part 1

Then, we visualize our three classes, one color for each.

```
plot(
    z1,
    col=2,
    xlim=c(-7, 4),
    ylim = c(-7, 6),
    xlab=expression('x'[1]),
    ylab=expression('x'[2]),
```

```
    pch=20
)
points(z2, col=3, pch=20)
points(z3, col=4, pch=20)
grid()
legend(
    "topleft",
    legend=c("Class 1", "Class 2", "Class 3"),
    pch=20,
    col=c(2, 3, 4),
    bg="white"
)
```

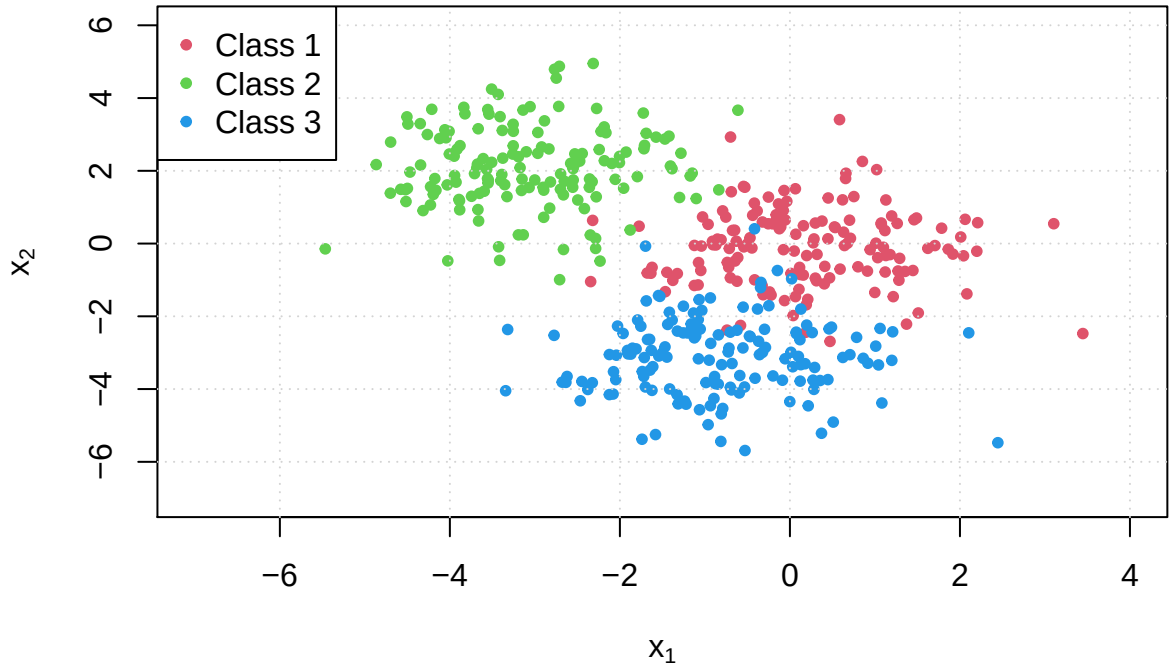


Figure 3: *Visualization of the three sets of simulated normal multivariate numbers.*

1.5.3.2 Part 2 In this part, we first stack our data into a new data frame, including all three classes.

```
# create data frame for all three classes
Z = rbind(z1, z2, z3)
Y = as.factor(c(lab1, lab2, lab3))
dat3 = as.data.frame(cbind(Z, Y))
```

Consequently, we fit a LDA model, using the built-in function.

```
# fit 3-class LDA
Mod = lda(
    Y ~ V1 + V2,
    data=dat3
)
```

1.5.3.3 Part 3

Then, we generate a "prediction grid" as requested.

```
# data frame
x1 = seq(-6, 4, 0.1)
x2 = seq(-6, 6, 0.1)
d = expand.grid(x1, x2)
names(d) = c("V1", "V2")
```

1.5.3.4 Part 4

The grid of points were then classified by the LDA model fitted in Part 2, and colored accordingly.

```
d_pred = predict(Mod, d)
```

1.5.3.5 Part 5

The separation of the grid, based on the previous LDA classification, is visualized by one color for each class.

```
# extract predicted points and label according to class
d1_idx = d_pred$class == 1
d2_idx = d_pred$class == 2
d3_idx = d_pred$class == 3

d1_pred = d[d1_idx, ]
d2_pred = d[d2_idx, ]
d3_pred = d[d3_idx, ]
```

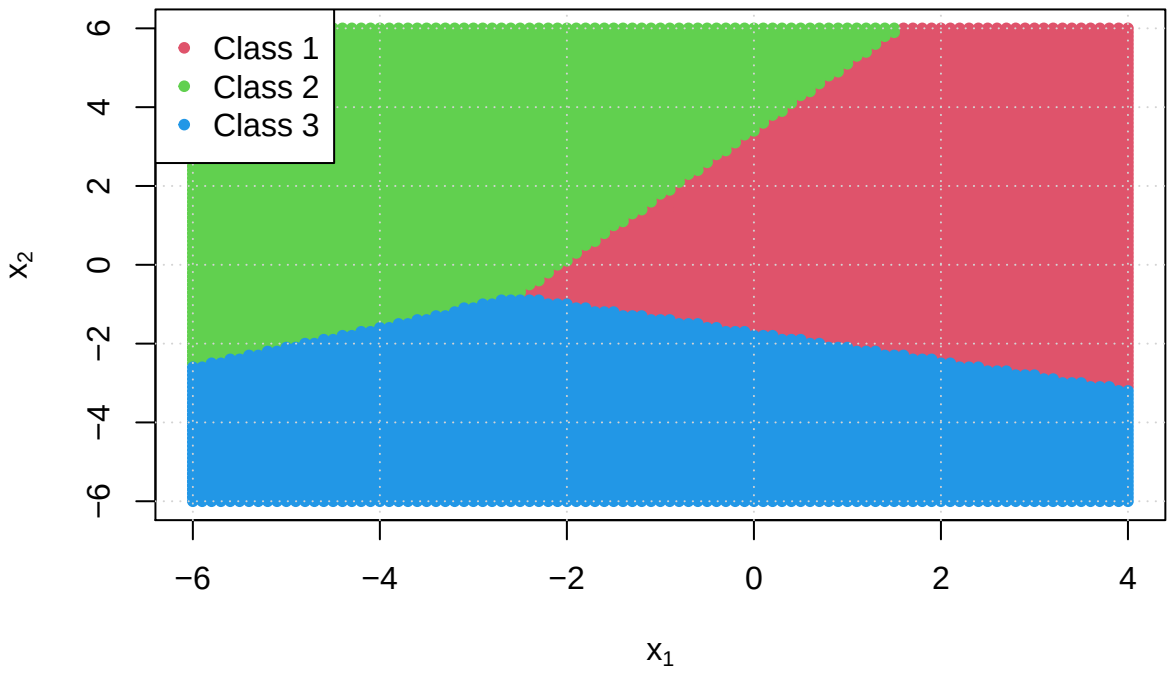


Figure 4: *Visualization of the predictions of the generated grid (in Part 3) based on the classifications in Part 4.*

Further, the two decision boundaries from the LDA model in Part 2 was visualized, along with the initial observations of the three classes. Solely to get a deeper understanding of the model.

1.5.3.6 Part 6 We fit another LDA based on MLE as in Task 2, but this time using an one-versus-one approach, which results in three different decision boundaries.

```
mu1_s = colMeans(z1)
mu2_s = colMeans(z2)
mu3_s = colMeans(z3)
S = pooled.cov(rbind(z1, z2, z3), c(lab1, lab2, lab3)) # pooled covariance
```

```
MU_pos = mu1_s + mu2_s
dim(MU_pos) = c(2, 1)

MU_neg = mu2_s - mu1_s
dim(MU_neg) = c(2, 1)

c = (1 / 2) * t(MU_pos) %*% inv(S) %*% MU_neg
x = t(MU_neg) %*% inv(S)

k1 = x[1] / -x[2]
m1 = c / x[2]
```

```
MU_pos = mu1_s + mu3_s
dim(MU_pos) = c(2, 1)

MU_neg = mu3_s - mu1_s
dim(MU_neg) = c(2, 1)

c = (1 / 2) * t(MU_pos) %*% inv(S) %*% MU_neg
x = t(MU_neg) %*% inv(S)

k2 = x[1] / -x[2]
m2 = c / x[2]
```

```
MU_pos = mu2_s + mu3_s
dim(MU_pos) = c(2, 1)

MU_neg = mu3_s - mu2_s
dim(MU_neg) = c(2, 1)

c = (1 / 2) * t(MU_pos) %*% inv(S) %*% MU_neg
x = t(MU_neg) %*% inv(S)

k3 = x[1] / -x[2]
m3 = c / x[2]
```

Lastly, the one-versus-one decision boundaries along with the prediction grid was plotted, in order to understand the relationship between the different approaches, i.e. one-versus-one and multiclass LDA, respectively.

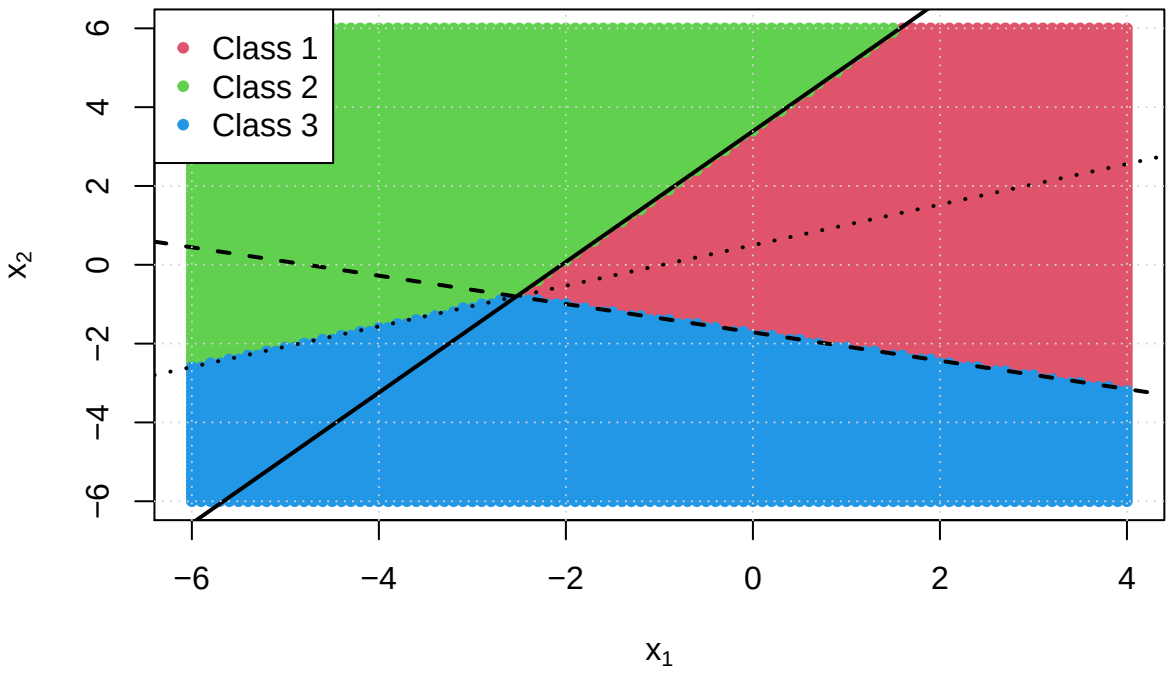


Figure 5: *The decision boundaries using the one-versus-one approach (Part 6), along with the predictions of the generated grid (Part 3).*