Special Tasks Lab 2

Mim Kemal Tekin

12/7/2018

Special Task 3

Task 3.1

Sometimes it can be interesting to see a decision boundary between classes, and lda() function does not provide this information directly. Thus, implement LDA with proportional priors, inputs RW and CL and output Sex for this data yourself (use only basic R functions), classify the observations and extract the discriminant functions and equation of the decision boundary.

We can define the LDA classifier function as following:

$$P(Y = c_i|x) = \text{softmax}(x\beta_i^T + \beta_{0i})$$

and β coefficients are changing for each class, they are defined as following:

$$\beta_{0i} = -\frac{1}{2}\mu_i^T \sum^{-1} \mu_i + log(\pi_i)$$

$$\beta_i = \sum^{-1} \mu_i$$

where π_i (prior) is proportions of classes in data, and

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} x_i$$

$$\hat{\Sigma}_c = \frac{1}{N_c} \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$$

$$\hat{\Sigma} = \frac{1}{N} \sum_{c=1}^{k} N_c \hat{\Sigma}_c$$

In my calculations, $\hat{\mu}_c$ is a matrix [2x2] which has columns are classes in order of ("Male", "Female") and rows are features ("RW", "CL"); $\hat{\Sigma}$ is another matrix [2x2]; in this task x (data) is a matrix [Nx2].

We can define the discriminant function as following:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \Sigma^{-1} \mu_k + log(\pi_i)$$

We used this function to classify our observations. We calculate 2 class discriminants for one observation and we classified the observation by selecting the class which has bigger discriminant.

We have two class in this task. And we calculate 2 discriminants for those classes. If we made equal this discriminant for one observation we can get the point of the decision boundary. And we can extract our decision boundary function by using following equation where 1: Male, 2: Female:

$$x\beta_1 + \beta_{01} = x\beta_2 + \beta_{02}$$

$$x(\beta_1 - \beta_2) = \beta_{02} - \beta_{01}$$

where only for one observation of data, x is [1x2] matrix and has x_1 and x_2 for features which are ("RW", "CL"). The target feature is "CL" and we will solve this equation for x_2. With this way we find "CL" value for a specific "RW" on the boundary line. After some calculations we get x_2 as following:

$$x_2 = \frac{\beta_{02} - \beta_{01} - x_1(\beta_{11} - \beta_{21})}{\beta_{12} - \beta_{22}}$$

Task 3.2

Make a plot of the original data australian-crabs.csv coloured by the classification label obtained and plot also the decision boundary. Comment on the quality of fit.

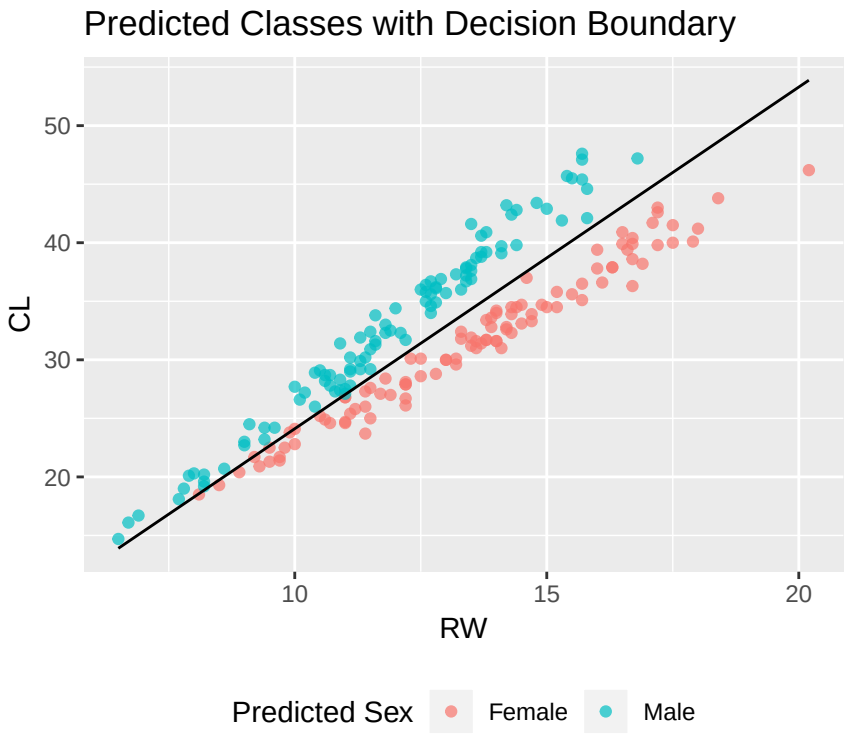
Table 1: Confusion Matrix

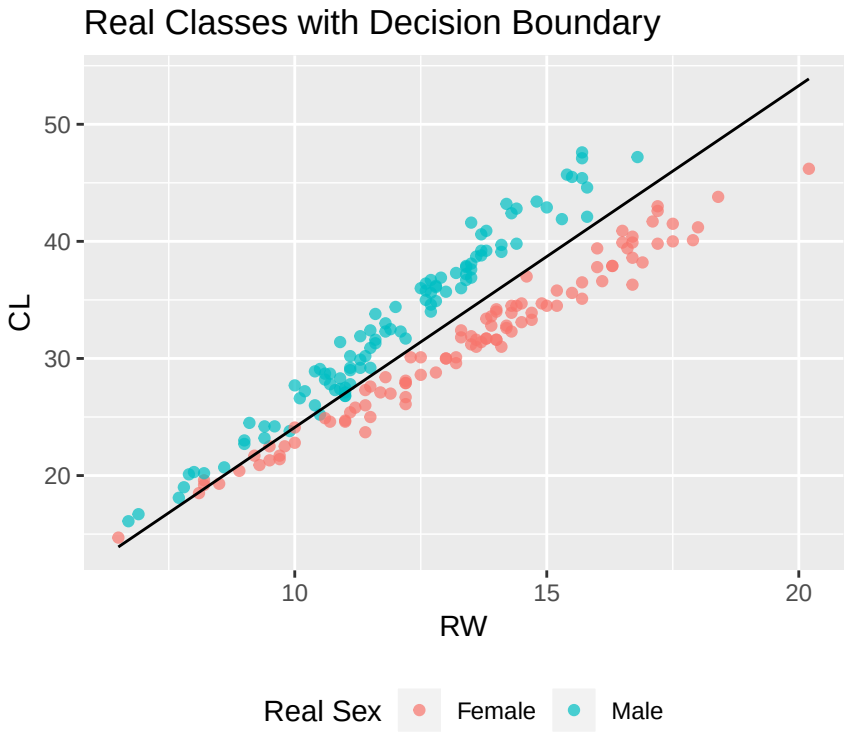
	Female	Male	Frequencies
Female	97	3	100
Male	4	96	100
Frequencies	101	99	200

Table 2: Misclassification Rate

Misclassification
0.035

As we can see misclassification rate is so small, we can say that quality of fit is great. Furthermore, if we examine two plot below, we can say the boundry is also good located and seperate the data well for 2 classes. There are only some misclassifications at low RW and CL levels.


Predicted Classes with Decision Boundary
CL
RW
50
40
30
20
10
15
20
Predicted Sex
Female
Male


Real Classes with Decision Boundary
CL
RW
50
40
30
20
10
15
20
Real Sex
Female
Male

Appendix

```
knitr::opts_chunk$set(echo = FALSE, fig.width = 4.5, fig.height = 4,
                      fig.align = "center")

library(kableExtra)

conf_matrix = function(real_data, predicted_data){
  # make the values factor
  real_data = as.factor(real_data)
  predicted_data = factor(predicted_data, levels = levels(real_data))
  # we can have "not predicted" values in predicted data
  # make equal the levels of predicted values with real data
  levels(predicted_data) = levels(real_data)

  ct = table(real_data, predicted_data)
  df = data.frame(c(as.vector(ct[,1]), sum(ct[,1])),
                  c(as.vector(ct[,2]), sum(ct[,2])),
                  c(sum(ct[1,]), sum(ct[2,]), sum(ct)))
  rownames(df) = c("Female", "Male", "Frequencies")
  colnames(df) = c("Female", "Male", "Frequencies")
  return(df)
}

kable_cm = function(cm, capture){
  return(kableExtra::kable(cm, "latex", booktabs = T, align = "c",
             caption = capture) %>%
             row_spec(2, hline_after = T) %>%
             column_spec(c(1,3), border_right = T) %>%
             kable_styling(latex_options = "hold_position"))
}

calculate_rate = function(conf_matrix){
  conf_matrix = as.matrix(conf_matrix)
  return(1 - sum(diag(conf_matrix[1:2,1:2]))/sum(conf_matrix[1:2,1:2]))
}

library(ggplot2)

df_crabs = read.csv("../dataset/australian-crabs.csv")
df_selected = df_crabs[, c("RW", "CL", "sex")]

# library(MASS)
# resLDA = lda(sex~RW+CL, data=df_selected, method="mle")

classes = unique(df_selected$sex)
class_count = length(classes)
feature_count = dim(df_selected)[2] - 1
n = nrow(df_selected)

n_c = rep(0, class_count)
# columns are classes, rows are features
mu_c = matrix(rep(0, class_count*feature_count), ncol = class_count)
```

```r
cov_c = list()
prior = rep(0, class_count)

for(cl in 1:class_count){
  # select current class in the data
  data_c = df_selected[df_selected$sex == classes[cl], 1:feature_count]
  # store size of current class
  n_c[cl] = nrow(data_c)
  # calculate prior
  prior[cl] = n_c[cl] / nrow(df_selected)
  # calculate means of all features
  mu_c[, cl] = colSums(data_c) / n_c[cl]
  # calculate variations of all features for the class
  cov_c[[cl]] = cov(data_c)
}

# calculate pool cov matrix
cov_mat = (n_c[1]*cov_c[[1]] + n_c[2]*cov_c[[2]]) / (n)

# calculate beta [2x2]
# column1: Beta_Male
# column2: Beta_Female
beta_c = solve(cov_mat) %*% mu_c

# calculate gamma_c
# column1: Beta_Male
# column2: Beta_Female
gamma_c = c()
gamma_c[1] = t(mu_c[,1]) %*% solve(cov_mat) %*% mu_c[,1] / (-2) + log(prior[1])
gamma_c[2] = t(mu_c[,2]) %*% solve(cov_mat) %*% mu_c[,2] / (-2) + log(prior[2])

# get data as matrix without target var
x = as.matrix(df_selected[, 1:feature_count] )
n = nrow(x)
# calculate delta
delta = x %*% beta_c + matrix(rep(gamma_c, n), ncol=2, byrow = T)
# classify from delta
preds = ifelse(delta[,1]>delta[,2], "Male", "Female")
# get z (decision boundry y values)
# in this case I have different indexes for elements from the formula
# because I set dimensions in the code other way around
num = gamma_c[2] - gamma_c[1] - x[,1] * (beta_c[1,1] - beta_c[1,2])
denum = beta_c[2,1] - beta_c[2,2]
z = num / denum

df_selected$pred = preds
df_selected$bound_y = z

# plot for the predictions
# with boundary
plot_prediction = ggplot(df_selected) +
  geom_point(aes(x = RW, y = CL, color = pred), alpha = 0.7) +
  geom_line(aes(x = RW, y = bound_y)) +
```

```
  labs(title = "Predicted Classes with Decision Boundary",
       color = "Predicted Sex") +
  theme(legend.position = "bottom")

# plot for the original classlabels
# with boundary
plot_org = ggplot(df_selected) +
  geom_point(aes(x = RW, y = CL, color = sex), alpha = 0.7) +
  geom_line(aes(x = RW, y = bound_y)) +
  labs(title = "Real Classes with Decision Boundary",
       color = "Real Sex") +
  theme(legend.position = "bottom")


cm = conf_matrix(df_selected$sex, df_selected$pred)

kable_cm(cm, "Confusion Matrix")

kableExtra::kable(data.frame(Misclassification = calculate_rate(cm)),
                  "latex", booktabs = T, align = "c",
                  caption = "Misclassification Rate") %>%
            kable_styling(latex_options = "hold_position")
plot_prediction
plot_org
```