Bayesian Classifier

Jaime Díaz-Trechuelo Sánchez-Moliní

2024-01-22

The Bayes' Classifier Regions

This part of the code gives the `BayesClassifier` function which asks for the means and variances of two bivariate normal distributions. It then represents the 2-dimensional critical regions associated with these random variables. Mathematically we have: Let X and Y be two random variables, 2 and 1 dimensional respectively s.t.

$$(X|Y=k) \sim \mathcal{N}_k(\mu_k, \Sigma_k) \quad k = 1, 2$$

Our objective is to give a rule so that given an observation of X, noted X_i we'll infere wether X_i is associated to $Y = 1$ or $Y = 2$. One way to do this is with the Bayes' Classifier.

Let's now go over the parameters of this function:

`mu11` is the value of $(\mu_1)_1$

`mu12` is the value of $(\mu_1)_2$

`mu21` is the value of $(\mu_2)_1$

`mu22` is the value of $(\mu_2)_2$

`S11` is the value of $(\Sigma_1)_{11}$

`S12` is the value of $(\Sigma_1)_{22}$

`S1D` is the value of $(\Sigma_1)_{12} = (\Sigma_1)_{21}$

`S21` is the value of $(\Sigma_2)_{11}$

`S22` is the value of $(\Sigma_2)_2$

`S2D` is the value of $(\Sigma_2)_{12} = (\Sigma_2)_{21}$

If any of this values isn't specified then function assumes that the means are 0 and the variances are I_2 `margins` is a 4 element vector with the limits of the graphic window to be created in the order $(x_{1,min}, x_{1,max}, x_{2,min}, x_{2,max})$. By default, if omitted, `c(-5,5,-5,5)` is taken. `labels`is a boolean variable which is set to `TRUE` by default, it determines wether or not the regions are labelled R_1, R_2.

```r
BayesClassifier <-
function(mu11=0,mu12=0,mu21=0, mu22=0, S11=1,S12=1,S1D=0,S21=1,S22=1,S2D=0, margins = c(-5,5,-5,5), labe
  mu1 <- c(mu11, mu12)
  mu2 <- c(mu21, mu22)
  Sigma1 <- matrix(c(S11, S1D, S1D, S12), nrow = 2, byrow = TRUE)
  Sigma2 <- matrix(c(S21, S2D, S2D, S22), nrow = 2, byrow = TRUE)

  if (all(Sigma1==Sigma2)) {
    b <- inv(Sigma1) %*% (mu1 - mu2)
    region_condition <- function(x, y) {
```

```
    v <-matrix(c(x, y), ncol = 1)
    return(t(v) %*% b > 0.5 * t((mu1 + mu2)) %*% b)
  }} else {
    B <- inv(Sigma1) - inv(Sigma2)
    b <- inv(Sigma1) %*% mu1 - inv(Sigma2) %*% mu2
    k <- 0.5 * (log(det(Sigma2) / det(Sigma1)) + t(mu2) %*% inv(Sigma2) %*% mu2 - t(mu1) %*% inv(Sigma
    region_condition <- function(x, y) {
      v <- matrix(c(x, y), ncol = 1)
      return(-0.5 * t(v) %*% B %*% v + t(v) %*% b > -k)
    }
  }

x <- seq(margins[1], margins[2], length.out = 350)
y <- seq(margins[3], margins[4], length.out = 350)
data <- expand.grid(x = x, y = y)
in_region = c()
for (j in 1:length(x)) {
  for (i in (1:length(y))){
    in_region[350*(i-1)+j] <- region_condition(x[j],y[i])
  }}
data$in_region <- in_region


plot <- ggplot(data[,1:2], aes(x, y, fill = in_region)) +
  geom_tile() +
  scale_fill_manual(values = c(rgb(1, 0, 0, alpha = 0.4), rgb(0, 0, 1, alpha = 0.4)), na.value = "whi
  ggtitle("Critical Regions") +
  theme(legend.position = "none") +
  ylab(TeX("$X_2$")) + xlab(TeX("$X_1$"))

if (labels) {
  x0 <- 0
  y0 <- 0
  if (region_condition(x0, y0)) {
    plot <- plot + annotate("text", x = 0, y = 0, label = TeX("$R_1$"), parse = TRUE, col = "blue", si
    indices <- in_region == FALSE
    coordinatex <- (data[indices,1:2])[5000,1]
    coordinatey <- (data[indices,1:2])[5000,2]
    plot <- plot + annotate("text", x = coordinatex , y = coordinatey, label = TeX("$R_2$"), parse =
    } else {
    plot <- plot +
      annotate("text", x = 0, y = 0, label = TeX("$R_2$"), parse = TRUE, col = "red", size = 12)
     indices <- in_region == TRUE
    coordinatex <- (data[indices,1:2])[5000,1]
    coordinatey <- (data[indices,1:2])[5000,2]
    plot <- plot + annotate("text", x = coordinatex , y = coordinatey, label = TeX("$R_1$"), parse =
  }
}
plot}
```

Bayesian Classifier for data sets

In this case we are given a data set of n observations of a variable that follows what we laid out before. The function `BayesClassifierForSets` returns a plot were the points or the dataset are colored based on where they'd be classified.

All parameters are the same as in the previous function, with the same defaults. The only changes are that there's no `margins` parameter and there is a parameter `observed_data` which is the data set which we want to classify, there's no default value for this parameter.

```r
BayesClassifierForSets <- function(collected_data,mu11=0,mu12=0,mu21=0, mu22=0, S11=0,S12=0,S1D=0,S21=0
  (mu1 <- c(mu11, mu12))
  mu2 <- c(mu21, mu22)
  Sigma1 <- matrix(c(S11, S1D, S1D, S12), nrow = 2, byrow = TRUE)
  Sigma2 <- matrix(c(S21, S2D, S2D, S22), nrow = 2, byrow = TRUE)
  if (all(Sigma1==Sigma2)) {
    b <- inv(Sigma1) %*% (mu1 - mu2)
    region_condition <- function(x, y) {
      v <-matrix(c(x, y), ncol = 1)
      return(t(v) %*% b > 0.5 * t((mu1 + mu2)) %*% b)
    }} else {
      B <- inv(Sigma1) - inv(Sigma2)
      b <- inv(Sigma1) %*% mu1 - inv(Sigma2) %*% mu2
      k <- 0.5 * (log(det(Sigma2) / det(Sigma1)) + t(mu2) %*% solve(Sigma2) %*% mu2 - t(mu1) %*% solve(S
      region_condition <- function(x, y) {
        v <- matrix(c(x, y), ncol = 1)
        return(-0.5 * t(v) %*% B %*% v + t(v) %*% b > -k)
      }
    }
  in_region = c()
  for (i in 1:length(collected_data[,1])) {
    in_region[i] <- region_condition(collected_data[i,1],collected_data[i,2])
  }
  collected_data<- cbind(collected_data,(abs(in_region-2)))
  indices_1 <-collected_data[ , 3] == 1
  indices_2 <- collected_data[ , 3] == 2
  xlim <- c(min(collected_data[,1])-1,max(collected_data[,1])+1)
  ylim <- c(min(collected_data[,2])-1,max(collected_data[,2])+1)
  plot(collected_data[indices_1,1],collected_data[indices_1,2], col = 'blue',ylab = TeX("$X_2"), xlab =
  points(collected_data[indices_2,1],collected_data[indices_2,2], col = 'red')
  title("Classification of the given points")
  if (labels) {legend("topright", legend = c(TeX("Points in $R_1$"), TeX("Points in $R_2$")), col = c('
  return(collected_data)}
```