(8) Prädiktive Modellierung

Ziel dieser Sitzuung ist es, das Vorgehen typischer Analysen der prädiktiven Modellierung auf ihrer Implementierungsebene zu verdeutlichen. Dafür wollen wir das Prinzip der n-fachen *leave-one-out cross-validation* mithilfe eines "Nearest-Neighbour-Klassifikationsansatzes" verdeutlichen: Zu zwei Klassen zweidimensionaler Trainingsfeaturevektoren bestimmen wir jeweils das Stichprobenmittel und die Stichprobenkovarianz und ordnen einen zweidimensionalen Testfeaturevektor dann derjenigen Klasse zu, deren Stichprobenmittel es im Sinne der Mahalanobis-Distanz in Bezug zur jeweiligen Stichprobenkovarianzmatrix näher liegt.

Datensatz generieren

Wir erstellen zunächst einen simulierten Datensatz mit $n = 40$ Datenpunkten, wobei je die Hälfte der Datenpunkte auf eine Klasse entfallen und jeder Datenpunkt aus einer bivariaten Normalverteilung mit klassenspezifischen Erwartungswertparameter und klassenspezifischem Kovarianzmatrixparameter kommt.

```r
# R-Paket
library(mvtnorm)                                                    # multivariate Normalverteilung
set.seed(0)                                                         # reproduzierbare Ergebnisse

# Modellparameter
m       = 2                                                         # Featurevektordimension
n       = 40                                                        # Anzahl Trainingsdatenpunkte
mu_0    = c(1,1)                                                    # w.a.u. Erwartungswertparameter von Klasse 0
mu_1    = c(2,2)                                                    # w.a.u. Erwartungswertparameter von Klasse 1
Sigma_0 = matrix(c( 0.5, -0.3,                                      # w.a.u. Kovarianzmatrixparameter von Klasse 0
                   -0.3,  0.5),
                 byrow = TRUE,
                 nrow  = m)
Sigma_1 = matrix(c( 1.0,  0.5,                                      # w.a.u. Kovarianzmatrixparameter von Klasse 1
                    0.5,  1.0),
                 byrow = TRUE,
                 nrow  = m)

# Datensimulation
y      = matrix(rep(NaN,n)  , nrow = 1)                             # Labeldatenarray
x      = matrix(rep(NaN,n*m), nrow = m)                             # Featurevektorarray
for(i in 1:n){                                                      # Iteration über Datenpunkte

    # klassenabhängige Datengeneration
    if(i <= n/2){
        y[i]  = 0                                                   # Label
        x[,i] = rmvnorm(1, mu_0, Sigma_0)                           # Featurevektor
    } else {
        y[i]  = 1                                                   # Label
        x[,i] = rmvnorm(1, mu_1, Sigma_1)                           # Featurevektor
    }

}
D       = rbind(x,y)                                                # Datensatz konkatenieren
fname   = "Prädiktive_Modellierung.csv"                             # Dateiname
write.csv(D, file = fname, row.names = FALSE)                       # Datenspeichern
```

Mit folgendem **R**-Code visualisieren wir den Datensatz zusammen mit den klassenpezifischen Stichprobenmitteln und Stichprobenkovarianzmatrizen visualisieren. Das Ergebnis wird in Abbildung 1 dargestellt.

```
# Laden einer m x n Datenmatrix eines Datensatzes mit binären Klassen {0,1}
D       = read.csv("Prädiktive_Modellierung.csv")                    # Datensatz
X       = as.matrix(D[1:2,])                                         # Featurevektoren beider Klassen
m       = nrow(X)                                                    # Featurevektordimension
y       = as.vector(D[3,])                                           # Label beider Klassen
L       = c(0,1)                                                     # Klassenlabels

# klassenspezfische Deskriptivstatisken
x_bar   = matrix(rep(NaN,m*2)  , nrow = 2)                           # m x 2     Array für klassenspezifische Stichprobenmittel
C       = array(rep( NaN,m*m*2), dim = c(m,m,2))                     # m x m x 2 Array für klassenspezifische Stichprobenkovarianzmatrizen
for (l in L){
    Xl              = X[,y == l]                                     # x^{(i)} für Label l
    n               = ncol(Xl)                                       # Anzahl Datenvektorealisierungen
    I_n             = diag(n)                                        # Einheitsmatrix I_n
    J_n             = matrix(rep(1,n^2), nrow = n)                   # 1_{nn}
    x_bar[,l+1]     = (1/n)* Xl %*% J_n[,1]                          # Stichprobenmittel
    C[ , , l+1]     = (1/(n-1))*(Xl %*% (I_n-(1/n)*J_n) %*% t(Xl))   # Stichprobenkovarianzmatrix
}

# Abbildungsparameter
library(latex2exp)
par(
    family      = "sans",
    mfcol       = c(1,1),
    pty         = "s",
    bty         = "l",
    lwd         = 1,
    las         = 1,
    mgp         = c(2,1,0),
    xaxs        = "i",
    yaxs        = "i",
    font.main   = 1,
    cex         = 1,
    cex.main    = 1)

# Gaussian Isokontur Paket
library(ellipse)
iso_0       = ellipse(C[,,1], level = 0.50, centre = x_bar[,1])
iso_1       = ellipse(C[,,2], level = 0.50, centre = x_bar[,2])

# Klasse 0
plot(iso_0,
    type = "l",
    col = "Red",
    xlim = c(0,4),
    ylim = c(0,4),
    xlab = TeX("$x_1$"),
    ylab = TeX("$x_2$"))
points(x_bar[1,1], x_bar[2,1],
    col  = "Red",
    pch  = 3,
    cex  = 1.5)

# Klasse 1
lines(iso_1[,1], iso_1[,2],
    col  = "Blue")
points(x_bar[1,2], x_bar[2,2],
    col  = "Blue",
    pch  = 3,
    cex  = 1.5)

# Daten y^{(i)} = 0
points(D[1,D[3,] == 0], D[2,D[3,] == 0],
    col  = "White",
    bg   = "Red",
    pch  = 21)

# Daten y^{(i)} = 1
points(D[1,D[3,] == 1], D[2,D[3,] == 1],
    col  = "White",
    bg   = "Blue",
    pch  = 21)

# Legende
legend("topleft", c("y = 0", "y = 1"),
    pch         = 16,
    col         = c("Red", "Blue"),
    bty         = "n",
    cex         = 1,
    x.intersp   = 1)

# Speicherung
```

```
dev.copy2pdf(
    file        = "./Abbildungen/datensatz.pdf",
    width       = 4.5,
    height      = 4.5)
dev.off()
```

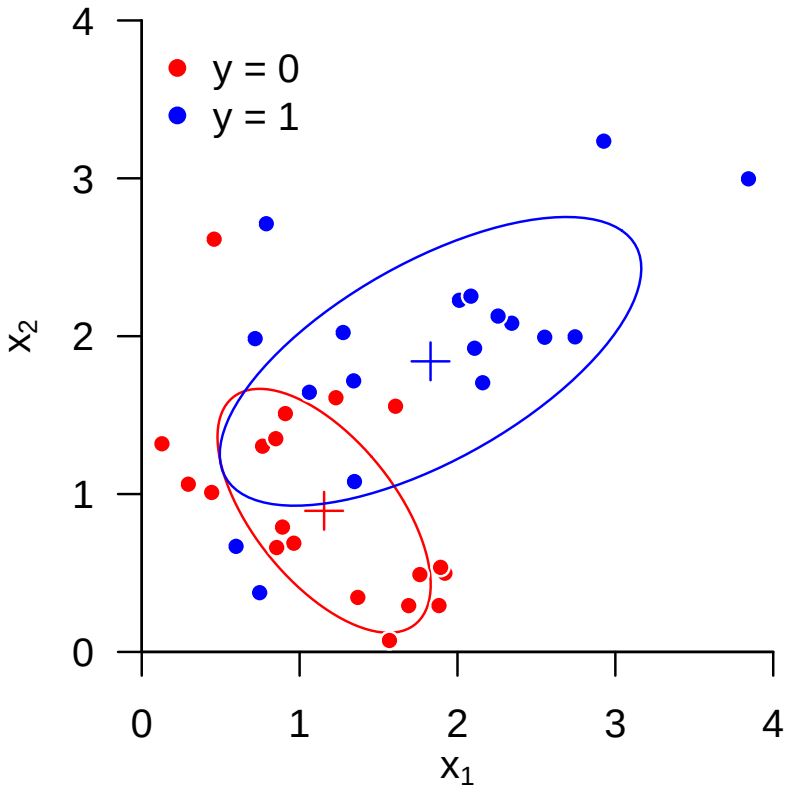

Abbildung 1. Beispieldatensatz mit Datenpunkten aus zwei Klassen $y = 0$ (rot) und $y = 1$ (blau).

n-fache leave-one-out Kreuzvalidierung

Mit folgendem **R**-Code führen wir im Sinne des oben beschriebenen Klassifizierungsansatzes "Nearest Neighbour" eine n-fache *leave-one-out cross-validation* durch.

```
# Datensatz
D       = read.csv("Prädiktive_Modellierung.csv")                    # Datensatz
L       = c(0,1)                                                     # Klassenlabels
x       = as.matrix(D[1:2,])                                         # Featurevektoren
y       = as.matrix(D[3,])                                           # Label
n       = ncol(y)                                                    # Anzahl Datenpunkte
y_pred  = matrix(rep(NaN,n*2), nrow = n)                             # Array wahrer und prädizierter Label

# n-fache leave-one-out cross-validation
for(i in 1:n){

    # Datensatzpartition
    x_train     = as.matrix(x[,-i])                                  # iter Featurevektor nicht im Trainingsdatendatz
    y_train     = as.matrix(y[,-i])                                  # ites Label nicht im Trainingsdatendatz
    x_test      = as.matrix(x[, i])                                  # iter Featurevektor als Testdatenpunkt
    y_test      = as.matrix(y[, i])                                  # ites Label als Testdatenpunkt
    y_pred[i,1] = y_test                                             # wahres Label des iten Labels

    # klassenspezfische Mahalanobisdistanz ("Training")
    D           = matrix(rep(NaN,2), ncol = 2)                       # 1 x 2 Array für klassenspezifische Mahalanobisdistanzen
    for (l in L){                                                    # Iteration über Klassen
        Xl      = x_train[,y_train == l]                             # x^{(i)} für Label l
        nl      = ncol(Xl)                                           # Anzahl Datenvektorealisierungen
        I_nl    = diag(nl)                                           # Einheitsmatrix I_n
        J_nl    = matrix(rep(1,nl^2), nrow = nl)                     # 1_{nn}
        x_bar   = (1/nl)* Xl %*% J_nl[,1]                            # Stichprobenmittel
        C       = (1/(nl-1))*(Xl %*% (I_nl-(1/nl)*J_nl) %*% t(Xl))   # Stichprobenkovarianzmatrix
        D[l+1]  = t(x_test - x_bar) %*% solve(C) %*% (x_test - x_bar) # Mahalanobisdistanz
    }
    # Prädiktion des iten Labels ("Test")
    if (D[1] <= D[2]){y_pred[i,2] = 0} else {y_pred[i,2] = 1}        # prädiziertes Label

}

# Evaluation der leave-one-out cross-validation
rp      = sum(y_pred[y_pred[,1] == 1,2] == 1)                        # Anzahl richtig positiver Prädiktionen (1,1)
rn      = sum(y_pred[y_pred[,1] == 0,2] == 0)                        # Anzahl richtig negativer Prädiktionen (0,0)
fp      = sum(y_pred[y_pred[,1] == 0,2] == 1)                        # Anzahl falsch  positiver Prädiktionen (0,1)
fn      = sum(y_pred[y_pred[,1] == 1,2] == 0)                        # Anzahl falsch  positiver Prädiktionen (1,0)
ACC     = (rp+rn)/(rp+fp+rn+fn)                                      # Genauigkeit ("Accuracy")
SEN     = rp/(rp+fn)                                                 # Sensitivität
SPE     = rn/(rn+fp)                                                 # Spezifivität

# Ergebnisausgabe
cat(  "Accuracy    : ", ACC,
    "\nSensitivity : ", SEN,
    "\nSpecificity : ", SPE, "\n")
```

```
Accuracy    :  0.75
Sensitivity :  0.85
Specificity :  0.65
```

Wir erhalten für den Beispieldatensatz eine Genauigkeit von 0.75, eine Sensitivität von 0.85 und eine Spezifizität von 0.65.