(12) Logistische Regression

Ziel dieser Sitzung ist es, die logistische Regression für einen simulierten Beispieldatensatz auf der Implementationsebene nachzuvollziehen und die resultierenden Ergebnisse im Hinblick auf die darauf aufbauende Vorhersage darzustellen.

Modell visualisieren

Um die Mean-Funktion der logistischen Regression zu verstehen, generieren wir zunächst 30 Datenpunkte ($n = 30$) aus einem logistischen Regressionsmodell mit nur einer Prädiktorvariable ($m = 1$). Diese kontinuierliche Variable x_2 wird dabei zunächst in einen linearen Prädiktor η überführt, der wiederum mithilfe der logistischen Standardfunktion in einen Bernoulliparameter μ übersetzt wird, der schließlich der Wahrscheinlichkeit entspricht, dass ein Datenpunkt $y = 1$ realisiert wird.

```
# Modellparameter
m    = 1                                # Featurevektorendimensionalität
n    = 30                               # Anzahl Datenpunkte
x    = matrix(c(rep(1, n),              # erweiterte Featurevektoren
                seq(-5, 5, len = n)),
              nrow  = 2,
              byrow = TRUE)
beta = matrix(c(-2,2), nrow = 2)        # wahrer, aber unbekannter Parametervektor
eta  = t(x) %*% beta                    # wahrer, aber unbekannter linearer Prädiktor
mu   = 1/(1+exp(-eta))                  # wahrer, aber unbekannter Bernoulliparametervektor

# Dategeneration
set.seed(2)                             # reproduzierbare Ergebnisse
y    = rep(NaN,2)                       # Datenarray
for(i in 1:n){
    y[i] = rbinom(1, 1, mu[i])          # Bernoullivariablenrealisierung
}
```

Wir visualisieren diesen simulierten Beispieldatensatz in Abbildung 1 mithilfe des folgenden **R**-Codes.

```
# Abbildungsparameter
library(latex2exp)
par(
    family      = "sans",
    mfcol       = c(1,2),
    pty         = "m",
    bty         = "l",
    lwd         = 1,
    las         = 1,
    mgp         = c(2,1,0),
    xaxs        = "i",
```

```
    yaxs        = "i",
    font.main   = 1,
    cex         = 1.1,
    cex.main    = 1.2)

# linearer Prädiktor \eta
plot(x[2,], eta,
    type = "b",
    pch  = 16,
    xlab = TeX("$x_2$"),
    ylab = "",
    xlim = c(min(x[2,])-1, max(x[2,])+1),
    ylim = c(min(eta)-1, max(eta)+1),
    main = TeX("$\\eta = x^T \\beta$"))
grid()

# Bernoulli-Parameter \mu
plot(x[2,], mu,
    type = "b",
    xlab = TeX("$x_2$"),
    ylab = "",
    col  = "gray60",
    pch  = 16,
    xlim = c(min(x[2,])-1, max(x[2,])+1),
    ylim = c(-.1,1.1),
    main = TeX("$\\mu = 1/(1 + exp(-\\eta))$ und $y$"))

# simulierte Daten y
points(x[2,], y,
    type = "p")
grid()

# Legende
legend("topleft", c("y", TeX("$\\mu$")),
    pch         = c(1,16),
    col         = c("Black","gray60"),
    bty         = "n",
    cex         = 1,
    x.intersp   = 1,
    y.intersp   = 2)

# Speicherung
dev.copy2pdf(
    file        = "./Abbildungen/lr_datengeneration.pdf",
    width       = 10,
    height      = 5)
dev.off()
```

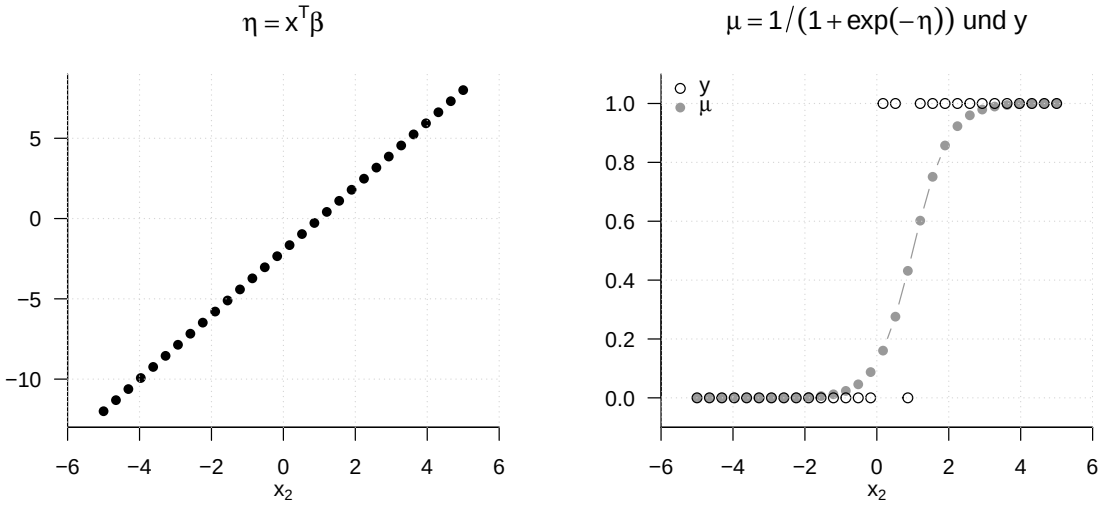

Abbildung 1. Einfaches logistisches Regressionsmodell mit kontinuierlicher Prädiktorvariable x_2, linearem Prädiktor η (links) und Bernoulliparameter μ (rechts).

Kreuzvalidierte Modellschätzung

Im Anschluss wenden wir uns einem Datensatz zu, der durch die Prädiktorvariablen `BDI` und `GLU` ($m = 2$) sowie die abhängige Variable `RES` ($y = 1$: Therapie-Non-Responder) mit 60 Datenpunkten ($n = 60$) gekennzeichnet ist.

Folgender **R**-Code führt dann mithilfe des in der Vorlesung besprochenen Verfahrens *Iteratively Weighted Least Squares* (IWLS) im Rahmen die **R**-Funktion `glm` eine kreuzvalidierte Schätzung der Parameter der logistischen Regression durch, wobei das Schema der *leave-one-out*-Kreuzvalidierung angewendet wird.

```
D       = read.csv("Logistische_Regression.csv")          # Datensatz
K       = nrow(D)                                         # Anzahl Kreuzvalidierungsschritte
p_y     = matrix(rep(NaN, K)  , nrow = 1)                 # p(y = 1)
y_pred  = matrix(rep(NaN, K*2), nrow = K)                 # Prädiktionsarray
for(k in 1:K){                                            # K-fold LOOCV
    x_train      = t(D[-k,1:2])                           # Trainingsdatensatzfeatures
    y_train      = t(D[-k,3  ])                           # Trainingsdatensatzlabels
    x_test       = t(D[ k,1:2])                           # Testdatensatzfeaturevektor
    y_pred[k,1]  = t(D[ k,3])                             # Testdatensatzlabel (wahr)
    n            = ncol(x_train)                          # n
    m            = nrow(x_train)                          # m
    lr           = glm(t(y_train) ~ t(x_train), family = 'binomial')  # IWLS-Parameterlernen
    beta_hat     = as.matrix(lr$coefficients, nrow = m + 1)           # Parameterschätzer
    x_test_tilde = rbind(1, x_test)                       # erweiteter Featurevektor
    p_y[k]       = 1/(1+exp(-t(x_test_tilde) %*% beta_hat))           # p(y)
    y_pred[k,2]  = as.numeric(p_y[k] >= 0.5)              # Testdatensatzlabel (prädiziert)
}
```

Auf Grundlage der in jedem Kreuzvalidierungsschritt prädizierten Labels lassen sich Prädiktionsperformancemaße für die LDA-basierte Klassifikation bestimmen und ausgeben, wie aus folgendem **R**-Code hervorgeht.

```
rp      = sum(y_pred[y_pred[,1] == 1,2] == 1)                    # |(1,1)|
rn      = sum(y_pred[y_pred[,1] == 0,2] == 0)                    # |(0,0)|
fp      = sum(y_pred[y_pred[,1] == 0,2] == 1)                    # |(0,1)|
fn      = sum(y_pred[y_pred[,1] == 1,2] == 0)                    # |(1,0)|
ACC     = (rp+rn)/(rp+fp+rn+fn)                                  # Accuracy
SEN     = rp/(rp+fn)                                             # Sensitivity
SPE     = rn/(rn+fp)                                             # Specificity
cat(  "Accuracy   : ", ACC,                                      # Ergebnisausgabe
    "\nSensitivity: ", SEN,
    "\nSpecificity: ", SPE, "\n")
```

```
Accuracy   :  0.7166667
Sensitivity:  0.8235294
Specificity:  0.5769231
```

Ergebnisse visualisieren

Schließlich wollen wir die Ergebnisse der logistischen Regression veranschaulichen, indem wir im Raum der Prädiktorvariablen (`BDI`, `GLU`) die Wahrscheinlichkeit berechnen, dass ein hypothetischer Datenpunkt als Therapie-Non-Responder ($y = 1$) klassifiziert wird

```
# Modellschätzung
D         = read.csv("Logistische_Regression.csv")               # Datensatz
x         = t(D[,1:2])                                           # Featurevektoren
y         = t(D[,3])                                             # Label
n         = ncol(x)                                              # n
m         = nrow(x)                                              # m
lr        = glm(t(y_train) ~ t(x_train), family = 'binomial')    # IWLS-Parameterlernen
beta_hat  = as.matrix(lr$coefficients, nrow = m + 1)             # Parameterschätzer

# Prädiktorraum
x_min     = 0                                                    # GLU/BDI-Minimum
x_max     = 3                                                    # GLU/BDI-Maximum
x_res     = 5e2                                                  # GLU/BDI-Auflösung
bdi       = seq(x_min, x_max, length.out = x_res)                # BDI
glu       = seq(x_min, x_max, length.out = x_res)                # GLU

# Prädiktion
p_y       = matrix(rep(NaN, x_res*x_res), nrow = x_res)          # p_{(BDI,GLU}(y=1)
for(i in 1:x_res){                                               # BDI-Iterationen
    for(j in 1:x_res){                                           # GLU-Iterationen
        x_tilde   = rbind(1, bdi[i], glu[j])                     # \tilde{x}
        p_y[i,j]  = 1/(1+exp(-t(x_tilde) %*% beta_hat))          # p_{(BDI, GLU}(y=1)
    }
}
```

Schließlich visualisieren wir diese Wahrscheinlichkeiten in Abbildung 2 mithilfe des folgenden **R**-Codes.

```
# 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)

# Konturplot
filled.contour(bdi, glu, t(p_y[nrow(p_y):1, ncol(p_y):1]),
    col         = hcl.colors(20, "YlOrRd"),
    levels      = seq(0,1,len = 20),
    xlab        = "BDI",
    ylab        = "GLU",
    main        = "P(Non-Response)")

# Speicherung
dev.copy2pdf(
    file        = "./Abbildungen/lr_anwendungsbeispiel.pdf",
    width       = 5,
    height      = 5)
dev.off()
```

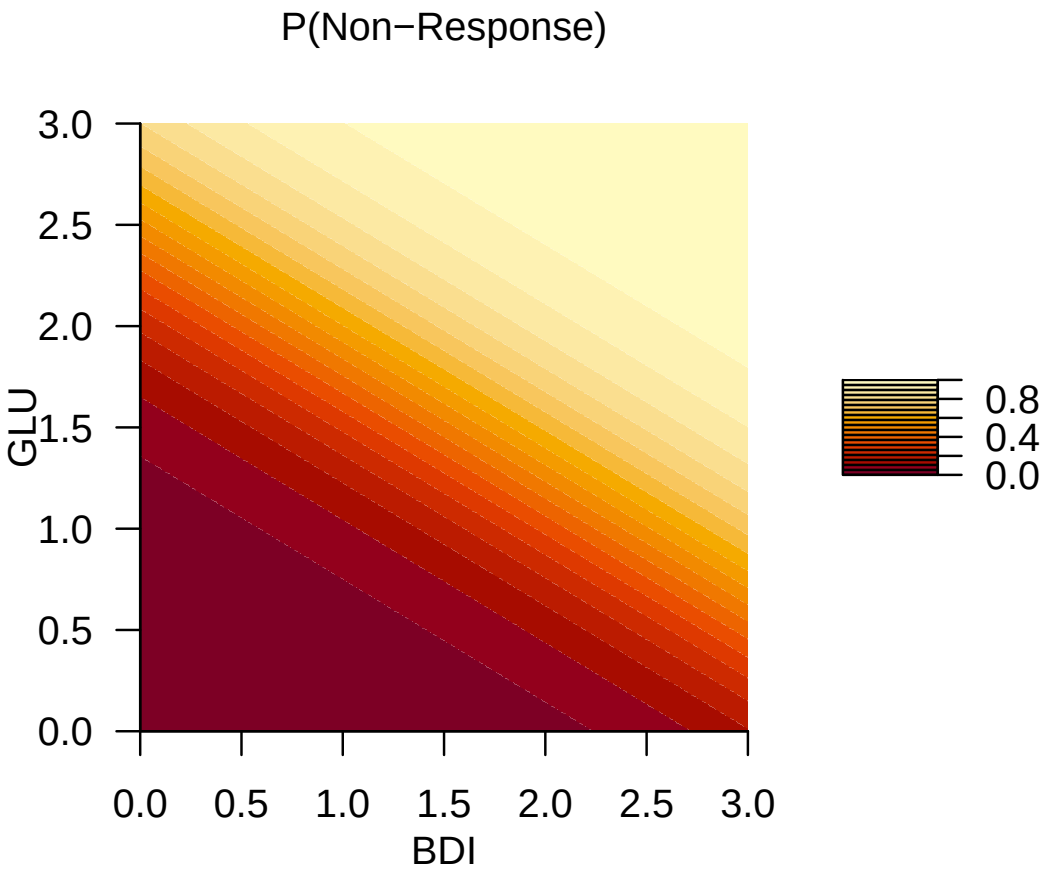

Abbildung 2. Non-Response-Wahrscheinlichkeit in Abhängigkeit von BDI und GLU.