(9) Nichtlineare Optimierung

Ziel dieser Sitzuung ist es, die Implementation eines Gradientenverfahrens basierend auf der in der Vorlesung vorgestellten Theorie nachzuvollziehen und in einem Anwendungsbeispiel zur anhand einer einfachen Funktion zu diskutieren.

Funktionsdefinition

Wir betrachten die multivariate reellwertige Funktion

$$f : \mathbb{R}^2 \to \mathbb{R}, x \mapsto f(x) := x_1^2 + x_2^2 \, . \tag{1}$$

Wie in Vorlesung gezeigt wurde, hat diese Funktion den Gradienten

$$\nabla f(x) := \begin{pmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} \tag{2}$$

und die Hesse-Matrix

$$\nabla^2 f(x) := \begin{pmatrix} \frac{\partial^2}{\partial x_1 \partial x_1} f(x) & \frac{\partial^2}{\partial x_1 \partial x_2} f(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1} f(x) & \frac{\partial^2}{\partial x_2 \partial x_2} f(x) \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \, . \tag{3}$$

Wir beginnen zunächst damit, dass wir in **R** Funktionen schreiben, die die multivariate reellwertige Funktion und ihren Gradienten berechnen.

```
# Funktion
f = function(x) {
    return(x[1]^2 + x[2]^2)              # f(x) := x_1^2 + x_2^2
}

# Gradient
nabla_f = function(x) {
    return(matrix(c(2*x[1],              # \nabla f(x) = (2x_1, 2x_2)^T
                    2*x[2]),
                  nrow = 2))
}
```

Gradientenbeispiele

Wir wollen nun den Gradienten der Funktion an einigen Stellen evaluieren, und zwar bei

$$x = \begin{pmatrix} 0.7 \\ 0.7 \end{pmatrix} \qquad x = \begin{pmatrix} -0.3 \\ 0.1 \end{pmatrix} \qquad x = \begin{pmatrix} -0.5 \\ -0.4 \end{pmatrix} \qquad x = \begin{pmatrix} 0.1 \\ -1.0 \end{pmatrix}$$

Dies ermöglicht uns folgender **R**-Code, der Abbildung 1 erzeugt.

```
# R-Pakete
library(latex2exp)

# Definitionsmengenraum
x_min  = -2                                          # x_i Minimum
x_max  =  2                                          # x_i Maximum
x_res  = 1e2                                         # x_i Auflösung
x_1    = seq(x_min, x_max, length.out = x_res)       # x_1 Raum
x_2    = seq(x_min, x_max, length.out = x_res)       # x_2 Raum
X      = expand.grid(x_1,x_2)                        # X = (x_1,x_2)^T Raum
fX     = matrix(as.matrix(f(X)), nrow = x_res )      # f(x)

# Abbildungsparameter
par(
    family    = "sans",
    mfcol     = c(1,1),
    pty       = "s",
    bty       = "l",
    lwd       = 1,
    las       = 1,
    xaxs      = "i",
    yaxs      = "i",
    font.main = 1,
    cex       = 1.1,
    cex.main  = 1.3)

# Funktion
contour(x_1, x_2, fX,
    xlim      = c(x_min,x_max),
    ylim      = c(x_min,x_max),
    xlab      = TeX("$x_1$"),
    ylab      = TeX("$x_2$"),
    levels    = c(0.1, 0.5, 1:8),
    main      = "")

# Gradientenbeispiele
x12   = matrix(c(0.7, -0.5, -0.3,  0.1,
                 0.7, -0.4,  0.1, -1),
               nrow  = 2,
               byrow = T)
cols  = c("black", "red", "blue", "lightgreen")
```

```r
# Punkte und Pfeile
for(i in 1:ncol(x12)){
    points(x12[1,i], x12[2,i],
        col     = cols[i])
    arrows(
        x0      = x12[1,i],
        y0      = x12[2,i],
        x1      = nabla_f(x12[,i])[1],
        y1      = nabla_f(x12[,i])[2],
        col     = cols[i],
        angle   = 30,
        length  = 0.1,
        lwd     = 2)
}

# Speichern als PDF
dev.copy2pdf(
    file        = "./Abbildungen/gradientbeispiel.pdf",
    width       = 5,
    height      = 5)
dev.off()
```

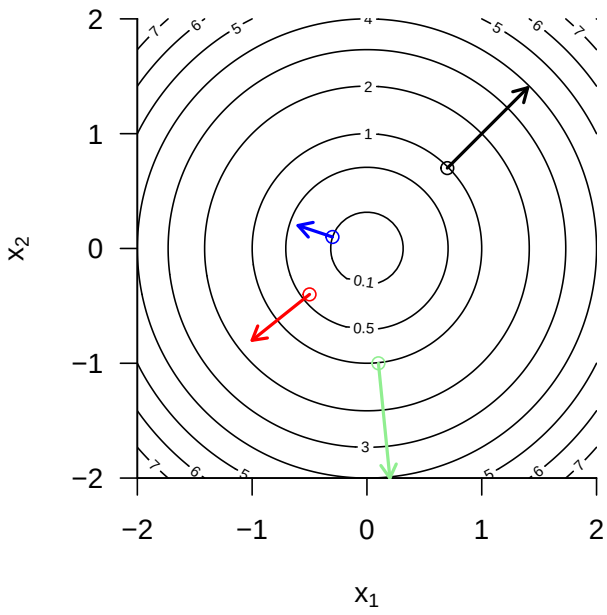

Abbildung 1. Beispielhafte Gradienten von $f(x) = x_1^2 + x_2^2$.

Gradientenverfahren

Um das Minimum von $f(x)$ zu finden, d.h. den Wert $x \in \mathbb{R}^2$, für den die Funktion f ihren kleinsten Wert annimmt, bedienen wir uns eines Gradientenverfahrens, das in folgendem **R**-Code implementiert wird.

```
# Parameter
n      = 2                                  # Dimension
alpha  = 1e-1                               # Lernrate
delta  = 1e-2                               # Konvergenzkriterium

# Initialisierung
x_k    = matrix(c(0.61, 0.85), nrow = 2)    # zufälliger Startpunkt in [0,1]^2
x      = x_k                                # Initialisierung Iteranden
fx     = f(x_k)                             # Initialisierung Funktionswerte
crt    = norm(nabla_f(x_k))                 # Initialisierung Kriterium

# Iterationen
while(norm(nabla_f(x_k)) > delta){

    # Argumentupdate
    x_k  = x_k - alpha*nabla_f(x_k)         # Funktionsargument aktualisieren

    # Dokumentation
    x    = cbind(x,x_k)                     # Funktionsargument abspeichern
    fx   = c(fx,f(x_k))                     # Funktionswert abspeichern
    crt  = c(crt, norm(nabla_f(x_k)))       # Kriterium abspeichern

}
```

Schließlich visualisieren wir das Ergebnis und den Verlauf der Minimierung von $f(x)$ in Abbildung 2, die durch folgenden **R**-Code erzeugt wird.

```
# R-Pakete
library(latex2exp)

# Abbildungsparameter
par(
    family      = "sans",
    mfcol       = c(1,3),
    pty         = "s",
    bty         = "l",
    lwd         = 1,
    las         = 1,
    xaxs        = "i",
    yaxs        = "i",
    font.main   = 1,
    cex         = 1.1,
```

```
    cex.main    = 1.3)

# Definitionsmengenraum
x_min  = -1                                          # x_i Minimum
x_max  = 1                                           # x_i Maximum
x_res  = 1e2                                         # x_i Auflösung
x_1    = seq(x_min, x_max, length.out = x_res)       # x_1 Raum
x_2    = seq(x_min, x_max, length.out = x_res)       # x_2 Raum
X      = expand.grid(x_1, x_2)                       # X = (x_1,x_2)^T Raum
fX     = matrix(as.matrix(f(X)), nrow = x_res )      # f(x)
cols   = hcl.colors(10, "YlOrRd")

# Funktion
contour(x_1, x_2, fX,
    col     = cols,
    xlim    = c(x_min,x_max),
    ylim    = c(x_min,x_max),
    xlab    = TeX("$x_1$"),
    ylab    = TeX("$x_2$"),
    nlevels = 10,
    main    = "f(x)")

# Argumente
points(x[1,], x[2,],
    type = "o")

# Funktionswerte
plot(1:length(fx), fx,
    type = "l",
    lwd  = 1,
    xlab = "k",
    ylab = "",
    main = TeX("$f(x^k)$"),
    ylim = c(0,1))

# Kriterien
plot(1:length(crt), crt,
    type = "l",
    lwd  = 1,
    xlab = "k",
    ylab = "",
    main = TeX("$|| \\nabla f(x^k) ||$"),
    ylim = c(0,2))

# Speichern als PDF
dev.copy2pdf(
    file        = "./Abbildungen/gradientenverfahren.pdf",
    width       = 12,
    height      = 4)
dev.off()
```

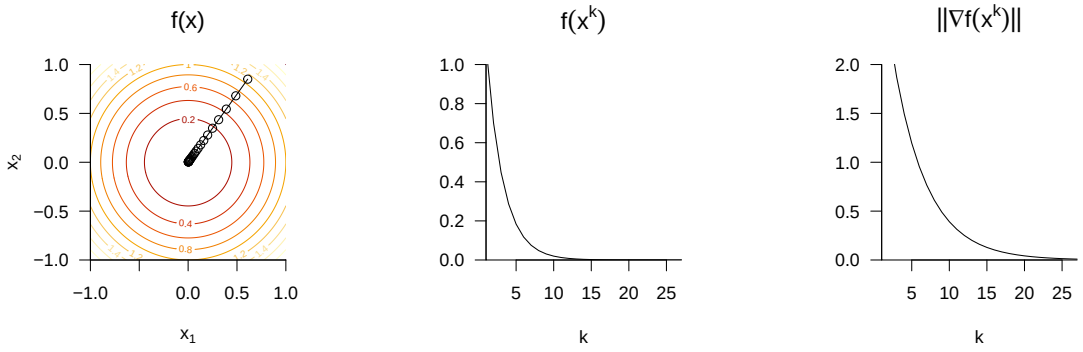

Abbildung 2. Gradientenverfahren für die Funktion $f(x) = x_1^2 + x_2^2$.