Übungsblatt 2

Aufgabe 2.1

Finde für die gegebene Folge a_n zwei einfachere Folgen b_n und c_n, sodass $a_n = O(b_n)$ und $a_n = o(c_n)$.

(a) $a_n = 100n^{-1} + 2n^{0.5} + 50n^{1.5}$,
(b) $a_n = 2\log(n)/n + \log(2)/n^2 + \log(1 + 1/n)$.

Aufgabe 2.2

Sei ϵ_n eine Folge und $a_n = O(\epsilon_n + \epsilon_n^2)$. Zeige, dass $a_n = O(\epsilon_n^2)$, falls $\epsilon_n \to \infty$, aber $a_n = O(\epsilon_n)$, falls $\epsilon_n \to 0$.

Aufgabe 2.3

Seien $\boldsymbol{X}, \boldsymbol{X}_1, \dots, \boldsymbol{X}_n \in \mathbb{R}^d$ unabhängig und identisch verteilte Zufallsvektoren mit $\mathbb{V}\mathrm{ar}(\boldsymbol{X}) = \Sigma$. Für jeden Parameter $\boldsymbol{\beta}$ gilt

$$\sigma^2(\boldsymbol{\beta}) = \mathbb{V}\mathrm{ar}(\boldsymbol{\beta}^\top \boldsymbol{X}) = \boldsymbol{\beta}^\top \Sigma \boldsymbol{\beta}.$$

Die Kovarianzmatrix Σ lässt sich wie folgt schätzen:

$$\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^{1} (\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})(\boldsymbol{X}_i - \widehat{\boldsymbol{\mu}})^\top, \qquad \widehat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}_i.$$

Entsprechend definieren wir $\widehat{\sigma}^2(\boldsymbol{\beta}) = \boldsymbol{\beta}^\top \widehat{\Sigma} \boldsymbol{\beta}$. Aus diesen Formeln können wir den folgenden Algorithmus zur Berechnung von $\widehat{\sigma}^2(\boldsymbol{\beta})$ ableiten:

```r
#' @param X Matrix mit Zeilen X_1, ..., X_n.
#' @param beta Parametervektor
sigma_2 <- function(X, beta) {
  n <- nrow(X)
  d <- ncol(X)
  Sig <- matrix(0, d, d)
  for (i in 1:n) {
    Sig <- Sig + (X[i, ] - colMeans(X)) %*% t(X[i, ] - colMeans(X))
  }
  as.numeric(t(beta) %*% Sig %*% beta / n)
}
```

(a) Gebe die Laufzeit- und Speicher-Komplexität des Algorithmus in Landau-Notation an.

(b) Überprüfe die Laufzeiteigenschaften numerisch. Erzeuge dazu mehrere Datensätze mit $n \in [10, 1000]$ und $d \in [10, 1000]$ und benchmarke den Algorithmus (z.B. mit `microbenchmark::microbenchmark()`).

(c) Finde und benchmarke einen alternativen Algorithmus mit Laufzeit- und Speicherkomplexität $O(nd)$.