Computational Statistics

Prof. Dr. Dominik Liebl

Inhaltsverzeichnis

Informationen

Dies ist das Skript zur Vorlesung *Computational Statistik* (B.Sc. Informatik & Data Science)

Vorlesungszeiten

Wochentag	Uhrzeit	Hörsaal
Dienstag	9:15-10:45	Online-Vorlesung
Freitag	8:30-10:00	Online-Vorlesung

RCodes

Die RCodes zu den einzelnen Kapiteln können hier heruntergeladen werden: RCodes

Leseecke

Folgende *frei zugängliche* Lehrbücher enthalten Teile dieses Kurses. In den jeweiligen Kapiteln, werde ich auf die einzelnen Bücher verweisen.

- Pattern Recognition and Machine Learning (by Christopher Bishop)
- An Introduction to Statistical Learning, with Applications in R (by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani).
- Statistical Learning with Sparsity: the Lasso and Generalizations (by Trevor Hastie, Robert Tibshirani and Martin Wainwright).
- Elements of Statistical Learning: Data mining, Inference and Prediction (by Trevor Hastie, Robert Tibshirani and Jerome Friedman).
- Computer Age Statistical Inference: Algorithms, Evidence and Data Science (by Bradley Efron and Trevor Hastie)

Florence Nightingale

Das Logo zu diesem Skript stammt von einer Briefmarke zur Erinnerung an die Krankenschwester und inspirierende Statistikerin, Florence Nightingale. Nightingale war die Begründerin der modernen westlichen Krankenpflege und Pionierin der visuellen Veranschaulichung von Zusammenhängen in der Statistik. Sie ist die erste Frau, die in die britische Royal Statistical Society aufgenommen wurde; später erhielt sie auch die Ehrenmitgliedschaft der American Statistical Association.

Kapitel 1

Der Expectation Maximization (EM) Algorithmus

Der EM Algorithmus wird häufig verwendet, um komplizierte Maximum Likelihood Schätzprobleme zu vereinfachen bzw. überhaupt erst möglich zu machen. In diesem Kapitel stellen wir den EM Algorithmus zur Schätzung von Gaußschen Mischverteilungen vor, da der EM Algorithmus hier wohl seine bekannteste Anwendung hat. Bereits die originale Arbeit zum EM Algorithmus (Dempster et al., 1977) beschäftigt sich mit der Schätzung von Gaußschen Mischverteilungen.

Mögliche Anwendungen von Gaußschen Mischverteilungen:

- Automatisierte Videobearbeitungen: Z.B. Bildeinteilungen in Vorder- und Hintergrund.
- Automatisierte Erkennung von Laufstilen
- Generell: Auffinden von Gruppierungen (zwei oder mehr) in den Daten (**Clusteranalyse**).

Lernziele für dieses Kapitel

Sie können …

- ein **Anwendungsfeld** des EM Algorithmuses **benennen**.
- die **Probleme** der klassischen Maximum Likelihood Methode zur Schätzung von Gaußschen Mischverteilungen **benennen und erkläutern**.
- die **Grundidee** des EM Algorithmuses **erläutern**.
- den **EM Algorithmus** zur Schätzung von Gaußschen Mischverteilungen **anwenden**.

- das **Grundidee** der Vervollständigung der Daten durch latente Variablen **erläutern**.

Begleitlektüre(n)

Zur Vorbereitung der Klausur ist es grundsätzlich aussreichend das Kursskript durchzuarbeiten - aber Lesen hat ja noch nie geschadet. Dieses Kapitel basiert hauptsächlich auf:

- Kapitel 9 in **Pattern Recognition and Machine Learning** (Bishop, 2006). Die pdf-Version des Buches ist frei erhältlichen: **pdf-Version**

Weiteren guten Lesestoff zum EM Algoithmus gibt es z.B. hier:

- Kapitel 8.5 in **Elements of Statistical Learning: Data Mining, Inference and Prediction** (Hastie et al., 2009). Die pdf-Version des Buches ist frei erhältlichen: **pdf-Version**

R-Pakete für diese Kapitel

Folgende R-Pakete werden für dieses Kapitel benötigt:

```
pkgs <- c("tidyverse",       # Die tidyverse-Pakete
          "palmerpenguins", # Pinguin-Daten
          "scales",          # Transparente Farben: alpha()
          "RColorBrewer",    # Hübsche Farben
          "mclust",          # Schätzung/Verwendung
                             # Gaußschen Mischverteilungen
          "MASS")            # Erzeugung von Zufallszahlen aus
                             # einer multiv. Normalverteilung
install.packages(pkgs)
```

1.1 Motivation: Clusteranalyse mit Hilfe Gaußscher Mischverteilungen

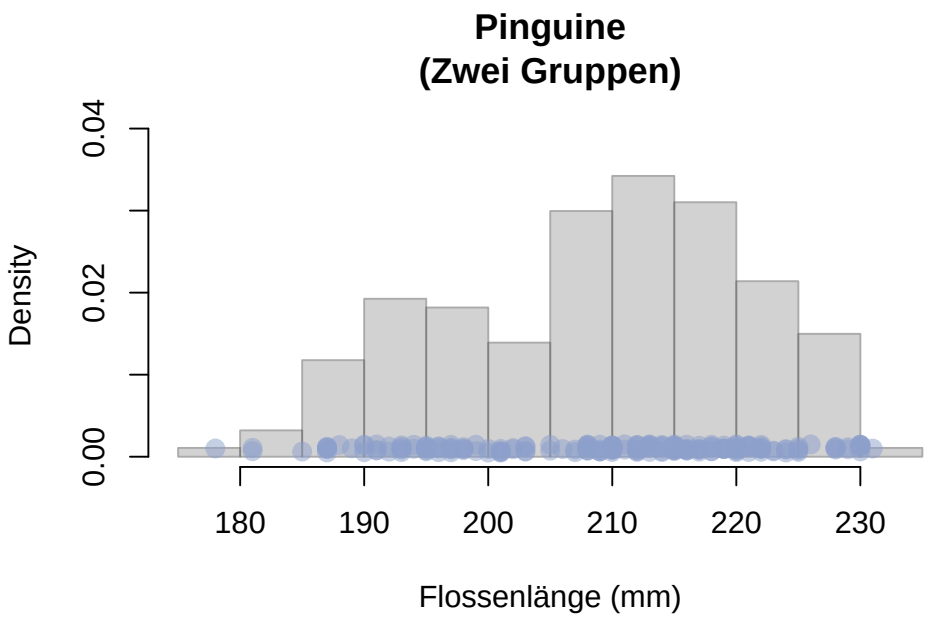
Als Datenbeispiel verwendent wir die `palmerpenguins` Daten (Horst et al., 2020).

Diese Daten stammen aus Vermessungen von Pinguinpopulationen auf dem Palmer-Archipel (Antarktische Halbinsel). Pinguine sind oft schwer von einander zu unterscheiden. Wir werden versuchen, mit Hilfe einer Gaußschen Mischverteilung Gruppierungen in den Pinguindaten (Flossenlänge) zu finden. Um solche Mischverteilungen schätzen zu können, führen wir den EM Algorithmus ein.

Der folgende Code-Chunck bereitet die Daten auf.

Achtung: Wir haben zwar die Information zu den verschiedenen Pinguin-Arten (`Penguine_Art`) tun aber im Folgenden so, also ob wir diese Information nicht kennen würden. Wir wollen alleine auf Basis der Flossenlängen (`Penguine_Flosse`) die Gruppenzugehörigkeiten per Clusteranalyse bestimmen. (Im Nachhinein können wir dann mit Hilfe der Daten in `Penguine_Art` prüfen, wie gut unsere Clusteranalyse ist.)

```r
library("palmerpenguins") # Pinguin-Daten
library("RColorBrewer")   # Hübsche Farben
library("scales")         # Für transparente Farben: alpha()

col_v <- RColorBrewer::brewer.pal(n = 3, name = "Set2")

## Vorbereitung der Daten:
Pinguine <- palmerpenguins::penguins %>% # Pinguin-Daten
  tidyr::as_tibble() %>%                 # Datenformat: 'tibble'-dataframe
  dplyr::filter(species!="Adelie") %>%   # Pinguin-Art 'Adelie' löschen
  droplevels() %>%                       # Lösche das nicht mehr benötigte Adelie-Level
  tidyr::drop_na() %>%                   # NAs löschen
  dplyr::mutate(Art    = species,        # Variablen umbenennen
                Flosse = flipper_length_mm) %>%
  dplyr::select(Art, Flosse)             # Variablen auswählen

##
n      <- nrow(Pinguine)                 # Stichprobenumfang

## Variable 'Penguine_Art' aus Pinguine-Daten "herausziehen"
Penguine_Art    <- dplyr::pull(Pinguine, Art)

## Variable 'Penguine_Flosse' aus Pinguine-Daten "herausziehen"
Penguine_Flosse <- dplyr::pull(Pinguine, Flosse)

## Plot
## Histogramm:
hist(x = Penguine_Flosse, freq = FALSE,
     xlab="Flossenlänge (mm)", main="Pinguine\n(Zwei Gruppen)",
     col=gray(.65,.5), border=gray(.35,.5), ylim=c(0.0003, 0.039))
## Stipchart hinzufügen:
stripchart(x = Penguine_Flosse, method = "jitter",
           jitter = .0005, at = .001,
           pch = 21, col=alpha(col_v[3],.5),
           bg=alpha(col_v[3],.5), cex=1.3, add = TRUE)
```



Das Clusterverfahren basierend auf Gaußschen Mischverteilungen:

1. Gaußsche Mischverteilung (**per EM Algorithmus**) schätzen
2. Die Datenpunkte wie bei der Linearen (oder Quadratischen) Diskriminanz-Analyse den Gruppen zuordnen (siehe Abbildung 1.1)

Abbildung 1.1 zeigt das Resultat einer Clusteranalyse basierend auf einer Mischverteilung zweier gewichteter Normalverteilungen. Cluster-Ergebnis: 95% der Pinguine konnten richtig zugeordnet werden - lediglich auf Basis ihrer Flossenlängen.

Mit Hilfe der folgenden R-Codes kann die obige Clusteranalyse und die Ergebnisgrafik repliziert werden:

```
## mclust R-Paket:
## Clusteranalyse mit Hilfe von Gaußschen Mischmodellen
suppressMessages(library("mclust"))

## Anzahl der Gruppen
G <- 2

## Schätzung des Gaußschen Mischmodells (per EM Algorithmus)
## und Clusteranalyse
mclust_obj <- mclust::Mclust(data = Penguine_Flosse, G=G,
                             modelNames = "V",
                             verbose = FALSE)

# summary(mclust_obj)
```

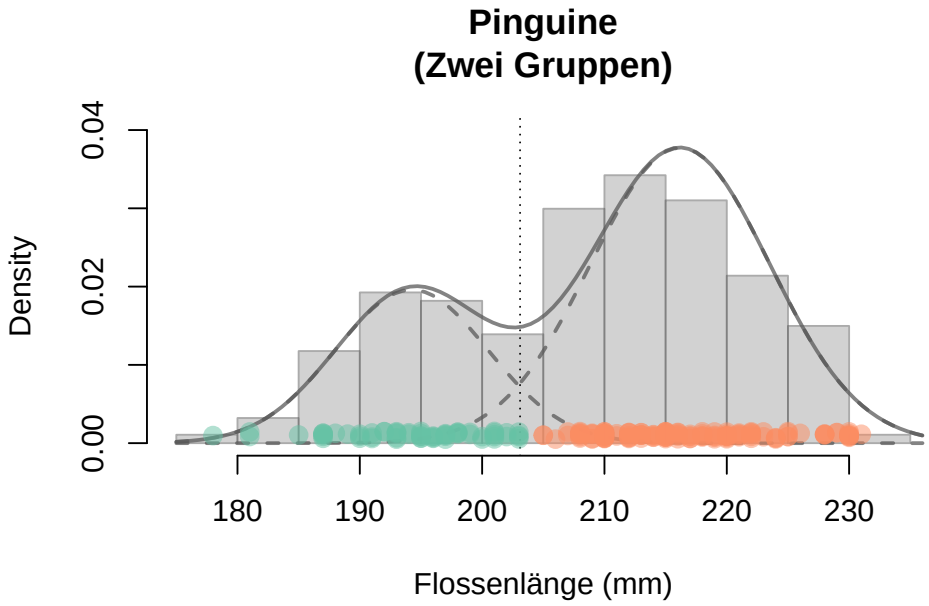


Abbildung 1.1: Clusteranalyse basierend auf einer Mischverteilung mit zwei gewichteten Normalverteilungen.

```
# str(mclust_obj)

## Geschätzte Gruppen-Zuordnungen (Cluster-Resultat)
class <- mclust_obj$classification

## Anteil der korrekten Zuordnungen:
# cbind(class, Penguine_Art)
round(sum(class == as.numeric(Penguine_Art))/n, 2)

## Geschätzte Mittelwerte
mean_m <- t(mclust_obj$parameters$mean)

## Geschätzte Varianzen (und evtl. Kovarianzen)
cov_l  <- list("Cov1" = mclust_obj$parameters$variance$sigmasq[1],
               "Cov2" = mclust_obj$parameters$variance$sigmasq[2])

## Geschätzte Gewichte (a-priori-Wahrscheinlichkeiten)
prop_v <- mclust_obj$parameters$pro

## Auswerten der Gaußsche Mischung-Dichtefunktion
np      <- 100 # Anzahl der Auswertungspunkte
xxd     <- seq(min(Penguine_Flosse)-3, max(Penguine_Flosse)+5, length.out = np)
```

```
## Mischungs-Dichte
yyd     <- dnorm(xxd, mean_m[1], sqrt(cov_l[[1]]))*prop_v[1] +
           dnorm(xxd, mean_m[2], sqrt(cov_l[[2]]))*prop_v[2]
## Einzel-Dichten
yyd1    <- dnorm(xxd, mean_m[1], sqrt(cov_l[[1]]))*prop_v[1]
yyd2    <- dnorm(xxd, mean_m[2], sqrt(cov_l[[2]]))*prop_v[2]

## Plot
hist(x = Penguine_Flosse, xlab="Flossenlänge (mm)", main="Pinguine\n(Zwei Gruppen)",
     col=gray(.65,.5), border=gray(.35,.5), freq = FALSE, ylim=c(0, 0.04))
lines(x = xxd, y=yyd, lwd=2, col=gray(.35,.75))
lines(x = xxd, y=yyd1, lwd=2, col=gray(.35,.75), lty=2)
lines(x = xxd, y=yyd2, lwd=2, col=gray(.35,.75), lty=2)
stripchart(Penguine_Flosse[class==1], method = "jitter", jitter = .0005, at = .001,
           pch = 21, col=alpha(col_v[1],.5), bg=alpha(col_v[1],.5), cex=1.3, add = TRU
stripchart(Penguine_Flosse[class==2], method = "jitter", jitter = .0005, at = .001,
           pch = 21, col=alpha(col_v[2],.5), bg=alpha(col_v[2],.5), cex=1.3, add = TRU
```

1.2 Der EM Algorithmus zur ML-Schätzung Gaußscher Mischverteilungen

1.2.1 Gaußsche Mischmodelle (GMM)

Eine Zufallsvariable X, die einer Gauschen Mischverteilung folgt, bezeichnen wir als

$$X \sim \mathcal{N}_{mix}(G, \pi, \mu, \sigma)$$

Die dazugehörige Dichtefunktion einer Gaußschen Mischverteilung ist folgendermaßen definiert:

$$f_G(x|\pi, \mu, \sigma) = \sum_{g=1}^{G} \pi_g f(x|\mu_g \sigma_g) \tag{1.1}$$

- **Gewichte:** $\pi = (\pi_1, \dots, \pi_G)$ mit $\pi_g > 0$ und $\sum_{g=1}^{G} \pi_g = 1$
- **Mittelwerte:** $\mu = (\mu_1, \dots, \mu_G)$ mit $\mu_g \in \mathbb{R}$
- **Standardabweichungen:** $\sigma = (\sigma_1, \dots, \sigma_G)$ mit $\sigma_g > 0$
- **Normalverteilung der Gruppe** $g = 1, \dots, G$**:**

$$f(x|\mu_g \sigma_g) = \frac{1}{\sqrt{2\pi}\sigma_g} \exp\left(-\frac{1}{2}\left(\frac{x - \mu_g}{\sigma_g}\right)^2\right)$$

- **Unbekannte Parameter:** π, μ und σ

1.2.2 Maximum Likelihood (ML) Schätzung

Man kann versuchen die unbekannten Parameter $\pi = (\pi_1, \dots, \pi_G)$, $\mu = (\mu_1, \dots, \mu_G)$ und $\sigma = (\sigma_1, \dots, \sigma_G)$ eines Gaußschen Mischmodells klassisch mit Hilfe der Maximum Likelihood Methode zu schätzen.

Ich sag's gleich: Der Versuch wird scheitern.

Wiederholung der Grundidee der ML-Schätzung:

- **Annahme:** Die Daten $\mathbf{x} = (x_1, \dots, x_n)$ sind eine Realisation einer einfachen (also i.i.d.) Zufallsstichprobe $(X_1, \dots, X_n)$ mit

  $$X_i \sim \mathcal{N}_{mix}(G, \pi, \mu, \sigma)$$

  für alle $i = 1, \dots, n$.

  Die Daten $\mathbf{x} = (x_1, \dots, x_n)$ „kennen" also die unbekannten Parameter π, μ und σ und wir müssen ihnen diese Informationen „nur noch" entlocken.

- **Schätz-Idee:** Wähle π, μ und σ so, dass $f_G(\cdot|\pi, \mu, \sigma)$ **„optimal"** zu den beobachteten Daten $\mathbf{x}$ passt.

- **Umsetzung der Schätz-Idee:** Maximiere (bzgl. π, μ und σ) die Likelihood Funktion

  $$\mathcal{L}(\pi, \mu, \sigma|\mathbf{x}) = \prod_{i=1}^{n} f_G(x_i|\pi, \mu, \sigma)$$

  Bzw. maximiere die Log-Likelihood Funktion (einfachere Maximierung)

  $$\begin{aligned}\ln\left(\mathcal{L}(\pi, \mu, \sigma|\mathbf{x})\right) = \ell(\pi, \mu, \sigma|\mathbf{x}) &= \sum_{i=1}^{n} \ln\left(f_G(x_i|\pi, \mu, \sigma)\right) \\ &= \sum_{i=1}^{n} \ln\left(\sum_{g=1}^{G} \pi_g f(x_i|\mu_g, \sigma_g)\right)\end{aligned}$$

  Beachte: Die Maximierung muss die Parameterrestriktionen in (1.1) berücksichtigen ($\sigma_g > 0$ und $\pi_g > 0$ für alle $g = 1, \dots, G$ und $\sum_{g=1}^{G} \pi_g = 1$).

- Die maximierenden Parameterwerte $\hat{\pi}$, $\hat{\mu}$ und $\hat{\sigma}$ sind die **ML-Schätzer**. Das kann man so ausdrücken:

  $$(\hat{\pi}, \hat{\mu}, \hat{\sigma}) = \arg\max_{\pi,\mu,\sigma} \ell(\pi, \mu, \sigma|\mathbf{x})$$

Probleme mit Singularitäten bei Numerische Lösungen: Versucht man obiges Maximierungsproblem numerisch mit Hilfe des Computers zu lösen, wird man schnell merken, dass die Ergebnisse höchst instabil, unplausibel und wenig vertrauenswürdig sind. Der Grund für diese instabilen Schätzungen sind Probleme mit Singularitäten: Für echte GMMs ($G > 1$) treten während einer

numerischen Maximierung sehr leicht Probleme mit Singularitäten auf. Dies geschieht immer dann, wenn eine der Normalverteilungskomponenten versucht den ganzen Datensatz $\mathbf{x}$ zu beschreiben und die andere(n) Normalverteilungskomponente(n) versuchen lediglich einzelne Datenpunkte zu beschreiben. Eine Gaußsche Dichtefunktion, die sich um einen einzigen Datenpunkt x_i konzentriert (d.h. $\mu_g = x_i$ und $\sigma_g \to 0$) wird dabei sehr große Werte annehmen (d.h. $f(x_i|\mu_g = x_i, \sigma_g) \to \infty$ für $\sigma_g \to 0$) und so die Log-Likelihood auf unerwünschte Weise maximieren (siehe Abbildung 1.2). Solch **unerwünschte, triviale Maximierungslösungen** führen i.d.R. zu unplausiblen Schätzergebnissen.

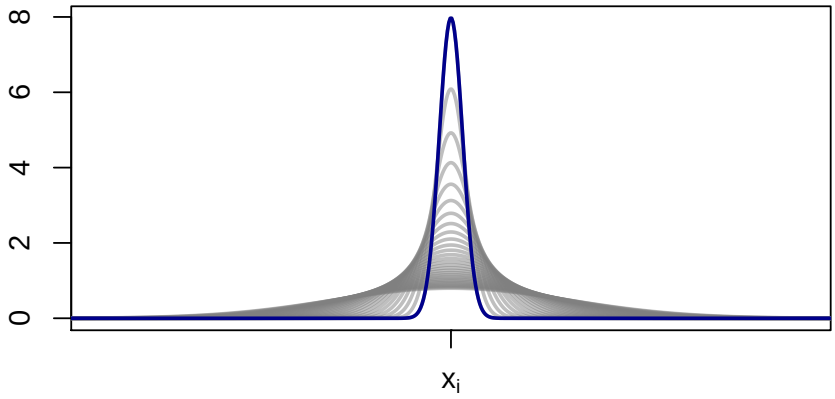


Abbildung 1.2: Normalverteilung mit $\mu_g = x_i$ für $\sigma_g \to 0$.

Analytische Lösung: Es ist zwar etwas mühsam, aber man kann versuchen die Log-Likelihood analytisch zu maximieren. Tut man dies sich das an, kommt man zu folgenden Ausdrücken:

$$\hat{\pi}_g = \frac{1}{n}\sum_{i=1}^{n} p_{ig}$$

$$\hat{\mu}_g = \sum_{i=1}^{n} \frac{p_{ig}}{\left(\sum_{j=1}^{n} p_{jg}\right)} x_i$$

$$\hat{\sigma}_g = \sqrt{\sum_{i=1}^{n} \frac{p_{ig}}{\left(\sum_{j=1}^{n} p_{jg}\right)} \left(x_i - \hat{\mu}_g\right)^2}$$

für $g = 1, \ldots, G$.

> Die Herleitung der Ausdrücke für μ_g, σ_g und π_g, $g = 1, \ldots, G$, ist wirklich etwas lästig (mehrfache Anwendungen der Kettenregel, Produktregel, etc., sowie Anwendung des Lagrange-Multiplikator Verfahrens zur Optimierung unter Nebenbedingungen) aber machbar. In einer der Übungsaufgaben dürfen Sie den Ausdruck für $\hat{\mu}_g$ herleiten.

Aber: Diese Ausdrücke für $\hat{\pi}_g$, $\hat{\mu}_g$ und $\hat{\sigma}_g$ hängen von den **unbekannten**

Parametern $\pi = (\pi_1, \dots, \pi_G)$, $\mu = (\mu_1, \dots, \mu_G)$ und $\sigma = (\sigma_1, \dots, \sigma_G)$, denn:

$$p_{ig} = \frac{\pi_g f(x_i|\mu_g, \sigma_g)}{f_G(x_i|\pi, \mu, \sigma)}$$

für $i = 1, \dots, n$ und $g = 1, \dots, G$. Die Ausdrücke für $\hat{\pi}_g$, $\hat{\mu}_g$, und $\hat{\sigma}_g$ erlauben also keine keine direkte Schätzung der unbekannten Parameter π_g, μ_g und σ_g.

Lösung: Der EM Algorithmus

1.2.3 Der EM Algorithmus für GMMs

Die Ausdrücke für $\hat{\pi}_g$, $\hat{\mu}_g$ und $\hat{\sigma}_g$ legen jedoch ein einfaches, iteratives ML-Schätzverfahren nahe: Nämlich einer alternierenden Schätzung von p_{ig} und $(\hat{\pi}_g, \hat{\mu}_g, \hat{\sigma}_g)$.

Der Der EM Algorithmus:

1. Setze Startwerte $\pi^{(0)}$, $\mu^{(0)}$ und $\sigma^{(0)}$
2. Für $r = 1, 2, \dots$
   - **(Expectation)** Berechne:

$$p_{ig}^{(r)} = \frac{\pi_g^{(r-1)} f(x_i|\mu_g^{(r-1)}, \sigma_g^{(r-1)})}{f_G(x_i|\pi^{(r-1)}, \mu^{(r-1)}, \sigma^{(r-1)})}$$

   - **(Maximization)** Berechne:

$$\hat{\pi}_g^{(r)} = \frac{1}{n}\sum_{i=1}^n p_{ig}^{(r)}, \qquad \hat{\mu}_g^{(r)} = \sum_{i=1}^n \frac{p_{ig}^{(r)}}{\left(\sum_{j=1}^n p_{jg}^{(r)}\right)} x_i$$

$$\hat{\sigma}_g^{(r)} = \sqrt{\sum_{i=1}^n \frac{p_{ig}^{(r)}}{\left(\sum_{j=1}^n p_{jg}^{(r)}\right)} \left(x_i - \hat{\mu}_g^{(r)}\right)^2}$$

3. Prüfe Konvergenz:
   - Stoppe, falls sich der Wert der maximierten Log-Likelihood Funktion, $\ell(\pi^{(r)}, \mu^{(r)}, \sigma^{(r)}|\mathbf{z})$, nicht mehr ändert.

Der obige pseudo-Code wird im Folgenden Code-Chunck umgesetzt:

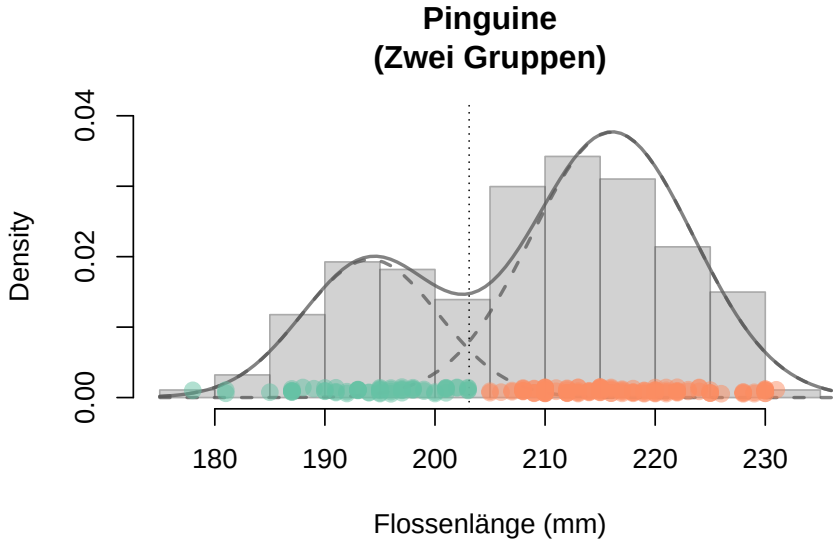
```
library("MASS")
library("mclust")

## Daten:
x <- cbind(Penguine_Flosse) # Daten [n x d]-Dimensional.
d <- ncol(x)                # Dimension (d=1: univariat)
n <- nrow(x)                # Stichprobenumfang
G <- 2                      # Anzahl Gruppen
```

```
## Weitere Deklarationen:
llk       <- matrix(NA, n, G)
p         <- matrix(NA, n, G)
loglikOld <- 1e07
tol       <- 1e-05
it        <- 0
check     <- TRUE


## EM Algorithmus

## 1. Startwerte für pi, mu und sigma:
pi    <- rep(1/G, G)               # Naive pi
sigma <- array(diag(d), c(d,d,G)) # Varianz = 1
mu    <- t(MASS::mvrnorm(G, colMeans(x), sigma[,,1]*4) )

while(check){

  ## 2.a Expectation-Schritt
  for(g in 1:G){
    p[,g] <- pi[g] * mclust:::dmvnorm(x, mu[,g], sigma[,,g])
  }
  p <- sweep(p, 1, STATS = rowSums(p), FUN = "/")

  ## 2.b Maximization-Schritt
  par   <- mclust::covw(x, p, normalize = FALSE)
  mu    <- par$mean
  sigma <- par$S
  pi    <- colMeans(p)

  ## 3. Prüfung der Konvergenz
  for(g in 1:G) {
    llk[,g] <- pi[g] * mclust:::dmvnorm(x, mu[,g], sigma[,,g])
  }
  loglik <- sum(log(rowSums(llk))) # aktueller max. Log-Likelihood Wert
  ##
  diff      <- abs(loglik - loglikOld)/abs(loglik) # Änderungsrate
  loglikOld <- loglik
  it        <- it + 1
  ## Änderungsrate noch groß genug (> tol)?
  check     <- diff > tol
}

## Schätz-Resultate:
results <- matrix(c(pi, mu, sqrt(sigma)),
```

```
                  nrow = 3, ncol = 2, byrow = TRUE,
                  dimnames = list(
            c("Gewichte", "Mittelwerte", "Standardabweichungen"),
            c("Gruppe 1", "Gruppe 2")))
##
results %>% round(., 2)
#>                      Gruppe 1 Gruppe 2
#> Gewichte                 0.31     0.69
#> Mittelwerte            194.27   216.20
#> Standardabweichungen     6.27     7.32
```

Das finale Schätzergebnis erlaubt es uns, Abbildung 1.1 zu replizieren.

```
## Auswerten der Gaußsche Mischungs-Dichtefunktion
np      <- 100 # Anzahl der Auswertungspunkte
xxd     <- seq(min(Penguine_Flosse)-3, max(Penguine_Flosse)+5, length.out = np)
## Mischungs-Dichte
yyd     <- dnorm(xxd, mu[1,1], sqrt(sigma)[,,1])*pi[1] +
           dnorm(xxd, mu[1,2], sqrt(sigma)[,,2])*pi[2]
## Einzel-Dichten
yyd1    <- dnorm(xxd, mu[1,1], sqrt(sigma)[,,1])*pi[1]
yyd2    <- dnorm(xxd, mu[1,2], sqrt(sigma)[,,2])*pi[2]

## Plot
hist(x = Penguine_Flosse, xlab="Flossenlänge (mm)", main="Pinguine\n(Zwei Gruppen)",
     col=gray(.65,.5), border=gray(.35,.5), freq = FALSE, ylim=c(0, 0.04))
lines(x = xxd, y=yyd, lwd=2, col=gray(.35,.75))
lines(x = xxd, y=yyd1, lwd=2, col=gray(.35,.75), lty=2)
lines(x = xxd, y=yyd2, lwd=2, col=gray(.35,.75), lty=2)
abline(v=203.1, lty=3)
stripchart(Penguine_Flosse[class==1], method = "jitter", jitter = .0005, at = .001,
           pch = 21, col=alpha(col_v[1],.5), bg=alpha(col_v[1],.5), cex=1.3, add = TRUE)
stripchart(Penguine_Flosse[class==2], method = "jitter", jitter = .0005, at = .001,
           pch = 21, col=alpha(col_v[2],.5), bg=alpha(col_v[2],.5), cex=1.3, add = TRUE)
```



1.3 Der alternative (wahre) Blick auf den EM Algorithmus

Der EM Algorithmus ermöglicht es Maximum Likelihood Probleme zu vereinfachen, indem man die Daten durch nicht beobachtete („latente“) Variablen vervollständigt. Dieses Prinzip ist der eigentliche wahre Beitrag des EM Algorithmus. Es ermöglicht die Lösung verschiedener Maximum Likelihood Probleme - wie bleiben aber hier bei der Schätzung von GMMs.

> **Zur Erinnerung:** Wir haben es ja nicht geschafft, die Log-Likelihood Funktion
>
> $$\ell(\pi, \mu, \sigma|\mathbf{x}) = \sum_{i=1}^{n} \ln \left(\sum_{g=1}^{G} \pi_g f(x_i|\mu_g, \sigma_g) \right)$$
>
> direkt zu maximieren. Die $\ln(\sum_{g=1}^{G}[...])$-Konstruktion macht einem hier das Leben schwer.

1.3.1 Vervollständigung der Daten

In unseren Pinguin-Daten gibt zwei Gruppen ($g \in \{1, 2\}$). Es gäbe also im Prinzip $G = 2$-dimensionale Zuordnungsvektoren (z_{i1}, z_{i2}) mit

$$(z_{i1}, z_{i2}) = \begin{cases} (1,0) & \text{falls Pinguin } i \text{ zu Gruppe } g = 1 \text{ gehört.} \\ (0,1) & \text{falls Pinguin } i \text{ zu Gruppe } g = 2 \text{ gehört.} \end{cases}$$

Im Fall von $G > 2$ Gruppen:

$$(z_{i1}, \dots, z_{ig}, \dots, z_{iG}) = \begin{cases} (1, 0, \dots, 0) & \text{falls Datenpunkt } i \text{ zu Gruppe } g = 1 \text{ gehört.} \\ (0, 1, \dots, 0) & \text{falls Datenpunkt } i \text{ zu Gruppe } g = 2 \text{ gehört.} \\ \quad \vdots & \\ (0, 0, \dots, 1) & \text{falls Datenpunkt } i \text{ zu Gruppe } g = G \text{ gehört.} \end{cases}$$

Die Zuordnungen z_{ig} können also die Werte $z_{ig} \in \{0, 1\}$ annehmen, wobei aber gelten muss, dass $\sum_{g=1}^{G} z_{ig} = 1$.

> **Beachte:** Für jeden Datenpunkt i (jeder Pinguin i) gibt es nur **eine** Gruppe (daher $\sum_{g=1}^{G} z_{ig} = 1$). Dies ist eine wichtige Restriktion von GMMs, welche bei den Pinguindaten unproblematisch ist. Bei Anwendungen mit hirarchischen Gruppierungen ist dies aber evtl. problematisch.

Die Zuordnungen z_{ig} sind leider unbekannt (latent). Wir wissen aber trotzdem etwas über diese Zuordnungen. Die Gewichte $\pi_1, \dots, \pi_G$ der Gaußschen Mischverteilung

$$f_G(x|\pi, \mu, \sigma) = \sum_{g=1}^{G} \pi_g f(x|\mu_g \sigma_g),$$

geben die Anteile der Einzelverteilungen $f(\cdot|\mu_g \sigma_g)$ an der Gesamtverteilung f_G an. Im Schnitt kommen also $\pi_g \cdot 100\%$ der Datenpunkte von Gruppe g, $g = 1, \dots, G$. Somit können wir die (latente) Zuordnung z_{ig} als eine Realisation der Zufallsvariablen Z_{ig} mit Wahrscheinlichkeitsfunktion

$$P(Z_{ig} = 1) = \pi_g$$

auffassen.

Wegen der Bedingung $\sum_{g=1}^{G} z_{ig} = 1$, gilt dass

$$Z_{ig} = 1 \quad \Rightarrow \quad Z_{i1} = 0, \dots, Z_{ig-1} = 0, Z_{ig+1} = 0, \dots, Z_{iG} = 0.$$

1.3.2 A-priori und A-posteriori Wahrscheinlichkeiten: π_g und p_{ig}

A-priori-Wahrscheinlichkeit π_g: Man bezeichnet die Wahrscheinlichkeiten π_g als die *a-priori-Wahrscheinlichkeiten*. Wenn wir nichts über die Flossenlänge von Pinguin i wissen, dann bleiben uns nur die a-priori-Wahrscheinlichkeiten:

„Mit Wahrscheinlichkeit $\pi_g = P(Z_{ig} = 1)$ gehört Pinguin i zu Gruppe g."

A-posteriori-Wahrscheinlichkeit p_{ig}: Falls wir die Flossenlänge x_i von Pinguin i erfahren, können wir die a-priori-Wahrscheinlichkeiten mit Hilfe des **Satzes von Bayes** aktualisieren. Dies führt dann zur *a-posteriori-Wahrscheinlichkeit*:

„Mit Wahrscheinlichkeit $p_{ig} = P(Z_{ig=1}|X_i = x_i)$ gehört ein Pinguin i mit Flossenlänge x_i zu Gruppe g.

Satz von Bayes:

$$p_{ig} = \frac{\pi_g f(x_i|\mu_g, \sigma_g)}{f_G(x_i|\pi, \mu, \sigma)}$$

$$= \frac{\overbrace{P(Z_{ig} = 1)}^{\text{„A-priori-Wahrs.“}} \; f(x_i|\mu_g, \sigma_g)}{f_G(x_i|\pi, \mu, \sigma)} = \overbrace{P(Z_{ig} = 1|X_i = x_i)}^{\text{„A-posteriori-Wahrs.“}} = p_{ig}$$

1.3.3 Der (bedingte) Mittelwert: p_{ig}

Beachte: Die a-posteriori-Wahrscheinlichkeiten p_{ig} sind tatsächlich (bedingte) Erwartungswerte:

$$p_{ig} = 1 \cdot P(Z_{ig} = 1|X_i = x_i) + 0 \cdot P(Z_{ig} = 0|X_i = x_i) = E(Z_{ig}|X_i = x_i)$$

Damit ist die Berechnung von p_{ig} im **(Expectation)**-Schritt des EM Algorithmuses (siehe Kapitel 1.2.3) also tatsächlich eine Erwartungswertberechnung.

1.4 Das Große Ganze

Hätten wir neben den Datenpunkten $\mathbf{x} = (x_1, \dots, x_n)$ auch die Gruppenzuordnungen $\mathbf{z} = (z_{11}, \dots, z_{nG})$ beobachtet, dann könnten wir folgende **Likelihood ($\tilde{\mathcal{L}}$) bzw. Log-Likelihood ($\tilde{\ell}$) Funktion** aufstellen:

$$\tilde{\mathcal{L}}(\pi, \mu, \sigma|\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} \prod_{g=1}^{G} \left(\pi_g f(x_i|\mu_g, \sigma_g)\right)^{z_{ig}}$$

$$\tilde{\ell}(\pi, \mu, \sigma|\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \sum_{g=1}^{G} z_{ig} \left\{\ln\left(\pi_g\right) + \ln\left(f(x_i|\mu_g, \sigma_g)\right)\right\}$$

- Im Gegensatz zur ursprünglichen Log-Likelihood Funktion (ℓ), wäre die neue Log-Likelihood Funktion $\tilde{\ell}$ **einfach zu maximieren**, da hier keine Summe innerhalb der Logarithmusfunktion steckt, sodass wir direkt den Logarithmus der Normalverteilung berechnen können. Dies vereinfacht das Maximierungsproblem deutlich, da die Normalverteilung zur Exponentialfamilie gehört.
- Aber: Wir beobachten die Realisationen $\mathbf{z} = (z_{11}, \dots, z_{nG})$ nicht, sondern kennen lediglich die Verteilung der Zufallsvariablen $\mathbf{Z} = (Z_{11}, \dots, Z_{nG})$.

Dies führt zu einer stochastischen Version der Log-Likelihood Funktion:

$$\tilde{\ell}(\pi, \mu, \sigma|\mathbf{x}, \mathbf{Z}) = \sum_{i=1}^{n} \sum_{g=1}^{G} Z_{ig} \left\{ \ln\left(\pi_g\right) + \ln\left(f(x_i|\mu_g, \sigma_g)\right) \right\}$$

- Von dieser können jedoch den bedingten Erwartungswert berechnen:

$$E(\tilde{\ell}(\pi, \mu, \sigma|\mathbf{x}, \mathbf{z})|X_i = x_i) = \sum_{i=1}^{n} \sum_{g=1}^{G} p_{ig} \left\{ \ln\left(\pi_g\right) + \ln\left(f(x_i|\mu_g, \sigma_g)\right) \right\}$$

1.4.1 Der EM Algorithmus: *Die abstrakte Version*

Der folgende EM Algorithmus unterscheidet sich wieder lediglich in der Notation von den oben bereits besprochenen Versionen (siehe Kapitel 1.2.3). Die hier gewählte Notation verdeutlicht, dass der **Expectation**-Schritt die zu maximierende Log-Likelihood Funktion aktualisiert und diese dann im **(Maximization)**-Schritt maximiert wird. Darüber hinaus ist die gewählte Notation abstrakt genug, um die Grundidee des EM Algorithmuses auf andere Maximum Likelihood Probleme zu übertragen. Im Folgenden wird der Parametervektor (π, μ, σ) der Einfachheit halber auch mit θ bezeichnet.

1. Setze Startwerte $\theta^{(0)} = (\pi^{(0)}, \mu^{(0)}, \sigma^{(0)})$
2. Für $r = 1, 2, ...$
   - **(Expectation)** Berechne:

$$\begin{aligned} \mathcal{Q}(\theta, \theta^{(r-1)}) &= E_{\theta^{(r-1)}}(\tilde{\ell}(\pi, \mu, \sigma|\mathbf{x}, \mathbf{z})|X_i = x_i) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ig}^{(r-1)} \left\{ \ln\left(\pi_g\right) + \ln\left(f(x_i|\mu_g, \sigma_g)\right) \right\} \end{aligned}$$

   - **(Maximization)** Berechne:

$$\theta^{(r)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(r-1)})$$

3. Prüfe Konvergenz:
   - Stoppe, falls sich der Wert der maximierten Log-Likelihood Funktion, $\mathcal{Q}(\theta^{(r)}, \theta^{(r-1)})$, nicht mehr ändert.

Ende

Dem gemeinen Pinguin ist der EM Algorithmus egal.

Literaturverzeichnis

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Science & Business Media.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–22.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. Springer Science & Business Media.

Horst, A. M., Hill, A. P., and Gorman, K. B. (2020). *palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*. R Package Version 0.1.0.