

# Computational Statistics (B.Sc. Data Science)

Prof. Dr. Dominik Liebl



# Inhaltsverzeichnis

<b>Informationen</b>	<b>5</b>
<b>1 Der Expectation Maximization (EM) Algorithmus</b>	<b>7</b>
1.1 Motivation: Clusteranalyse mit Hilfe Gaußscher Mischverteilungen	8
1.2 Der EM Algorithmus zur ML-Schätzung Gaußscher Mischverteilungen . . . . .	12
1.3 Der alternative (wahre) Blick auf den EM-Algorithmus . . . . .	17



# Informationen

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

## Vorlesungszeiten

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

**Zoom-Link:** <https://uni-regensburg.zoom.us/j/82236631432?pwd=eWxIZlVqeDdUR2g5YlQ4ZlRPdjZ0dz09>

## Lesecke

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 Florence Nightingale eine britische Krankenschwester und inspirierende Statistikerin.



# Kapitel 1

## Der Expectation Maximization (EM) Algorithmus

Der EM Algorithmus wird häufig verwendet, um komplizierte Maximum Likelihood Schätz-Probleme zu vereinfachen. Wir stellen den Algorithmus zur Schätzung von Gaußschen Mischverteilungen (GMV) vor, da der EM-Algorithmus hier wohl seine häufigste Anwendung hat. Bereits die originale Arbeit zum EM-Algorithmus (Dempster et al., 1977) beschäftigt sich mit solchen Mischverteilungen.

Mögliche Anwendungen von Gaußschen Mischverteilungen:

- Automatisierte Videobearbeitungen: Z.B. Bildeinteilungen in Vorder- und Hintergrund. (Hier würde man jede Pixel-Farbkodierung mit Hilfe einer Gaußschen Mischverteilungen modellieren.)
- 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 erläutern.
- die Grundidee des EM-Algorithmuses erläutern.
- den EM-Algorithmus zur Schätzung von Gaußschen Mischverteilungen anwenden.

## 8KAPITEL 1. DER EXPECTATION MAXIMIZATION (EM) ALGORITHMUS

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

### Begleitliteratur(n)

Zur Vorbereitung der Klausur ist es grundsätzlich ausreichend 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). (Der **Link** führt zur frei erhältlichen pdf-Version des Buches.)

Weiterer guter Lesestoff zum EM Algorithmus gibt es z.B. hier:

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

### R-Pakete für diese Kapitel

Installiere die notwendige R-Pakete für dieses Kapitel:

```
pkgs <- c("tidyverse",      # Die tidyverse-Pakete
          "palmerpenguins", # Pinguin-Daten
          "scales",         # Für transparente Farben: alpha()
          "RColorBrewer",   # Hübsche Farben
          "mclust")         # Schätzung und Verwendung von GMVs

install.packages(pkgs)
```

## 1.1 Motivation: Clusteranalyse mit Hilfe Gaußscher Mischverteilungen

Der folgende Code-Chunck bereitet die Daten auf.

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

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



## 1.1. MOTIVATION: CLUSTERANALYSE MIT HILFE GAUßSCHER MISCHVERTEILUNGEN<sup>9</sup>

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

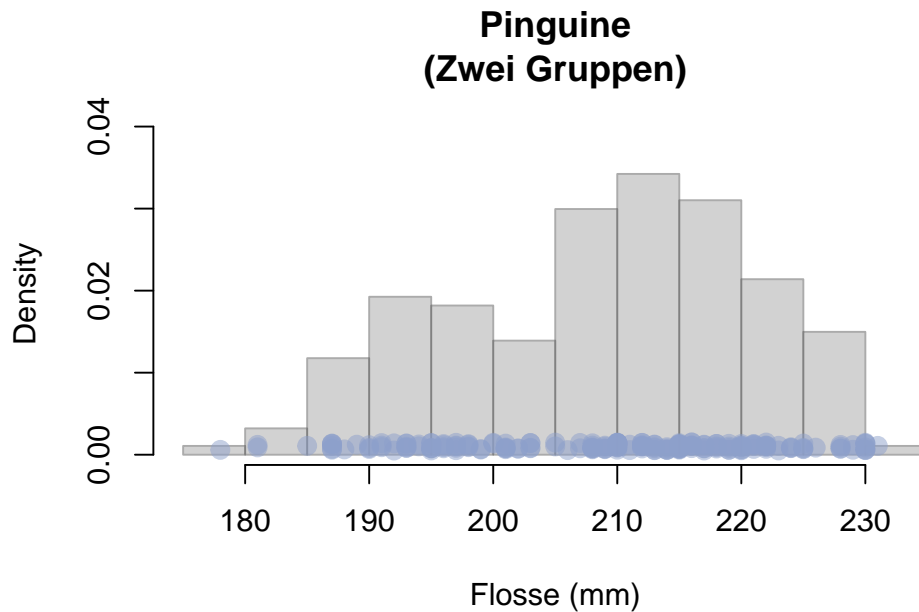
## Vorbereitung der Daten:
Pinguine <- palmerpenguins::penguins %>%           # Pinguin-Daten
  tidyr::as_tibble() %>%                           # Datenformat: 'tibble'-dataframe
  dplyr::filter(species!="Adelie") %>%              # Pinguin-Art 'Adelie' löschen (verbleiben: 'Chin
  droplevels() %>%                                  # 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 'Penguin_Art' aus Pinguine-Daten herausziehen
Penguin_Art <- dplyr::pull(Pinguine, Art)

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

## Plot
## Histogramm:
hist(x = Penguin_Flosse, freq = FALSE,
     xlab="Flosse (mm)", main="Pinguine\n(Zwei Gruppen)",
     col=gray(.65,.5), border=gray(.35,.5), ylim=c(0.0003, 0.039))
## Stripchart hinzufügen:
stripchart(x = Penguin_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:

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

## Anzahl der Gruppen
G <- 2

## Schätzung der Gaußschen Mischverteilung mit Hilfe des EM Algorithmuses
## (inkl. Clusteranalyse)
mclust_obj <- mclust::Mclust(data = Penguin_Flosse, G=G,
                             modelNames = "V",
                             verbose = FALSE)

# summary(mclust_obj)
# str(mclust_obj)
```

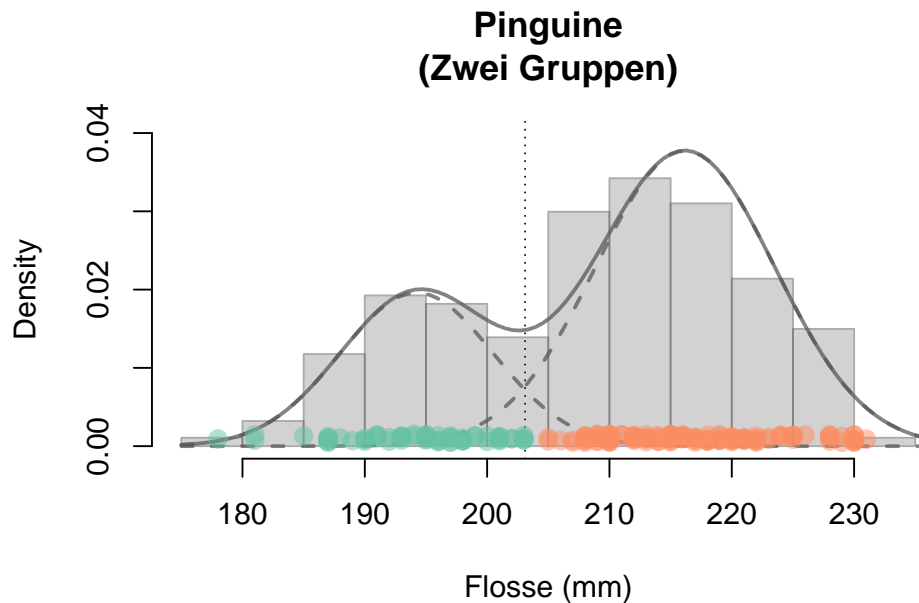


Abbildung 1.1: Clusteranalyse basierend auf einer Mischverteilung mit zwei gewichteten Normalverteilungen.

```
## Geschätzte Gruppen-Zuordnungen
class <- mclust_obj$classification

## Anteil der korrekten Zuordnungen:
# cbind(class, Penguin_Art)
round(sum(class == as.numeric(Penguin_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$sigma_sq[1],
             "Cov2" = mclust_obj$parameters$variance$sigma_sq[2])

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

## Auswerten der Gaußsche Mischungs-Dichtefunktion
np <- 100 # Anzahl der Auswertungspunkte
xxd <- seq(min(Penguin_Flosse)-3, max(Penguin_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 = Penguin_Flosse, xlab="Flosse (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(Penguin_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(Penguin_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)

```

## 1.2 Der EM Algorithmus zur ML-Schätzung Gaußscher Mischverteilungen

### 1.2.1 Gaußsche Mischmodelle (GMM)

Eine Zufallsvariable  $X$ , die einer Gaußschen Mischverteilung folgt, bezeichnen wir als

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

Die dazugehörige Dichtefunktion einer Gaußschen Mischverteilung ist folgendermaßen definiert:

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

- **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:**  $\pi, \mu$  und  $\sigma$

### 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  $\pi, \mu$  und  $\sigma$  und wir müssen ihnen diese Informationen „nur noch“ entlocken.

- **Schätz-Idee:** Wähle  $\pi, \mu$  und  $\sigma$  so, dass  $f_G(\cdot|\pi, \mu, \sigma)$  „optimal“ zu den beobachteten Daten  $\mathbf{x}$  passt.
- **Umsetzung der Schätz-Idee:** Maximiere (bzgl.  $\pi, \mu$  und  $\sigma$ ) 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)

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

**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:

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

**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.

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  $f_g$ , die sich um einen einzigen Datenpunkt  $x_i$  konzentriert (d.h.  $\mu_g = x_i$  und  $\sigma_g \rightarrow 0$ ) wird dabei sehr große Werte annehmen (d.h.  $f_g(x_i) \rightarrow \infty$ ) und so die Log-Likelihood auf unerwünschte maximieren. Solche trivialen Maximierungslösungen resultieren i.d.R. in unplausiblen Schätzergebnissen.

**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:

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

für  $g = 1, \dots, G$ .

Die Herleitung der Ausdrücke für  $\mu_g$ ,  $\sigma_g$  und  $\pi_g$ ,  $g = 1, \dots, G$ , 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:

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

für  $i = 1, \dots, n$  und  $g = 1, \dots, G$ . Erlauben also keine direkte Schätzung der unbekannten Parameter  $\pi$ ,  $\mu$  und  $\sigma$

**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  $\hat{\pi}_g$ ,  $\hat{\mu}_g$  und  $\hat{\sigma}_g$ .

**Der EM Algorithmus:**

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

- **(Expectation)** Berechne:

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

- **(Maximization)** Berechne:

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

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

### 3. Prüfe Konvergenz

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

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

## Daten:
x <- cbind(Penguin_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 Log-Likelihood Wert
##
diff <- abs(loglik - loglikOld)/abs(loglik)
loglikOld <- loglik
it <- it + 1
## Änderung der Log-Likelihood noch groß genug?
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)
#>
#> Gewichte      Gruppe 1 Gruppe 2
#> Mittelwerte   194.26  216.20
#> Standardabweichungen 6.26   7.32

```

Das Schätzergebnis erlaubt es uns, Abbildung 1.1 zu replizieren:

```

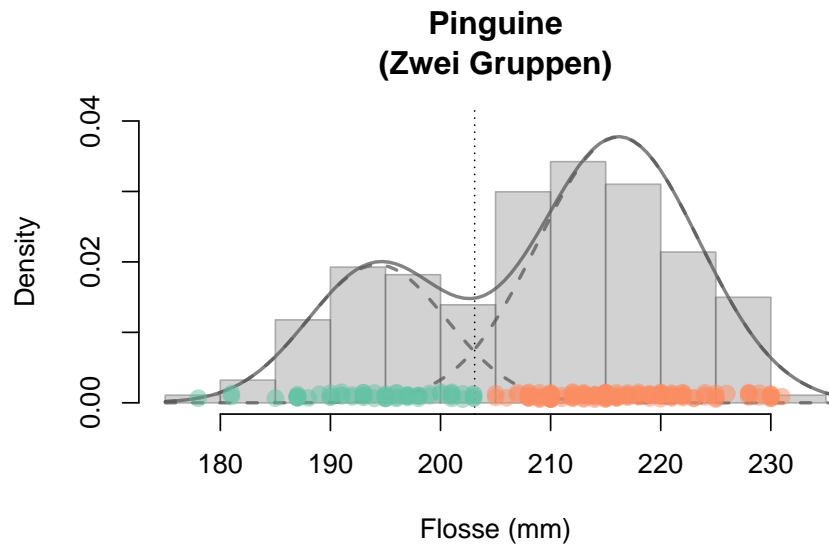
## Auswerten der Gaußsche Mischungs-Dichtefunktion
np <- 100 # Anzahl der Auswertungspunkte
xxd <- seq(min(Penguin_Flosse)-3, max(Penguin_Flosse)+5, length.out = np)
## Mischungs-Dichte
yyd <- dnorm(xxd, mu[1,1], sqrt(sigma)[,1])*pi[1] +
      dnorm(xxd, mu[1,2], sqrt(sigma)[,2])*pi[2]
## Einzel-Dichten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

```



### 1.3. DER ALTERNATIVE (WAHRE) BLICK AUF DEN EM-ALGORITHMUS<sup>17</sup>

```
## Plot
hist(x = Penguin_Flosse, xlab="Flosse (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(Penguin_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(Penguin_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.

Zur Erinnerung: Wie haben es ja nicht geschafft, die Log-Likelihood Funktion

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

direkt zu maximieren. Die  $\log(\sum_{g=1}^G [\dots])$ -Konstruktion macht einem das Leben schwer.

# 18KAPITEL 1. DER EXPECTATION MAXIMIZATION (EM) ALGORITHMUS

In unseren Pinguin-Daten gibt zwei Gruppen ( $g \in \{1, 2\}$ ). Ist gäbe im Prinzip also latente (unbeobachtete) Zuordnungsdaten  $z_{ig}$  mit

$$z_{ig} = \begin{cases} 1 & \text{falls Pinguin } i \text{ zu Gruppe } g \text{ gehört.} \\ 0 & \text{sonst.} \end{cases},$$

wobei  $\sum_{g=1}^G z_{ig} = 1$  für alle  $i = 1, \dots, n$ .

**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 Pinguin-Daten unproblematisch ist, in anderen Anwendungen aber evtl. problematisch sein kann.

Die Zuordnungsdaten  $\mathbf{z} = (z_{11}, \dots, z_{nG})$  sind leider unbekannt (latent). Wir wissen aber trotzdem etwas über diese Zuordnungen. Laut unserem Modell

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

ist die Zuordnung  $z_{ig}$  nämlich eine Realisation einer Bernoulli Zufallsvariable

$$Z_{ig} \sim \mathcal{B}(\pi_g)$$

Also

$$\begin{aligned} P(Z_{ig} = 1) &= \pi_g = P(\text{Pinguin } i \text{ gehört zu Gruppe } g) \\ P(Z_{ig} = 0) &= 1 - \pi_g = P(\text{Pinguin } i \text{ gehört nicht zu Gruppe } g) \end{aligned}$$

Man bezeichnet  $\pi_1, \dots, \pi_G$  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  $\pi_g$  gehört Pinguin  $i$  zu Gruppe  $g$ .

Falls wir die Flossenlänge von Pinguin  $i$  erfahren, können wir die a-priori-Wahrscheinlichkeiten mit Hilfe des Satzes von Bayes aktualisieren. Dies führt dann zur a-posteriori-Wahrscheinlichkeit:

**A-posteriori-Wahrscheinlichkeit**  $p_{ig}$ :

(Satz von Bayes)

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

### 1.3. DER ALTERNATIVE (WAHRE) BLICK AUF DEN EM-ALGORITHMUS 19

Beachte:  $p_{ig}$  ist ein (bedingter) Mittelwert (**Expectation**)

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

#### Der EM Algorithmus

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

- (**Expectation**) Berechne:

$$p_{ig}^{(r)} = E \left( Z_{ig}^{(r-1)} \mid X_i^{(r-1)} = x_i \right)$$

wobei  $Z_{ig}^{(r-1)} \sim \mathcal{B} \left( \pi_g^{(r-1)} \right)$

und  $X_i^{(r-1)} \sim \mathcal{N}_{mix}(G, \pi^{(r-1)}, \mu^{(r-1)}, \sigma^{(r-1)})$

- (**Maximization**) Berechne:  $\hat{\pi}_g^{(r)}$ ,  $\hat{\mu}_g^{(r)}$ ,  $\hat{\sigma}_g^{(r)}$

3. Prüfe Konvergenz

→



# 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.