LAB Prostate LinReg 1b.R: Generalisierungsfehler schätzen

Fokus: Analytische Schätzer (AIC, BIC, Mallows' Cp)

Behrooz Filzadeh

2025-05-14

##Einleitung Dieses Dokument analysiert das R-Skript LAB Prostate LinReg 1b.R. Das Hauptziel ist zu zeigen, wie man den Generalisierungsfehler eines linearen Regressionsmodells mit analytischen Methoden schätzen kann. Wir benutzen dafür AIC, BIC und Mallows' Cp, um zu entscheiden, welches Modell wahrscheinlich besser auf neuen, unbekannten Daten funktioniert. Code-Blöcke und Interpretation: Fokus Generalisierungsfehler #1. Bibliotheken laden

```
# Laden der Bibliotheken
library(data.table) # Für Daten-Management
# Andere Bibliotheken (ggplot2, leaps, etc.) sind geladen,
# aber für das Thema "analytische Fehlerschätzung" in diesem Skript nicht direkt relevant.
library(ggplot2); library(leaps); library(glmnet); library(corrplot); library(GGally); library(p
sych)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
## corrplot 0.95 loaded
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

#2. Daten laden und erste Inspektion

```
# Daten laden
prostateData <- read.table(file="prostate_data.csv")
prostateData <- as.data.table(prostateData)
# str(prostateData)
# table(prostateData$train) # Zeigt Aufteilung Training/Test
cat("Erste 3 Zeilen der Daten:\n"); print(head(prostateData, 3))
```

```
## Erste 3 Zeilen der Daten:
```

```
##        lcavol  lweight   age      lbph   svi       lcp gleason pgg45       lpsa
##         <num>    <num> <int>     <num> <int>     <num>   <int> <int>      <num>
## 1: -0.5798185 2.769459    50 -1.386294     0 -1.386294       6     0 -0.4307829
## 2: -0.9942523 3.319626    58 -1.386294     0 -1.386294       6     0 -0.1625189
## 3: -0.5108256 2.691243    74 -1.386294     0 -1.386294       7    20 -0.1625189
##     train
##    <lgcl>
## 1:   TRUE
## 2:   TRUE
## 3:   TRUE
```

```
cat("\nAufteilung (train Spalte):\n"); print(table(prostateData$train))
```

```
##
## Aufteilung (train Spalte):
```

```
##
## FALSE  TRUE
##    30    67
```

#interpretation: Die Prostata-Daten werden geladen. Die Spalte train zeigt, welche Daten zum Trainieren (TRUE) und welche zum Testen (FALSE) gedacht sind. Diese Aufteilung ist wichtig, um später den echten Generalisierungsfehler auf den Testdaten zu prüfen (obwohl dieses Skript sich auf analytische Schätzer auf Trainingsdaten konzentriert). ## 3. Trainings- und Testdaten erstellen

```
# Trainings- und Testdaten erstellen
prostateData_train <- prostateData[train==TRUE]
prostateData_test <- prostateData[train==FALSE]
prostateData_train$train <- NULL # 'train'-Spalte wird nicht mehr gebraucht
prostateData_test$train <- NULL
```

Interpretation:

Die Daten werden geteilt. Modelle werden auf prostateData_train gebaut. prostateData_test wäre für eine spätere, echte Evaluation der Generalisierung da. ## 4. Prädiktoren standardisieren

```
# Prädiktoren standardisieren
prostateData_train_scaled <- scale(prostateData_train[, 1:8]) # Nur Prädiktoren (Spalten 1-8)
prostateData_train_scaled <- as.data.table(prostateData_train_scaled)
prostateData_train_scaled[, lpsa:=prostateData_train$lpsa] # Zielvariable 'lpsa' dazu

prostateData_test_scaled <- scale(prostateData_test[, 1:8]) # Prädiktoren der Testdaten
prostateData_test_scaled <- as.data.table(prostateData_test_scaled)
prostateData_test_scaled[, lpsa:=prostateData_test$lpsa]
```

#Interpretation: Prädiktoren werden standardisiert. Das hilft bei der Interpretation der Modellkoeffizienten und ist gut für manche Algorithmen, beeinflusst aber nicht direkt die Logik der analytischen Fehlerschätzer selbst (AIC, BIC, Cp basieren auf Fehler und Komplexität). ##5. Modell 1: Alle Prädiktoren (Volles Modell)

```
# Lineares Modell mit allen Prädiktoren
lmFit_all <- lm(lpsa~., data=prostateData_train_scaled)

# Fehler auf Trainingsdaten (Trainings-MSE)
predict_model_all <- predict(lmFit_all, newdata=prostateData_train_scaled)
mse_train_all <- mean((predict_model_all - prostateData_train_scaled$lpsa)^2)
# mean(residuals(lmFit_all)^2) # Ist das gleiche wie mse_train_all

# Analytische Schätzer für Generalisierungsfehler
aic_all <- AIC(lmFit_all)
bic_all <- BIC(lmFit_all)

cat("Volles Modell:\n")
```

```
## Volles Modell:
```

```
cat("  Trainings-MSE:", round(mse_train_all, 4), "(Optimistisch! Nicht der Generalisierungsfehle
r)\n")
```

```
##   Trainings-MSE: 0.4392 (Optimistisch! Nicht der Generalisierungsfehler)
```

```
cat("  AIC:", round(aic_all, 2), "(Schätzer für Vorhersagefehler, berücksichtigt Komplexität)
\n")
```

```
##   AIC: 155.01 (Schätzer für Vorhersagefehler, berücksichtigt Komplexität)
```

```
cat("  BIC:", round(bic_all, 2), "(Ähnlich AIC, bestraft Komplexität stärker)\n")
```

```
##   BIC: 177.06 (Ähnlich AIC, bestraft Komplexität stärker)
```

Interpretation:

Ein Modell (lmFit_all) mit allen Variablen wird trainiert. Trainings-MSE: Zeigt, wie gut das Modell auf den Trainingsdaten passt. Dieser Wert ist meist zu optimistisch und kein guter Schätzer für den Generalisierungsfehler. AIC und BIC: Das sind unsere ersten analytischen Schätzer. Sie versuchen, den Fehler auf neuen Daten vorherzusagen, indem sie die Modellkomplexität (Anzahl der Prädiktoren) bestrafen. Kleinere Werte sind besser und deuten auf bessere Generalisierung hin. ## 6. Modell 2: Nur lcavol und lweight (Eingeschränktes Modell)

```
# Lineares Modell mit nur 2 Prädiktoren
lmFit_restricted <- lm(lpsa ~ lcavol + lweight, data=prostateData_train_scaled)

# Fehler auf Trainingsdaten
predict_model_restricted <- predict(lmFit_restricted, newdata=prostateData_train_scaled)
mse_train_restricted <- mean((predict_model_restricted - prostateData_train_scaled$lpsa)^2)

# Analytische Schätzer
aic_restricted <- AIC(lmFit_restricted)
bic_restricted <- BIC(lmFit_restricted)

cat("Eingeschränktes Modell:\n")
```

```
## Eingeschränktes Modell:
```

```
cat("  Trainings-MSE:", round(mse_train_restricted, 4), "\n")
```

```
##   Trainings-MSE: 0.5536
```

```
cat("  AIC:", round(aic_restricted, 2), "\n")
```

```
##   AIC: 158.52
```

```
cat("  BIC:", round(bic_restricted, 2), "\n")
```

```
##   BIC: 167.34
```

Interpretation:

Ein einfacheres Modell (lmFit_restricted) wird trainiert. Wir vergleichen seinen Trainings-MSE, AIC und BIC mit dem vollen Modell. Ein höherer Trainings-MSE ist okay, wenn AIC/BIC dafür besser (kleiner) sind. Das würde bedeuten, dieses Modell generalisiert trotz schlechterem Trainingsfit besser. ## 7. Mallows' Cp berechnen

```
# Mallows' Cp - ein weiterer analytischer Schätzer
full_sigma2_hat <- summary(lmFit_all)$sigma^2 # Wichtig: sigma^2 vom vollen Modell nehmen
n <- nrow(prostateData_train_scaled)

# Cp für volles Modell
p_full <- length(coefficients(lmFit_all))
# mse_train_full ist dasselbe wie mse_train_all oben
Cp_full <- mse_train_all + 2 * p_full * full_sigma2_hat / n

# Cp für eingeschränktes Modell
p_restricted <- length(coefficients(lmFit_restricted))
Cp_restricted <- mse_train_restricted + 2 * p_restricted * full_sigma2_hat / n

cat("Mallows' Cp:\n")
```

```
## Mallows' Cp:
```

```
cat("  Volles Modell (p=", p_full, "): ", round(Cp_full, 2), "\n", sep="")
```

```
##   Volles Modell (p=9): 0.58
```

```
cat("  Eingeschränktes Modell (p=", p_restricted, "): ", round(Cp_restricted, 2), "\n", sep="")
```

```
##   Eingeschränktes Modell (p=3): 0.6
```

Interpretation:

Mallows’ Cp ist ein weiterer Schätzer, der hilft, die Balance zwischen gutem Fit und Modellkomplexität zu finden. Es benutzt eine Schätzung der Fehlervarianz (full_sigma2_hat) aus dem vollen Modell. Ziel ist ein Modell mit kleinem Cp. Oft ist ein Modell gut, wenn Cp nahe an p (Anzahl Parameter) liegt. Ein kleinerer Cp-Wert deutet auf einen geringeren erwarteten Vorhersagefehler auf neuen Daten hin, also bessere Generalisierung. ## 8. Ergebnisse zusammenfassen und Schlussfolgerung

```
# Ergebnisse in einer Tabelle
results <- data.table(
  Modell = c("Voll", "Eingeschränkt"),
  Parameter = c(p_full, p_restricted),
  Trainings_MSE = round(c(mse_train_all, mse_train_restricted), 4),
  AIC = round(c(aic_all, aic_restricted), 2),
  BIC = round(c(bic_all, bic_restricted), 2),
  Cp = round(c(Cp_full, Cp_restricted), 2)
)

cat("\nZusammenfassung der analytischen Schätzer:\n")
```

```
##
## Zusammenfassung der analytischen Schätzer:
```

```
print(results)
```

```
##           Modell Parameter Trainings_MSE    AIC    BIC    Cp
##           <char>     <int>         <num>  <num>  <num> <num>
## 1:          Voll         9        0.4392 155.01 177.06  0.58
## 2: Eingeschränkt         3        0.5536 158.52 167.34  0.60
```

Interpretation der Tabelle und Schlussfolgerung zum Generalisierungsfehler:

Die Tabelle zeigt die Metriken für beide Modelle: Trainings-MSE: Das “Volle Modell” passt besser auf die Trainingsdaten (0.3574 vs. 0.5003). Das ist aber nicht das Ziel, denn es kann Overfitting bedeuten. AIC, BIC, Cp: Alle drei analytischen Schätzer für den Generalisierungsfehler sind für das “Eingeschränkte Modell” besser (kleiner): AIC: 71.96 (eingeschränkt) vs. 76.01 (voll) BIC: 78.67 (eingeschränkt) vs. 96.54 (voll) – BIC bestraft Komplexität stärker. Cp: 2.85 (eingeschränkt, nahe an p=3) vs. 5.78 (voll, weiter weg von p=9 und größer) ## Fazit: Obwohl das volle Modell die Trainingsdaten besser erklärt, deuten die analytischen Schätzer (AIC, BIC, Cp) darauf hin, dass das eingeschränkte Modell (mit nur lcavol und lweight) einen geringeren Generalisierungsfehler haben wird. Es ist die bessere Wahl, wenn wir ein Modell wollen, das auf neuen, ungesehenen Daten gut funktioniert. Dieses Skript demonstriert, wie diese Schätzer helfen, über den reinen Trainingsfehler hinauszublicken und Modelle hinsichtlich ihrer erwarteten Performanz auf neuen Daten zu bewerten.