

Machine Learning in R

– in 90 Minuten

Jonas Beste & Arne Bethmann

9. Oktober 2015

Einstieg

- ▶ Anwendung von Machine Learning Methoden in R
- ▶ Einstiegsbuch

<http://www-bcf.usc.edu/~gareth/ISL/>

ML-Pakete in R

- ▶ Viele Pakete für einzelnen Machine Learning Methoden (caret, klaR, e1071, rpart, randomForest, u.v.m.)
- ▶ Paket MLR vereint viele Methoden in einen gemeinsamen Framework
- ▶ <https://cran.r-project.org/web/packages/mlr/mlr.pdf>

Validation Set Approach

- ▶ Daten `mtcars` ansehen

```
1 > str(mtcars)
```

- ▶ Set setzen und Sample Index erstellen über `sample` Funktion

```
1 > set.seed(1)
2 > train=sample(32,16)
```

Validation Set Approach

► 1. Model mit linearen Term und Berechnung des MSE

```
1 > modell1=lm(mpg~hp, data=mtcars,  
2 + subset=train)  
3 > mean((mtcars$mpg - predict(modell1, mtcars))  
4 +[-train]^2)
```

► 2. Model mit quadratischem Term

```
1 > modell2=lm(mpg~poly(hp, 2), data=mtcars,  
2 + subset=train)  
3 > mean((mtcars$mpg - predict(modell2, mtcars))  
4 +[-train]^2)
```

► 3. Model mit kubischem Term

```
1 > modell3=lm(mpg~poly(hp, 3), data=mtcars,  
2 + subset=train)  
3 > mean((mtcars$mpg - predict(modell3, mtcars))  
4 +[-train]^2)
```

k-Fold Cross-Validation

- ▶ Daten `iris` ansehen

```
1 > str(iris)
```

- ▶ Trainingsgruppe definieren über `trainControl` Funktion

```
1 > train_control <- trainControl(method="cv",  
2 + number=10)
```

- ▶ Modell trainieren über `train` Funktion

```
1 > model <- train(Species~., data=iris,  
2 + trControl=train_control, method="knn")
```

k-Fold Cross-Validation

- ▶ Vorhersagen machen

```
1 > predictions <- predict(model, iris)
```

- ▶ Ergebnisse zusammenfassen über confusionMatrix Funktion

```
1 > confusionMatrix(predictions, iris$Species)
```

Validation Set Approach

► Aufgabe

1. Verwenden Sie den Datensatz `cars` und stehen Sie per Validation Set Approach fest, ob ein linearer, ein quadratischer oder ein kubischer Term der Variable `speed` am besten die Variable `distance` vorhersagt.

k-nearest-neighbor

- ▶ Zeit und Platz intensiver Algorithmus
- ▶ Beispiel:
 - ▶ mit 2 Klassen A und B
 - ▶ 2 Prediktoren X_1 und X_2
 - ▶ 4 Fälle pro Klasse

k-nearest-neighbor

► Klasse A Fälle

```
1 > A1=c (2, 4)
2 > A2=c (4, 4.5)
3 > A3=c (1, 3.5)
4 > A4=c (2.5, 4)
```

► Klasse B Fälle

```
1 > B1=c (6, 6.5)
2 > B2=c (6, 7)
3 > B3=c (4, 7)
4 > B4=c (5, 7)
```

► Fälle zusammenführen

```
1 > train=rbind(A1,A2,A3,B4,B1,B2,B3,B4)
```

k-nearest-neighbor

- ▶ Klasse zuteilen

```
1 > cl=factor(c(rep("A",4),rep("B",4)))
```

- ▶ Plot von X1 und X2

```
1 > plot(train)
```

- ▶ Zu klassifizierender Fall

```
1 > test=c(5,5)
```

k-nearest-neighbor

► Funktion knn in Paket class

```
1 > install.packages("class")  
2 > library(class)  
3 > help(knn)
```

► Ergebnis der Klassifikation aufrufen

```
1 > summary(knn(train, test, cl, k = 1))  
2 A B  
3 1 0
```

k-nearest-neighbor

► Aufgabe

1. Variiere k
2. Variiere die Werte des Testfalls
3. Erstelle eine Matrix mit 10 Testfällen und klassifiziere die Fälle

Naive Bayes

- ▶ Klassifikation von Blumen im Datensatz `iris`
- ▶ 3 verschiedene Blumenarten
- ▶ Informationen zu Länge und Breite von Stängel und Blüte

```
1 > str(iris)
```

- ▶ Index für Trainings- und Testdaten mit Funktion `createDataPartition` (Paket `caret`)

```
1 > set.seed(1)
2 > trainIndex <- createDataPartition(iris$Species,
3 + p=0.80, list=F)
```

Naive Bayes

- ▶ Trainings- und Testdaten definieren

```
1 > dataTrain=iris[trainIndex, ]  
2 > dataTest=iris[-trainIndex, ]
```

- ▶ Model bestimmen (naiveBayes Funktion) und Testdaten klassifizieren

```
1 > model <- naiveBayes(Species ~ ., data = dataTrain)  
2 > table(predict(model, dataTest), dataTest$Species)
```

► Aufgabe

1. Schauen Sie den Datensatz `mpg` an, erstellen Sie einen Trainings- und einen Testdatensatz (mit 25% der Fälle im Testdatensatz) und klassifizieren Sie die Testdaten hinsichtlich der Variable `class` mit Hilfe des Naive Bayes Klassifikators.

Entscheidungsbäume

- ▶ Es gibt viele verschiedenen Pakete für Entscheidungsbäume
- ▶ Wir nutzen hier zunächst `rpart` mit dem Datensatz `kyphosis`

```
1 > install.packages("rpart")  
2 > library(rpart)  
3 > str(kyphosis)
```

- ▶ Baum wachsen lassen

```
1 > fit <- rpart(Kyphosis ~ Age + Number + Start,  
2 + method="class", data=kyphosis)
```

Entscheidungsbäume

- ▶ Ergebnisse darstellen
- ▶ Wir nutzen hier zunächst `rpart` mit dem Datensatz `kyphosis`

```
1 > printcp(fit)
2 > plotcp(fit)
3 > summary(fit)
```

- ▶ Plot des Baums

```
1 > plot(fit, uniform=TRUE,
2 + main="Classification Tree for Kyphosis")
3 > text(fit, use.n=TRUE, all=TRUE, cex=.8)
```

Random Forest

- ▶ Extra Paket randomForest

```
1 > install.packages("randomForest")  
2 > library(randomForest)
```

- ▶ Jetzt lassen wir einen ganzen Wald wachsen

```
1 > fit <- randomForest(Kyphosis ~ Age+Number+Start,  
2 + data=kyphosis)
```

- ▶ Und schauen uns das Ergebnis an

```
1 > print(fit)  
2 > importance(fit)
```

Herzlichen Glückwunsch!

Sie haben Ihre ersten Machine Learning Methoden in R gerechnet.
Von hier ist es nur noch ein kurzer Weg bis zur Weltherrschaft.

Was kommt als nun?

- ▶ Die erste Hürde ist genommen. Jetzt heißt es weitermachen:
 - ▶ Weitere Bücher: The Elements of Statistical Learning (<http://statweb.stanford.edu/~tibs/ElemStatLearn/>)
 - ▶ Lesen Sie Blogs: z.B. <http://machinelearningmastery.com/blog/> und <http://R-blogger.com>
 - ▶ Besuchen Sie Tutorials: z.B. <http://blog.datacamp.com/machine-learning-in-r/> und <http://https://www.coursera.org/course/datascitoolbox>