Clustering

Ron Wehrens, Christoph Steinbeck, et al.

05/07/2018

Hierarchisches Clustering

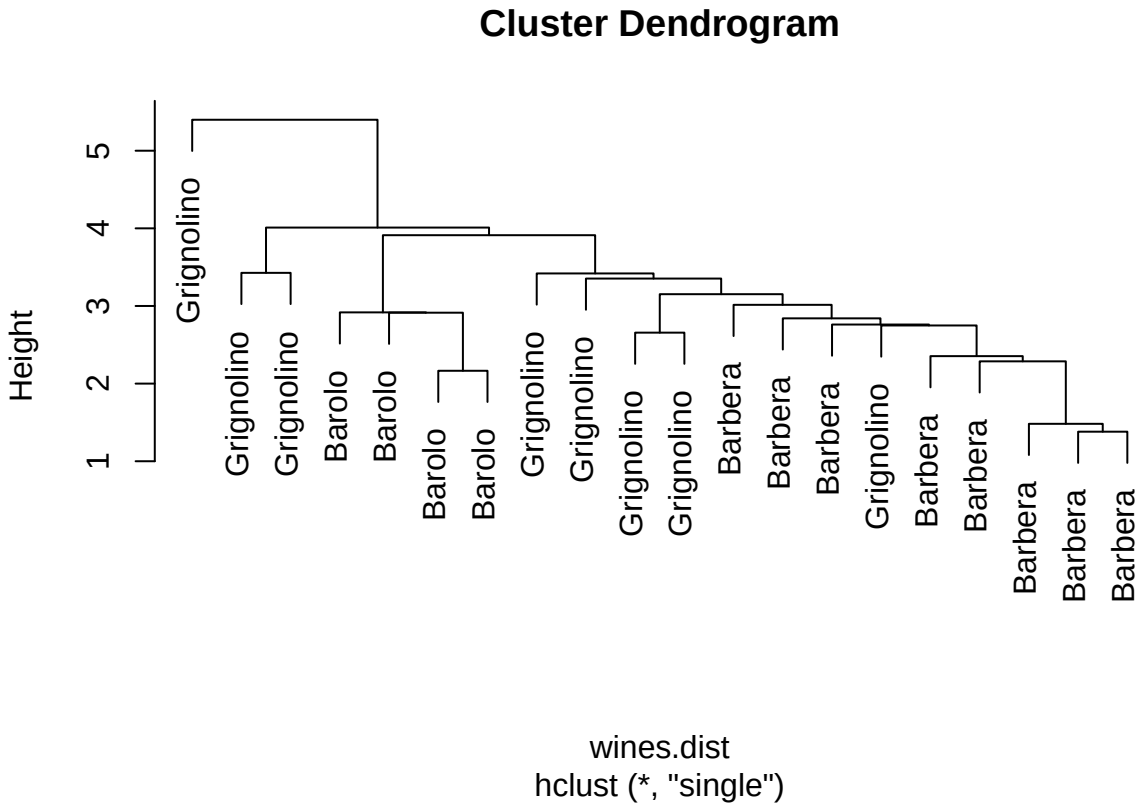
Wir verwenden wieder den Weindatensatz, um ein hierarchisches Clustering zu erstellen. Zunächst versuchen wir es mit der Ähnlichkeitmethode des Single-Linkage.

```
library(kohonen)
data(wines, package = "kohonen")
wines.sc <- scale(wines)
subset <- sample(nrow(wines), 20)
wines.dist <- dist(wines.sc[subset,])
print(wines.dist)
```

```
##           1        2        3        4        5        6        7        8
## 2  7.740045
## 3  1.380414 7.231818
## 4  5.537850 5.857485 5.775046
## 5  4.897111 6.370783 4.911060 2.913354
## 6  5.031559 7.510524 5.360317 3.911643 5.231656
## 7  6.491764 7.067758 6.604776 5.072056 5.353239 5.323910
## 8  2.286802 7.072633 2.382861 5.487436 4.522995 5.031715 6.348526
## 9  1.728437 7.301340 1.481936 5.481980 4.603509 4.978998 6.245608 2.925972
## 10 6.053584 6.033800 6.252509 2.165263 3.747380 4.875514 5.458950 6.000382
## 11 4.161093 7.215072 3.588223 6.519891 4.930071 6.565195 7.216304 2.760675
## 12 3.150745 6.100508 3.071410 4.789798 4.775047 3.958949 5.645210 2.748782
## 13 3.458734 7.145076 3.272443 6.244894 5.963464 5.119329 7.083096 2.840341
## 14 3.813231 9.526859 4.079798 7.267183 6.038813 6.260707 7.277997 3.014628
## 15 7.006043 7.021654 7.087224 3.986711 5.610941 3.418979 5.140411 6.545397
## 16 7.563923 5.398582 7.291735 5.914605 6.204462 6.167934 3.426478 6.747996
## 17 4.276855 6.833649 4.588917 4.008028 4.384634 2.656190 4.009865 4.172864
## 18 6.526273 5.536660 6.475771 3.552372 2.917444 6.380317 5.450830 5.819837
## 19 3.689447 7.134785 3.827289 5.739086 4.736341 5.025751 6.077102 2.353563
## 20 4.160845 7.263791 4.462829 4.744550 5.779851 3.360751 5.557500 5.102865
##           9       10       11       12       13       14       15       16
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10 5.994691
## 11 4.196247 6.689307
## 12 3.289768 5.805939 4.736228
## 13 3.741278 6.585842 3.459953 3.898859
## 14 4.463386 7.699231 4.312211 4.975485 4.689421
## 15 6.807239 4.541282 7.455426 5.745797 6.293362 7.731063
```

```
## 16 7.097351 6.293916 7.104782 5.909817 6.861165 8.175858 5.094479
## 17 3.976566 5.039281 5.961604 3.151825 4.968647 5.357963 4.136368 5.102493
## 18 6.420346 4.223594 6.064307 5.691885 7.128395 7.534591 5.717045 5.808653
## 19 3.636468 6.039442 3.750477 3.488211 4.148929 3.787913 6.583058 6.548363
## 20 3.885501 4.960061 6.875023 4.018983 5.315381 6.424968 5.411564 6.722749
##          17       18       19
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18 5.453919
## 19 3.779349 5.969200
## 20 3.352954 7.178800 5.068067
```

```
wines.hcsingle <- hclust(wines.dist, method = "single")
plot(wines.hcsingle, labels = vintages[subset])
```



Schneiden des Dendrograms

Zunächst liefert uns das hierarchische Clustering keine Cluster. Alle Elemente sind über Pfade miteinander verbunden. Erst das Schneiden des Dendrograms auf einer gewählten Höhe liefert uns eine Clustering, bei dem dann alle in über Pfade miteinander verbundene Elemente in einem Cluster sind. Das Werkzeug hierzu liefert uns die Funktion cutree().

```
wines.cl.single <- cutree(wines.hcsingle, h = 3.3)
table(wines.cl.single, vintages[subset])
```

```
##
## wines.cl.single Barbera Barolo Grignolino
##               1       8      0          3
##               2       0      0          1
##               3       0      4          0
##               4       0      0          1
##               5       0      0          1
##               6       0      0          1
##               7       0      0          1
```

AUFGABE: Wie viele Cluster sind entstanden und wie ist deren Komposition?

AUFGABE: Führen Sie das Clustering mit den gesamten Weindaten aus.

```
wines.dist <- dist(wines.sc)
wines.hcsingle <- hclust(wines.dist, method = "single")
#plot(wines.hcsingle, labels = vintages)
table(vintages, cutree(wines.hcsingle, k = 3))
```

```
##
## vintages      1  2  3
##   Barbera    48  0  0
##   Barolo     58  0  0
##   Grignolino 67  3  1
```

Fast alles Weine sind in Cluster 1. Nicht sehr befriedigend. AUFGABE: Versuchen Sie das gleiche mit den bekannten zwei anderen Ähnlichkeitsmaßen. Bewerten Sie die Ergebnisse.

K-Means Clustering

```
wines.km <- kmeans(wines.sc, centers = 3)
wines.km
```

```
## K-means clustering with 3 clusters of sizes 61, 65, 51
##
## Cluster means:
##      alcohol malic acid        ash ash alkalinity   magnesium tot. phenols
## 1  0.8333649 -0.3013131  0.3661731     -0.6065538  0.56922228   0.88768039
## 2 -0.9183253 -0.3953334 -0.4905017      0.1637039 -0.48321576  -0.07114136
## 3  0.1736447  0.8642504  0.1871775      0.5168437 -0.06497127  -0.97106500
##    flavonoids non-flav. phenols     proanth  col. int.   col. hue
## 1  0.98016451       -0.56173008  0.57583669  0.1702296  0.4753467
## 2  0.02658937       -0.03709561  0.06509498 -0.8955790  0.4614076
## 3 -1.20624204        0.71915195 -0.77171004  0.9378162 -1.1566204
##     OD ratio    proline
## 1  0.7753334  1.1296451
```

```
## 2  0.2823571 -0.7460740
## 3 -1.2872265 -0.4002655
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 1 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [176] 3 3
##
## Within cluster sum of squares by cluster:
## [1] 382.1859 559.2978 326.3534
##  (between_SS / total_SS =  44.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

AUFGABE: Geben Sie die Clustermitgliedschaften als Tabelle aus. Tipp: Wie schon oben verwenden Sie den table()-Befehl. Lesen Sie die Hilfe, um zu verstehen, was der table() aus dem base-Paket macht. Vergleichen Sie das Ergebniss mit den Ergebnissen der Hierarchischen Clustering und mit den Ergebnissen der PCA.

Modellierung

Lineare Diskriminanzanalyse (LDA)

Zur Demonstration teilen wir den Weindatensatz hälftig in Trainings- und Testdaten:

```
odd <- seq(1, nrow(wines), by = 2)
even <- seq(2, nrow(wines), by = 2)
wines.trn <- wines[odd,]
wines.tst <- wines[even,]
```

Nun führen wir ein Training auf den TRN daten aus:

```
wines.counts <- table(vintages[odd])
ngroups <- length(wines.counts)
wines.groups <- split(as.data.frame(wines.trn),vintages[odd])
wines.covmats <- lapply(wines.groups, cov)
wines.wcovmats <- lapply(1:ngroups, function(i, x, y) x[[i]]*y[i], wines.covmats, wines.counts)
wines.pooledcov <- Reduce("+", wines.wcovmats) / (nrow(wines.trn) - ngroups)
```

Dieses Stück Code veranschaulicht eine praktische Eigenschaft der lapply-Funktion: Wenn das erste Argument ein Vektor ist, kann es als Index für eine Funktion verwendet werden, die auch andere Argumente, hier, eine Liste und einen Vektor enthält. Jede der drei Kovarianzmatrizen wird mit einem Gewicht multipliziert, das der Anzahl der Objekte in dieser Klasse entspricht. Im nächsten Schritt fügt die Funktion reduce() die drei gewichteten Kovarianzmatrizen hinzu.

Die Anzahl der Parameter, die in der LDA geschätzt werden müssen, ist relativ gering: die gepoolte Kovarianzmatrix enthält p(p+1)/2 Zahlen und jeden Parameter des Clusterzentrums p. Bei G-Gruppen führt dies zu insgesamt Gp + p(p+1)/2 Schätzungen { für die Weindaten, mit drei Gruppen und dreizehn Variablen, diese impliziert 130 Schätzungen. Die LDA-Klassifizierung selbst ist nun einfach durchzuführen:

Zuerst berechnen wir die Mahalanobis-Abstände (mit der Mahalanobis-Funktion) zu den drei Klassenzentren unter Verwendung der gepoolten Kovarianzmatrix, und dann bestimmen wir, welche von ihnen für jede Probe im Trainingsset am nächsten sind.

```
distances <- sapply(1:ngroups, function(i, samples, means, covs) mahalanobis(samples, colMeans(means[[i]
trn.pred <- apply(distances, 1, which.min)
```

Lassen Sie uns schau'n, wie gut die Vorhersagen sind - zunächst für die Trainingsdaten, die wir ja schon in der Model hineingesteckt haben:

```
table(vintages[odd], trn.pred)
```

```
##             trn.pred
##               1  2  3
##   Barbera    24  0  0
##   Barolo      0 29  0
##   Grignolino  0  0 36
```

Und nun für die Test-Daten:

```
distances <- sapply(1:ngroups, function(i, samples, means, covs) mahalanobis(samples, colMeans(means[[i]
tst.pred <- apply(distances, 1, which.min)
table(vintages[even], tst.pred)
```

```
##             tst.pred
##               1  2  3
##   Barbera    24  0  0
##   Barolo      0 29  0
##   Grignolino  1  0 34
```

Hier wurde also offensichtlich ein Grignolino falsch klassifiziert - im Vergleich zum Clustering immer noch ein sehr gutes Resultat.