Mischmodelle in R

Achim Zeileis

2009-02-20

1 Installation von flexmix

In von R standardmäßig geladenen Paketen gibt es keine direkte Implementierung von Mischmodellen, allerdings stehen im Comprehensive R Archive Network CRAN (http://CRAN.R-project.org/) mehrere Erweiterungspakete zur Verfügung, mit Hilfe derer dies möglich ist.

Für die Vorlesung wurde das Paket **flexmix** (Leisch, 2004) verwendet.

Dieses (und andere) Zusatzpaket(e) läßt sich sehr leicht installieren, wenn man eine Internetzverbindung hat: Man klickt in R einfach auf 'Packages' und dann auf 'Install package(s) from CRAN...'. Dies öffnet dann einen Dialog, in dem man zunächst einen CRAN-Mirror auswählen muß (bspw. den an der TU-Wien, die CRAN Hauptseite) und dann aus einer Liste die gewünschten Pakete selektiert. Diese werden dann automatisch installiert.

Wenn man auf dem Rechner, wo R läuft, keine Anbindung an das Internetz hat, kann man sich die zugehörige ‚.zip'-Datei 'von Hand' von CRAN herunterladen, auf den eigenen Rechner spielen und dann im 'Packages'-Menü auf 'Install package(s) from local zip files' klicken. Die Pakete findet man auf http://CRAN.R-project.org/ unter > Software > Packages.

Nach der Installation kann das Paket einfach per

```
R> library(flexmix)
```

geladen werden.

2 Beispiel: Künstliche NPreg Daten

Der Datensatz `NPreg` ist im Paket **flexmix** enthalten und kann per

```
R> data(NPreg)
```

verfügbar gemacht werden. Er enthält verschiedene künstlich generierte Variablen

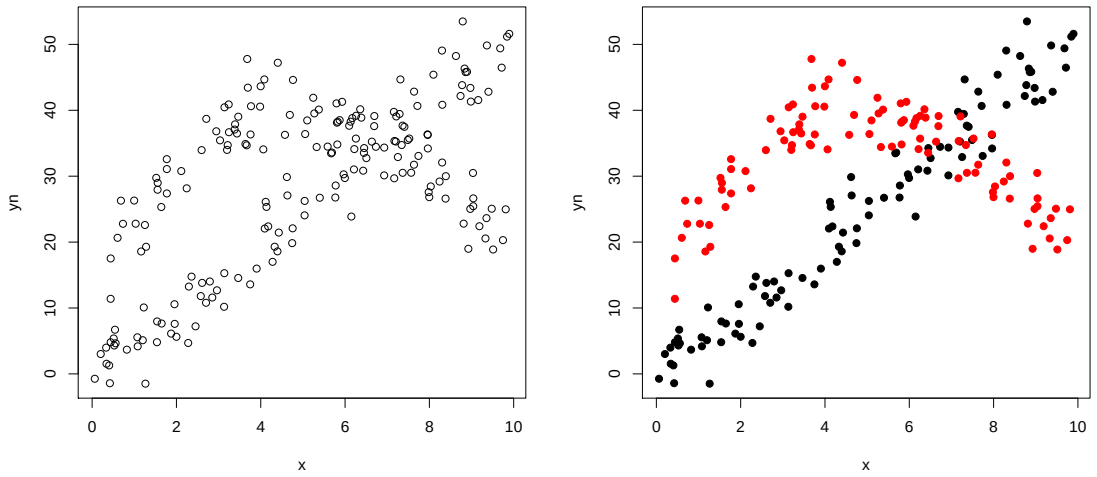
```
R> summary(NPreg)
```

```
       x                yn               yp              yb
 Min.   :0.06295   Min.   :-1.482   Min.   : 0.000   Min.   :0.000
 1st Qu.:2.67892   1st Qu.:20.489   1st Qu.: 2.000   1st Qu.:0.000
 Median :5.07820   Median :30.933   Median : 4.000   Median :0.000
 Mean   :5.03078   Mean   :28.607   Mean   : 3.955   Mean   :0.475
 3rd Qu.:7.37686   3rd Qu.:38.322   3rd Qu.: 5.000   3rd Qu.:1.000
 Max.   :9.89474   Max.   :53.487   Max.   :12.000   Max.   :1.000
```

```
     class          id1           id2
 Min.   :1.0   1      :  2   1      :  4
 1st Qu.:1.0   2      :  2   2      :  4
 Median :1.5   3      :  2   3      :  4
 Mean   :1.5   4      :  2   4      :  4
 3rd Qu.:2.0   5      :  2   5      :  4
 Max.   :2.0   6      :  2   6      :  4
               (Other):188   (Other):176
```

Hier sind wir daran interessiert `yn` durch `x` zu erklären. Die wahre Klasse, aus der die Beobachtung stammen, ist auch bekannt (`class`), wird aber bei der Schätzung des Modells nicht verwendet.

```
R> plot(yn ~ x, data = NPreg)
R> plot(yn ~ x, data = NPreg, pch = 19, col = NPreg$class)
```



Um in beiden Komponenten ein quadratisches Modell zu schätzen, muß man also ein lineares Regressionsmodell mit Formel `yn ~ x + I(x^2)` anpassen. Der zweite Term `I(x^2)` enthält dann die quadrierten Werte von `x` (**Anmerkung:** Er muß in `I()` eingepackt werden, damit der Operator `^` seine herkömmliche arithmetische Bedeutung hat). Diese Formel übergeben wir einfach `flexmix` und geben zusätzlich noch die gewünschte Anzahl von Komponenten an:

```
R> fm <- flexmix(yn ~ x + I(x^2), data = NPreg, k = 2)
```

Dies reicht aus, da ein latentes lineares Regressionsmodell die Voreinstellung in `flexmix` ist.
Das Modell liefert eine sehr knappe Zusammenfassung per

```
R> fm

Call:
flexmix(formula = yn ~ x + I(x^2), data = NPreg, k = 2)

Cluster sizes:
  1   2
100 100

convergence after 17 iterations
```

und die Parameterschätzungen der beiden Komponenten können durch folgende Befehle erfragt werden:

```
R> parameters(fm, component = 1)

                     Comp.1
coef.(Intercept) 14.7166497
coef.x            9.8472717
coef.I(x^2)      -0.9684196
sigma             3.4790480

R> parameters(fm, component = 2)

                      Comp.2
coef.(Intercept) -0.21033135
coef.x            4.81874275
coef.I(x^2)       0.03606276
sigma             3.47733228
```

Diese enthalten zusätzlich zu den Regressionskoeffizienten auch jeweils noch einen Schätzer der Fehlervarianz. Eine ausführlichere Zusammenfassung zusammen mit den gewohnten Teststatistiken für die Koeffizienten wird folgendermaßen berechnet:

```
R> summary(refit(fm))

$Comp.1
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) 14.71311    1.32377  11.114 < 2.2e-16 ***
x            9.84835    0.59158  16.648 < 2.2e-16 ***
I(x^2)      -0.96852    0.05526 -17.526 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

$Comp.2
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.208538   1.009205 -0.2066   0.8363
x            4.814473   0.509609  9.4474   <2e-16 ***
I(x^2)       0.036539   0.049755  0.7344   0.4627
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

aus der abzulesen ist, daß nur in einer Komponente der quadratische Term signifikant ist, in der anderen würde eine einfache lineare Regression schon ausreichen.

3 Beispiel: `seizure` Daten

Die `seizure` Daten, die die Informationen über die Behandlung eines epileptischen Patienten enthalten, sind ebenfalls direkt in **flexmix** verfügbar

```
R> data(seizure)
R> summary(seizure)
```

```
    Seizures          Hours         Treatment       Day
 Min.   : 0.00   Min.   : 3.000   No : 27   Min.   :  1.00
 1st Qu.: 1.00   1st Qu.: 7.000   Yes:113   1st Qu.: 35.75
 Median : 6.50   Median : 8.000             Median : 70.50
 Mean   :14.65   Mean   : 9.021             Mean   : 70.50
 3rd Qu.:22.25   3rd Qu.:12.000             3rd Qu.:105.25
 Max.   :72.00   Max.   :16.000             Max.   :140.00
```

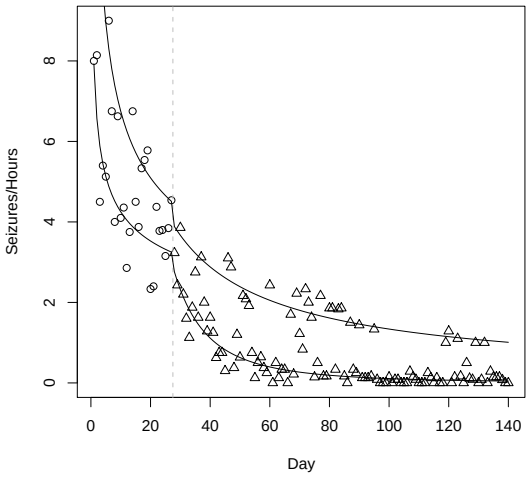
Da es sich hier um Zähldaten handelt, ist ein Poisson-GLM naheliegend, in der die Anzahl der epileptischen Anfälle `Seizures` erklärt wird durch die logarithmierte Dauer der Betreuung `log(Day)` und einem Faktor `Treatment`, der die Behandlung durch Injektionen kodiert. Außerdem soll um den Effekt bereinigt werden, daß die Dauer der elterlichen Betreuung von Tag zu Tag variierte (`Hours`). Dies kann leicht getan werden, indem man um die Stunden vorher durch einen sogenannten Offset bereinigt, d.h. man zieht vorher die Anzahl der Stunden (hier auf einer log-Skala) einfach ab.

Das Mischmodell wird dann geschätzt durch

```
R> fm <- flexmix(Seizures ~ Treatment * log(Day), data = seizure,
+      k = 2, model = FLXglm(family = "poisson", offset = log(seizure$Hours)))
```

Warum die Komponenten des GLMs über `FLXglm` spezifiziert werden müssen ist in Leisch (2004) im Detail erklärt. Hier wollen wir uns nur das angepaßte Modell einmal anschauen.

```
R> plot(Seizures/Hours ~ Day, pch = as.numeric(Treatment), data = seizure)
R> abline(v = 27.5, lty = 2, col = "grey")
R> matplot(seizure$Day, fitted(fm)/seizure$Hours, type = "l", add = TRUE,
+      col = 1, lty = 1)
```



Weitere Informationen gibt es auf der zugehörigen Hilfeseite `help("seizure")` und in Leisch (2004).

Literatur

Leisch F (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8). URL http://www.jstatsoft.org/v11/i08/.