

4. Übungsblatt zur Vorlesung Ökonometrie

Aufgabe 1) L^2 - und L^1 -Regression: Gegeben seien n Datenpunkte

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n).$$

In der Vorlesung haben wir die folgenden Probleme betrachtet:

$$\begin{aligned} \text{(i)} \quad F_{L^2}(\beta_0, \beta_1) &:= \sum_{i=1}^n [y_i - (\beta_0 + \beta_1 x_i)]^2 \xrightarrow{!} \min \\ \text{(ii)} \quad F_{L^1}(\beta_0, \beta_1) &:= \sum_{i=1}^n |y_i - (\beta_0 + \beta_1 x_i)| \xrightarrow{!} \min \end{aligned}$$

Dabei hatten wir gesehen, dass das Problem (i) explizit lösbar ist, β_0 und β_1 sind gegeben durch die Lösung des folgenden Gleichungssystems

$$A \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = b \quad (1)$$

mit

$$A = \begin{pmatrix} \vec{e} \cdot \vec{e} & \vec{e} \cdot \vec{x} \\ \vec{e} \cdot \vec{x} & \vec{x} \cdot \vec{x} \end{pmatrix}, \quad b = \begin{pmatrix} \vec{e} \cdot \vec{y} \\ \vec{x} \cdot \vec{y} \end{pmatrix} \quad (2)$$

und den Abkürzungen $\vec{e} := (1, 1, 1, \dots, 1, 1) \in \mathbb{R}^n$, $\vec{x} = (x_1, x_2, \dots, x_n)$, $\vec{y} = (y_1, y_2, \dots, y_n)$ und dem Standard-Skalarprodukt

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^n x_i y_i,$$

also etwa $\vec{e} \cdot \vec{e} = \sum_{i=1}^n 1^2 = n$. Das Problem (ii) ist nicht explizit lösbar, aber wir hatten gesehen, dass die β 's folgendes Gleichungssystem erfüllen müssen:

$$A_\varepsilon \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = b_\varepsilon \quad (3)$$

mit

$$A_\varepsilon := \begin{pmatrix} \langle \vec{e}, \vec{e} \rangle_\varepsilon & \langle \vec{e}, \vec{x} \rangle_\varepsilon \\ \langle \vec{e}, \vec{x} \rangle_\varepsilon & \langle \vec{x}, \vec{x} \rangle_\varepsilon \end{pmatrix}, \quad b_\varepsilon = \begin{pmatrix} \langle \vec{e}, \vec{y} \rangle_\varepsilon \\ \langle \vec{x}, \vec{y} \rangle_\varepsilon \end{pmatrix} \quad (4)$$

und

$$\langle \vec{x}, \vec{y} \rangle_\varepsilon := \sum_{i=1}^n \frac{x_i y_i}{\varepsilon_i}$$

wobei $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ mit

$$\varepsilon_i := |y_i - (\beta_0 + \beta_1 x_i)| = \varepsilon_i(\beta_0, \beta_1),$$

die epsilons hängen also auch von den β 's ab. Starten Sie jetzt eine R-Session und führen Sie die folgenden Berechnungen durch:

- a) Erzeugen Sie $n = 21$ Datenpunkte $(x_1, y_1), \dots, (x_n, y_n)$, indem Sie etwa den folgenden Code eingeben,

```
x = seq(from = -5, to = 5, by = 0.5)
n = length(x)
noise = rnorm(n)
y = 3 - 0.5 * x + noise
```

und schauen Sie sich die Daten dann mit dem Befehl `plot(x,y)` an.

- b) Legen Sie die Funktionen F_{L^2} und F_{L^1} aus (i) und (ii) an, indem Sie etwa folgendes Code-Fragment verwenden,

```
FL2 = function(beta0,beta1)
{
  res = sum( (y - beta0 - beta1 * x)^2 )
  return( res )
}
```

und etwas analoges für die Funktion F_{L^1} .

- c) Bevor wir die Gleichungssysteme $A\beta = b$ und $A_\varepsilon\beta = b_\varepsilon$ lösen, wollen wir das Minimum von F_{L^2} und F_{L^1} dadurch bestimmen, dass wir einfach verschiedene β 's in die F 's einsetzen und uns dann anschauen, wo die F 's am kleinsten werden. Da wir die Minima in der Gegend von $(\beta_0, \beta_1) = (3, -0.5)$ erwarten, legen wir etwa folgende Test-Betas an:

```
beta0 = seq(from = 0, to = 6, by = 0.01)
beta1 = seq(from = -3, to = 3, by = 0.01)
```

Mit dem folgenden Code

```
n0 = length(beta0)
n1 = length(beta1)

matFL2 = matrix(0,n0,n1)
matFL1 = matrix(0,n0,n1)

for(i in 1:n0)
{
  for(j in 1:n1)
  {
    matFL2[i,j] = FL2(beta0[i],beta1[j])
    matFL1[i,j] = FL1(beta0[i],beta1[j])
  }
}

contour(beta0,beta1,matFL2)
contour(beta0,beta1,matFL2,nlevels=50)

contour(beta0,beta1,matFL1)
contour(beta0,beta1,matFL1,nlevels=50)
```

können Sie sich dann einen Überblick über die Funktionen F_{L^2} und F_{L^1} verschaffen.

- d) Die Minima selber und die Werte von β_0 und β_1 , an denen jeweils das Minimum angenommen wird, können Sie sich dann mit dem folgenden Code anschauen:

```

minFL2 = min(matFL2)
minFL1 = min(matFL1)
minFL2
minFL1

# find position in matrix where
# the minimum is attained:
indicesL2 = which(matFL2 == minFL2, arr.ind = TRUE)
indicesL2
indicesL2[1]
indicesL2[2]

# also:
beta0_L2min = beta0[ indicesL2[1] ]
beta0_L2min # sieht ok aus
beta1_L2min = beta1[ indicesL2[2] ]
beta1_L2min # sieht ok aus

```

Für die Zahlen `beta0_L1min` und `beta1_L1min` müssen Sie noch entsprechenden Code ergänzen.

Jetzt wollen wir die β 's bestimmen, indem wir die Gleichungssysteme $A\beta = b$ und $A_\varepsilon\beta = b_\varepsilon$ lösen. Wir betrachten zunächst den L^2 -Fall (i), für den wir β_0 und β_1 explizit berechnen können, indem wir das Gleichungssystem $A\beta = b$, gegeben in den Gleichungen (1) und (2) weiter oben, lösen:

- e) Berechnen Sie die Skalarprodukte $\vec{e} \cdot \vec{e}$, $\vec{e} \cdot \vec{x}$, $\vec{x} \cdot \vec{x}$, $\vec{e} \cdot \vec{y}$ und $\vec{x} \cdot \vec{y}$ in \mathbb{R} und legen Sie dann die Matrix A und den Vektor b an.
- f) Das Gleichungssystem $A\beta = b$ können Sie dann etwa mit dem `solve()`-Befehl lösen. Finden Sie heraus, wie die genaue Syntax lautet. Vergleichen Sie dann Ihre Lösung mit den Zahlen `beta0_L2min` und `beta1_L2min` aus Teil (d). Ihre Lösung wird nicht ganz exakt mit den Zahlen aus (d) überein stimmen, warum nicht? Die Abweichungen sollten von der Größenordnung 0.01 sein.
- g) Im Falle der L^1 -Regression müssen wir das Gleichungssystem gegeben durch (3) und (4) weiter oben lösen. Da die Matrix A_ε und der Vektor b_ε aber ebenfalls von den β 's abhängen, ist durch $A_\varepsilon^{-1}b_\varepsilon$ keine explizite Lösung gegeben. Aber man kann folgende iterative Methode ausprobieren, um eine Lösung zu erhalten: Wir initialisieren

$$\beta^{(0)} = \begin{pmatrix} \beta_0^{(0)} \\ \beta_1^{(0)} \end{pmatrix} := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (5)$$

und dann wenden wir etwa 100 Mal die Iteration

$$\beta^{(k+1)} = A_{\varepsilon(\beta^{(k)})}^{-1} b_{\varepsilon(\beta^{(k)})} \quad (6)$$

an. Programmieren Sie jetzt diese iterative Methode, indem Sie etwa folgendes Code-Fragment benutzen:

```

# L^1-Regression:

betas = c(0,0)
e = rep(1,n)
for(k in 1:100)
{
  eps = abs( y - betas[1] - betas[2]*x )
  eps = pmax( eps , 10^(-12) )
  ee = sum(e*e/eps)
  ex = sum(e*x/eps)
  xx = ...
  ...
  ...
  b = c(ey,...)
  a1 = c(ee,ex)
  a2 = ...
  A = rbind(a1,a2)
  betas = solve( ... )
  print(betas)
}
beta0 = betas[1]
beta1 = betas[2]
beta0
beta1

```

An den mit ... gekennzeichneten Stellen müssen Sie noch entsprechenden Code einfügen. Vergleichen Sie dann Ihre Lösung mit den Zahlen `beta0_L1min` und `beta1_L1min` aus Teil (d).

- h) In dem obigen Code-Fragment haben wir den `pmax()`-Befehl benutzt, wieso? Informieren Sie sich mit `?pmax()` über diesen Befehl. Was ist der Unterschied zum `max()`-Befehl?
- i) In R gibt es eine eingebaute Funktion, die die lineare Regression durchführt, genauer, die lineare L^2 -Regression. Das ist die Funktion `lm()`. Dabei stehen die Buchstaben `lm` für "Linear Model". Das Problem (i) von oben können Sie damit einfach lösen, indem Sie den Befehl

$$\text{lm}(y \sim x)$$

eingeben. Die Zahlen, die Sie so erhalten, sollten exakt mit der Lösung von $A\beta = b$ aus Teil (f) übereinstimmen. Geben Sie dann noch folgende Befehle ein, um sich etwas mit der `lm()`-Funktion vertraut zu machen:

```

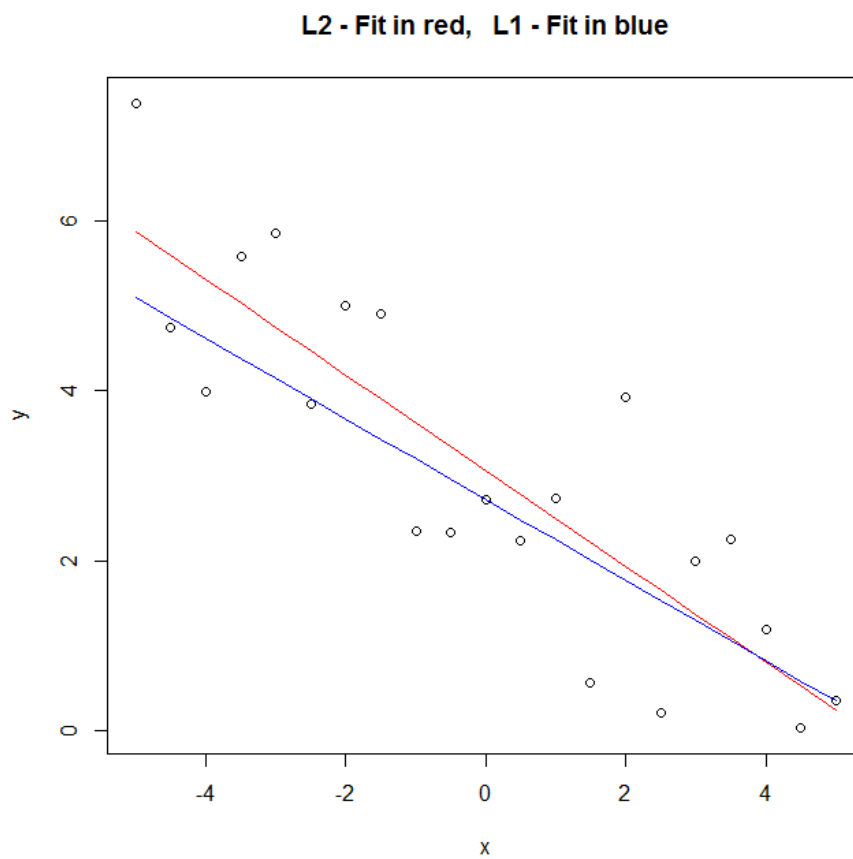
res = lm( y ~ x )
res
summary(res)

names(res)
res$coef
res$fit

mode(res)
str(res)

```

- j) Stellen Sie schliesslich den L^2 -Regression-Fit und den L^1 -Regression-Fit zusammen mit den Datenpunkten graphisch dar:



Wie bekommt man eine Überschrift " L^2 -Fit in rot, L^1 -Fit in blau" in das Plot-Fenster?