CMM, Predictive Modeling og Machine Learning
Øvelsesopgaver, uge 7

Peter Dalgaard

13. marts 2022

Opgave 1

Vi ser igen på datasættet `SAhearts`, som kan downloades fra `https://web.stanford.edu/~hastie/ElemStatLearn` eller Canvas. De to første spørgsmål er næsten identisk med sidste uges øvelser, blot gentaget her for at kunne sammenligne med resultater fra `locfit`,

1. Indlæs datasættet til en dataframe med fx `read.csv`
2. Fit en logistisk regression med naturlige kubiske splines for en af de prædiktorerne (fx `sbp`) og tegn den fittede kurve på sandsynlighedsskalaen (NB: Brug `predict(...., type="response")`, ellers fås værdier på link-skala, dvs logit p).
3. Brug `locfit()` til at estimere og tegne en tilsvarende kurve på den samme figur.
4. Tegn et punktvist konfidensbånd for kurven (se `help(predict.locfit)` og `help(preplot.locfit)` for opskriften).
5. Prøv at justere på båndbredden for `locfit` via `nn=` og/eller `h=` argumenterne (se eksemplerne under `help(locfit)`).

Opgave 2

Denne opgave går ud på at udføre krydsvalidering på et syntetisk datasæt

1. Definer

```
x <- seq(0, 1.5*pi, length=500)
y0 <- sin(x)
y <- y0 + rnorm(500,sd=.2)
plot(x,y, col="grey")
lines(x,y0, col="red", lwd=2)
```

og forklar hvad koden gør.

2. Fit en kurve med `locfit` og indtegn den på plottet. Prøv også med en reduceret båndbredde.
3. Vi kan lave en tilfældig opdeling i 50 delsæt sådan her: (forklar…)

```
z <- sample(rep(1:50,10))
```

4. En konstruktion svarende til formel (7.48) i bogen kan laves som følger

```
ymi <- y
for ( i in 1:50)
  ymi[z==i] <- predict(locfit(y~x, subset=(z != i)), newdata=x[z==i])
plot(x, ymi, pch=".")
```

Herefter kan man beregne CV størrelsen svarende til en kvadratisk tabsfunktion, simpelthen som kvadratsummen af `y - ymi`.

5. Hvis man deler `ymi` op efter grupperne i `z`, og beregner det gennemsnitlige tab i hver gruppe, så kan man også forsyne CV-scoren med en standard error, på samme måde som i bogens Figure 7.9 (vink: brug `tapply`). Prøv dette og forklar hvorfor det giver mening.
6. Modificer koden, så man kan ændre båndbredden i `locfit` og udregne CV-score for et sæt af mulige båndbredder

Opgave 3

Datafilen `covid.csv` indeholder oplysninger om antal test-positive med COVID-19 i perioden fra 1. april 2020 til 14. marts 2022.

Datasættet indeholder følgende variable:

date: Dato (yyyy-mm-dd)

dateno: Løbenummer for dato (1–713)

ntest: Antal testede personer (som ikke tidligere har været testet positiv)

pos: Antal nye positivt testende.

Vi ønsker at beskrive det tidsmæssige forløb af epidemien, idet der må korrigeres for antallet af udførte test N_i. Dette er en kompliceret problemstilling fordi antallet af personer der søger test dels afhænger af smitteniveauet, dels afhænger af om pågældende er i risiko for at være smittet. Man kan derfor ikke antage simpel proportionalitet med N_i. Ifølge en rapport for SSI kan man antage at effekten på det observerede antal positive er proportional med $N_i^{0.7}$.

1. Indlæs data med fx `read.csv2`. Check med `summary`.

2. Benyt i første omgang `locfit` til at estimere `pos` som funktion af `dateno`. Tegn rådata og den fittede kurve.

3. Bemærk at kurven er alvorligt oversmoothed, og tilmed giver negative forventede værdier af `pos`, som jo er et tælletal. Vi kan ændre på udglatningen ved at „skrue" på `nn=` parameteren. Prøv fx `nn=0.05`

4. Data er tælletal, så det var måske mere oplagt at fitte en Poissonmodel med overspredning. Den tilsvarende familie hedder `"qpoisson"` (ikke quasipoisson som i `glm`).

5. Det ændrer næsten ikke fittet at bruge en quasipoisson-model i stedet for almindelig least squares, men det har den fordel at log-link er præcis den transformation der gør det nemt at tage højde for antal test. Vi har jo

   $$\mu_t = N_t^{0.7} f(t) \implies \log \mu_t = 0.7 \log N_t + \log f(t)$$

   hvori $0.7 \log N_t$ optræder som et *offset*. Dette kan specificeres ved at tilføje `base=0.7*log(ntest)` i `locfit`. Man kan også bruge `0.7*log(ntest/100000)`, hvorved den fittede kurve bliver det man ville forvente ved 100000 daglige test (bemærk af også dette er lidt anderledes end i `glm`).

   Fit kurven og tegn den. Hvis man vil sammenligne med data, bør man bruge de justerede antal, `pos/(ntest/100000)^0.7`.

6. Det fittede funktion er egentlig $\log f(t)$, men den bliver automatisk transformeret til antals-skalaen når der plottes. Man kan specificere transformationen via argumentet `tr` til `plot.locfit`. Hvis man bruger `tr=function(x)x` fås den utransformerede version. Prøv dette og indtegn `log(pos/(ntest/100000)^0.7)` til sammenligning.

7. Den afledede af $\log f(t)$ er essentielt det såkaldte *kontakttal*, eller rettere, kontakttallet er med god approksimation lig med

   $$\exp(4.7 \times \frac{\partial}{\partial t} \log f(t)) \approx 1 + 4.7 \times \frac{\partial}{\partial t} \log f(t),$$

hvor 4.7 er længden af en gennemsnitlig smitteperiode. `locfit` kan bringes til at estimere den afledede ved at man skriver `deriv=1`. Prøv at tegne denne afledede (det er desværre ikke længere umiddelbart muligt at sammenholde kurven med data fra `https://covid19.ssi.dk/overvagningsdata`)

8. Hvis det skal være helt fint kan man bruge `tr=` mekanismen til at regne om til det approksimative kontakttal. Man kan også gøre de tegnede kurver pænere ved `where="data"`, ellers beregnes værdierne kun på et grovere gitter. Sidstnævnte kræver, at man bruger konstruktionen `plot(preplot(locfit(....)))`, hvor `where=` er et argument til `preplot`. (Se evt. `help(preplot.locfit)`).