STAT40810 — Stochastic Models

Brendan Murphy

Week 4

Cross Validation (Some Theory & Practice)

Background

- It can be shown (beyond the scope of this module) that the mathematics underlying fitting smoothing splines is very similar to regression.
  This may not be a surprise given how the spline regression worked.
- We want to minimize the cross validated mean squared error.

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}_{\lambda}^{(-i)}(x_i)]^2,$$

where $\hat{f}_{\lambda}^{(-i)}(\cdot)$ is the fitted smoothing spline when observation i is deleted.

More Background

- It turns out that smoothing splines fitting can be expressed as a generalized least squares problem.
- And that

$$CV(\lambda) = \frac{1}{n} \sum \left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - h_{ii}} \right)^2,$$

  where

  - B is a matrix of spline basis functions evaluated for the data,
  - Ω is dependent on the penalty term,
  - h_{ii} are the diagonal elements of the hat matrix $H = B(B'B + \lambda\Omega)^{-1}B'$.

Computational Advantage

- The form of $CV(\lambda)$ is important, because it only involves terms derived from the model fit using all of the data.
  Also, note the similarity of the hat matrix to that for ridge regression.
- Thus, we can find the cross validation error of the fit from a single fit of the spline smoothing to <u>all</u> of the data.
- Thus, the value of λ that minimizes $CV(\lambda)$ can be found very efficiently.

Generalized Cross Validation

- An alternative criterion for choosing λ in spline smoothing is to minimize the *generalized cross validation* error which has the form

$$GCV(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - \text{trace}(H)/n} \right),$$

where $\text{trace}(H)$ is the sum of the diagonal elements of H.

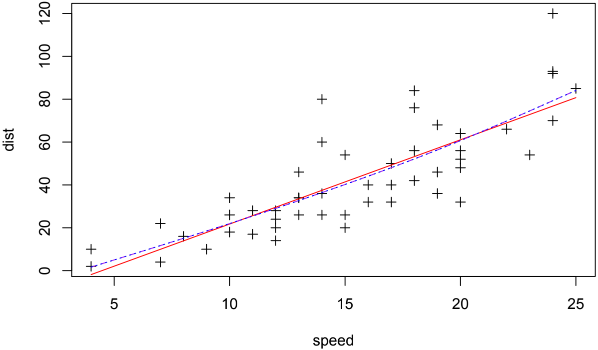
Code: Breaking Distance

- Here's code to do spline smoothing for the breaking distance data (using CV and GCV).

```
#Load the cars data
data(cars)

# Plot the data
plot(cars,pch=3)

# Fit the model using CV
fit1 <- smooth.spline(cars$speed,cars$dist,cv=TRUE)
points(predict(fit1),type="l",col="red")

#Fit the model using GCV
fit2 <- smooth.spline(cars$speed,cars$dist,cv=FALSE)
points(predict(fit2),type="l",col="purple",lty=2)

# An alternative function for GCV
library(mgcv)
fit3 <- gam(dist~s(speed),data=cars)
points(cars$speed,predict(fit3),type="l",col="blue",lty=3)

# Assess fit
mean(residuals(fit1)^2)
mean(residuals(fit2)^2)
```

Example: Breaking Distance

- Suppose that we use CV and GCV to choose the smoothing parameter. What does the fit look like?



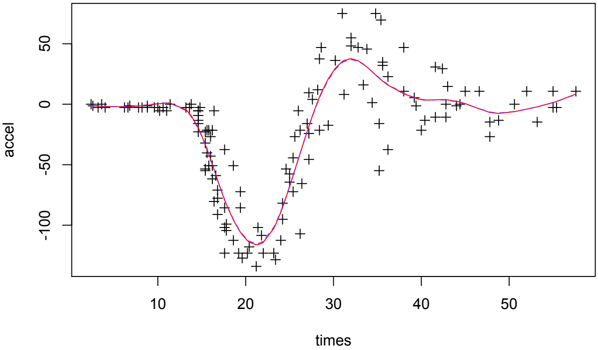
Code: Motorcycle Crash

- Here's code to do spline smoothing for the motorcycle crash data (using CV and GCV).

```
#Load the motorcycle data
library("MASS")
data(mcycle)

# Plot the data
plot(mcycle,pch=3)

# Fit the model using CV
fit1 <- smooth.spline(mcycle$times,mcycle$accel,cv=TRUE)
points(predict(fit1),type="l",col="red")

#Fit the model using GCV
fit2 <- smooth.spline(mcycle$times,mcycle$accel,cv=FALSE)
points(predict(fit1),type="l",col="purple",lty=2)

# Assess fit
mean(residuals(fit1)^2)
mean(residuals(fit2)^2)
```

Example: Motorcycle Crash

- Suppose that we use CV and GCV to choose the smoothing parameter. What does the fit look like?



Code: Cholostyramine Data

- Here's code to do spline smoothing for the cholostyramine data (using CV and GCV).

```
#Load the cholostyramine data
library("bootstrap")
data(cholost)
help(cholost)

# Plot the data
plot(cholost,pch=3)

# Fit the model using CV
fit1 <- smooth.spline(cholost$z,cholost$y,cv=TRUE)
points(predict(fit1),type="l",col="red")

#Fit the model using GCV
fit2 <- smooth.spline(cholost$z,cholost$y,cv=FALSE)
points(predict(fit2),type="l",col="purple",lty=2)

# Assess fit
mean(residuals(fit1)^2)
mean(residuals(fit2)^2)
```

Example: Cholostyramine Data

- Suppose that we use CV and GCV to choose the smoothing parameter. What does the fit look like?