HW5 - Lucas Fellmeth, Sven Bergmann

2023-11-16

1

The goal of this problem is to estimate the regression function of acceleration vs time for the `mcycle` data in the package `MASS`.

A

Show that the Nadaraya-Watson estimator can be expressed as $\hat{Y} = HY$. Find the "hat matrix" H explicitly.

We know that the Nadaraya-Watson estimator of $\hat{m}(x_i)$ is defined by

$$\hat{m}(x_i) = \frac{\sum_{j=1}^{n} K_h(x_j - x_i)Y_j}{\sum_{k=1}^{n} K_h(x_k - x_i)},$$

where

$$K_h(x) = \frac{1}{h}K\left(\frac{x}{h}\right)$$

with h as associated bandwidth.

So

$$\begin{aligned}
\hat{m}(x_i) &= \frac{\sum_{j=1}^{n} K_h(x_j - x_i)Y_j}{\sum_{k=1}^{n} K_h(x_k - x_i)} \\
&= \frac{\sum_{j=1}^{n} \frac{1}{h}K(\frac{x_j - x_i}{h})Y_j}{\sum_{k=1}^{n} \frac{1}{h}K(\frac{x_k - x_i}{h})} \\
&= \frac{\frac{1}{h}\sum_{j=1}^{n} K(\frac{x_j - x_i}{h})Y_j}{\frac{1}{h}\sum_{k=1}^{n} K(\frac{x_k - x_i}{h})} \\
&= \frac{\sum_{j=1}^{n} K(\frac{x_j - x_i}{h})Y_j}{\sum_{k=1}^{n} K(\frac{x_k - x_i}{h})} \\
&= \frac{\sum_{j=1}^{n} K(\frac{x_j - x_i}{h})}{\sum_{k=1}^{n} K(\frac{x_k - x_i}{h})}Y_j \\
&= \sum_{j=1}^{n} \underbrace{\frac{K(\frac{x_j - x_i}{h})}{\sum_{k=1}^{n} K(\frac{x_k - x_i}{h})}}_{H_{ij}} Y_j
\end{aligned}$$

$$\Longrightarrow \underbrace{\begin{pmatrix} \hat{m}(x_1) \\ \vdots \\ \hat{m}(x_n) \end{pmatrix}}_{\hat{Y}} = \underbrace{\begin{pmatrix} H(x_1, x_1) & \dots & H(x_1, x_n) \\ \vdots & \ddots & \vdots \\ H(x_n, x_1) & \dots & H(x_n, x_n) \end{pmatrix}}_{H} \cdot \underbrace{\begin{pmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{pmatrix}}_{Y}$$

B

For a reasonable range of bandwidths h, compute and plot the generalized cross validation measure $GCV(h)$ and find the optimal bandwidth.

```
library(MASS)
library(splines)
```

First, we implement the generalized cross validation measure $GCV(h)$ which is defined by

$$GCV(h) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{Y_i - \hat{m}_h(x)}{1 - \frac{trH(h)}{n}} \right]^2 .$$

Therefore, we have to compute the "hat-matrix".

Kernels

Inside the function for computing this matrix we can use different kernels, which are defined below.

```
sin_cos_exp_kernel <- function(x) {
    return((1/2) * exp(-abs(x)/sqrt(2)) * sin(abs(x)/2 + pi/4))
}

normal_kernel <- function(x) {
    return(dnorm(x))
}

epanechnikov_kernel <- function(x) {
    ifelse(abs(x) > 1, return(0), return(3/4 * (1 - x^2)))
}
```

Hat-matrix

This function computes the "hat-matrix" using vectorized operations for faster results.

```
hat_matrix <- function(x, h) {
    n <- length(x)
    hatmat <- matrix(0, n, n)
    for (i in 1:n) {
        denominator <- sum(normal_kernel((x - x[i])/h))
        hatmat[i, ] <- normal_kernel((x - x[i])/h)/denominator
    }
    return(hatmat)
}
```

Generalized cross validation measure

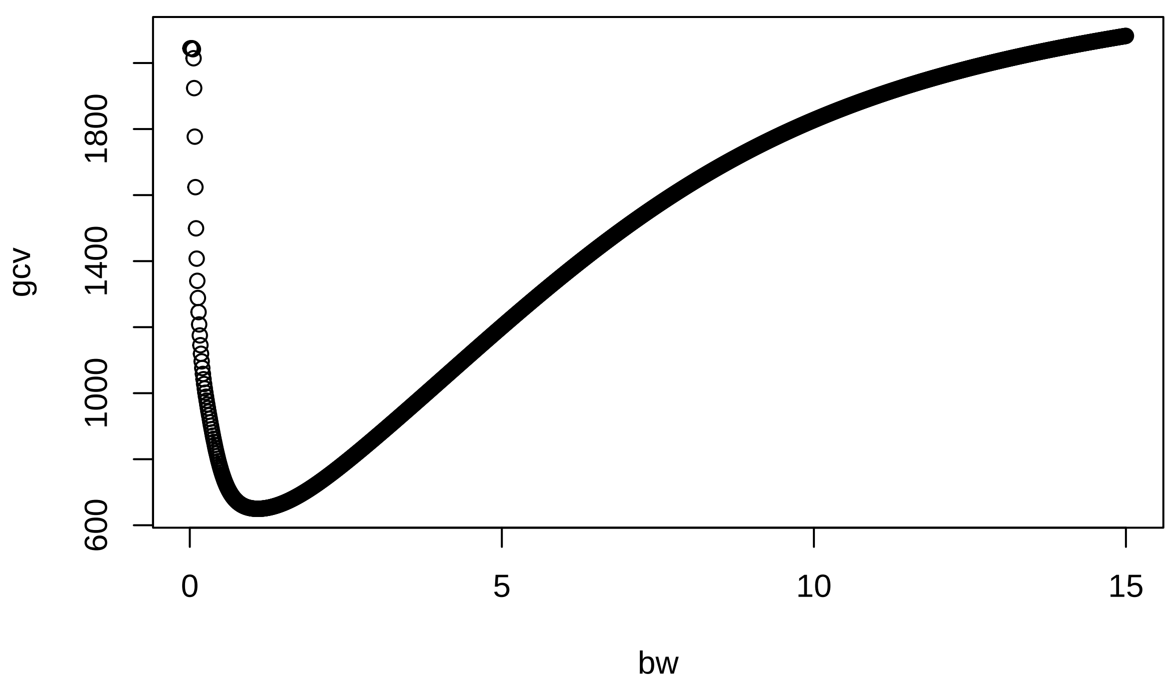
Here, we compute the GCV based on the definition from the book.

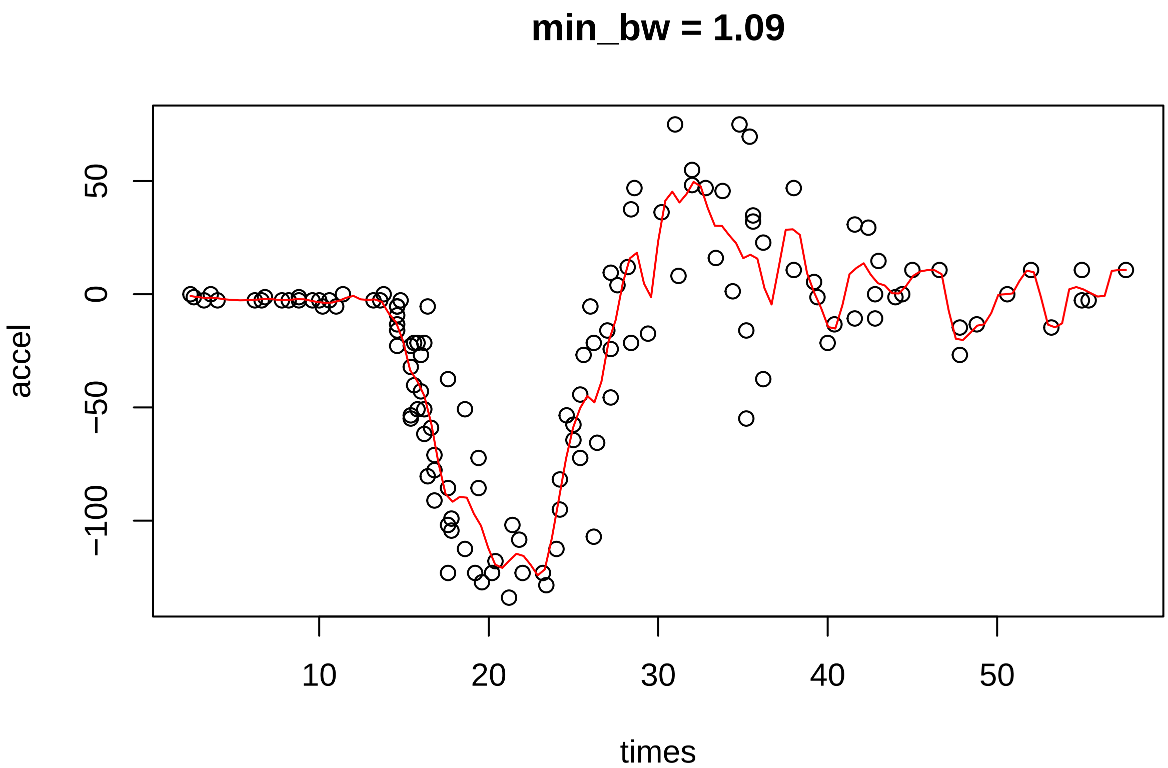
```r
GCV <- function(x, y, h) {
    H <- hat_matrix(x, h)
    y_hat <- H %*% y
    gcv <- mean((y - y_hat)^2/(1 - (sum(diag(H))/length(y)))^2)
    return(gcv)
}
```

Finding best bw

In the code below we tried to replace the kernel which we used for computing the "hat-matrix" to find the best result. We found that the gaussian kernel produced the best result.

```r
with(mcycle, {
    bw <- seq(0.01, 15, by = 0.01)
    gcv <- sapply(bw, GCV, x = times, y = accel)
    tmp <- data.frame(bw = bw, gcv = gcv)
    plot(tmp)
    plot.new()
    plot(times, accel, main = paste("min_bw =", tmp[which.min(tmp$gcv), ]$bw))
    lines(ksmooth(x = times, y = accel, kernel = "normal", bandwidth = tmp[which.min(tmp$gcv),
        ]$bw), col = "red", type = "l")
})
```




min_bw = 1.09
accel
times
50
0
−50
−100
10
20
30
40
50