Smoothing Spline (SS)

erlee

November 7, 2019

CV for the SS

Here, we would like to check if the SS satisfies the sufficient conditions (3)-(4), which implies the equqation (2) for the CV holds in the lecture note 1.

step1: compute the SS.

First we make R functons which give the smoothing matrix (S), design matrix (D) and penalty matrix W for SS, and the basis functions, that is, the cubic spline basis functions with knots equalto data points x_i's.

```
matrices_SS<-function(x,lam){
sx<-sort(x)
n<-length(x)
D<-cbind(rep(1,n), sx)

for (i in 1:(n-2)){

temp<-(sx-sx[i])^3*((sx-sx[i])>0)-(sx-sx[n])^3*((sx-sx[n])>0)
temp<-temp/(sx[n]-sx[i]) #d_i
temp1<-(sx-sx[n-1])^3*((sx-sx[n-1])>0)-(sx-sx[n])^3*((sx-sx[n])>0)
temp1<-temp1/(sx[n]-sx[n-1]) #d(n-1)

D<-cbind(D,temp-temp1)
}

z<-seq(0,1,0.0001)
J<-length(z)

B<-cbind(rep(1,J), z)

for (i in 1:(n-2)){

temp<-(z-sx[i])^3*((z-sx[i])>0)-(z-sx[n])^3*((z-sx[n])>0)
temp<-temp/(sx[n]-sx[i]) #d_i (z)

temp1<-(z-sx[n-1])^3*((z-sx[n-1])>0)-(z-sx[n])^3*((z-sx[n])>0)
temp1<-temp1/(sx[n]-sx[n-1]) #d(n-1)(z)

B<-cbind(B,temp-temp1)
}

#as a result, we get B:J x no. of basis functions containing Bk(zi) as (i,k) entry

#diff
Bd1<-matrix(NA, J-2, n)
for (j in 2:(J-1)){
```

```
Bd1[j-1,]<-(B[j+1,]-B[j-1,])/(z[j+1]-z[j-1]) #the first derivative of basis functions
}

#diff
Bd2<-matrix(NA, J-4, n)
for (j in 2:(J-3)){
Bd2[j-1,]<-(Bd1[j+1,]-Bd1[j-1,])/(z[j+1]-z[j-1]) #the second derivative of basis functions
}

W<-(t(Bd2)%*%Bd2)/(J-4 )


S<-D%*% solve(t(D)%*% D+lam*W) %*% t(D)
return(list(D=D, W=W, S=S))
}

# Basis function for given data
basis<-function(data,x){
  sx<-sort(x)
  n<-length(x)
  D<-c(1,data)
    for (i in 1:(n-2)){
    temp<-(data-sx[i])^3*((data-sx[i])>0)-(data-sx[n])^3*((data-sx[n])>0)
    t
emp<-temp/(sx[n]-sx[i]) #d_i
    temp1<-(data-sx[n-1])^3*((data-sx[n-1])>0)-(data-sx[n])^3*((data-sx[n])>0)
    temp1<-temp1/(sx[n]-sx[n-1]) #d(n-1)
    D<-c(D,temp-temp1)
    }
  return(D)
}
```

step 2: simulation check for (3)-(4)

Set any sample size n, generate data x_i from any distribution. When setting any λ and any i for (4) check if (3)-(4) holds. For example, choose $n = 5$, $x_i \sim U(0,1)$, $\lambda = 1$, $i = 3$ as follows:

```
n<-10
x<-sort(runif(n))
lam=1

i=3
```

```
S<-matrices_SS(x,lam)$S
apply(S,1,sum) #(3)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
xminusi<-x[-i]

Sminusi<-matrices_SS(xminusi,lam)$S
Wminusi<-matrices_SS(xminusi,lam)$W
Dminusi<-matrices_SS(xminusi,lam)$D
```

```r
lminusi_xi<-basis(x[i], xminusi)

lminusi_xi%*% solve(t(Dminusi)%*% Dminusi+lam*Wminusi) %*% t(Dminusi) #LHS of (4)
```

```
##           [,1]     [,2]      [,3]      [,4]       [,5]       [,6]
## [1,] 0.2877091 0.227533 0.1625128 0.1298018 0.09282575 0.08095179
##            [,7]       [,8]        [,9]
## [1,] 0.07651218 -0.0195141 -0.03833229
```

```r
S[i,-i]/sum(S[i,-i]) #RHS of (4)
```

```
## [1]  0.28777857  0.22749655  0.16248016  0.12977913  0.09281421  0.08094378
## [7]  0.07650549 -0.01949260 -0.03830529
```

The sufficient conditions (3)-(4) seem to hold for the SS if ignoring the numeical errors.