Constrained curve fitting based on splines

- Isotonic regression (保序迴歸). Consider the regression model:

$$Y_i = f(X_i) + \varepsilon_i,$$

$i = 1, \ldots, n$. Suppose that f is a monotone function, and we approximate f using a spline function. Then we would like to solve the constrained optimization problem:

$$\text{minimize } \sum_{i=1}^{n}(Y_i - \sum_{j=1}^{J} a_j B_j(X_i))^2 \tag{1}$$

under the constraint that

$$\sum_{j=1}^{J} a_j B_j$$

is an increasing function, where $B_1, \ldots, B_J$ are B-spline basis functions.

- **Fact 1** *For a quadratic spline function* $f_a = \sum_{j=1}^{J} a_j B_j$, *where* $a = (a_1, \ldots, a_J)$ *and* $B_1, \ldots, B_J$ *are B-spline basis functions (in order),* f_a *is increasing if and only if* $\{a_j\}_{j=1}^{J}$ *is increasing.*

Example 1. Generate an increasing sequence $\{a_j\}_{j=1}^{J}$ and plot the graph of $f_a = \sum_{j=1}^{J} a_j B_j$, where $B_1, \ldots, B_J$ are B-spline basis functions on $[0, 1]$. Also, plot the graph of the derivative function of f_a to see if $f'_a \geq 0$.

  - Plot f_a with generated a:

```
require("splines")
m <- 3
knt4 <- c(1:4)/5
a <- sort(runif(7))
fa <- function(x){
 bx <- bs(x, knots=knt4, Boundary.knots=c(0,1), deg=m-1, intercept=TRUE)
 ans <- bx %*% a
 return(ans[,1])
}
curve(fa, 0,1)
```

– Plot the derivative function. The derivative function of a B-spline basis function can be computed using splineDesign.

```
dbs <- function(x, knotlist, bknots, m=4, der=0){
 J <- m+length(knotlist)
 n <- length(x)
 knots.all <- c(rep(bknots[1], m), knotlist, rep(bknots[2], m))
 dbx <- matrix(0, n, J)
 for (j in 1:J){
  k <- knots.all[j:(j+m)]
  dbx[,j] <- splineDesign(k, x, ord=m, derivs=rep(der,n), outer.ok=TRUE)
 }
 return(dbx)
}

f1a <- function(x){
 bx <- dbs(x, knt4, c(0,1), m=m, der=1)
 ans <- bx %*% a
 return(ans[,1])
}

curve(f1a, 0,1)
#lines(c(0,1), c(0,0))
```

- Fitting the isotonic regression model based on quadratic spline approximation and reparametrization. To construct $(a_1, \ldots, a_J)$ so that $\{a_j\}_{j=1}^J$ is strictly inreasing, we can take a sequence $\{b_j\}_{j=1}^J$ and take

$$a_1 = b_1 \text{ and } a_j = a_{j-1} + b_j^2 \text{ for } j \geq 2.$$

Then the constrained optimization problem in (1) can be reduced to the unconstrained problem

$$\text{minimize } \sum_{i=1}^n (Y_i - \sum_{j=1}^J a_j(b) B_j(X_i))^2$$

as a function of $b = (b_1, \ldots, b_J)$.

- Example 2. Generate $Y_i = f(X_i) + \varepsilon_i$ with $f(x) = (x - 0.5)_+^2$ for $x \in (-\infty, \infty)$ as follows:

```
set.seed(1)
n <- 1000
x <- seq(0, 1, length=n)
f <- function(x){ ans <- (x-0.5)^2; ans[x<0.5] <- 0; return(ans) }
y <- f(x)+rnorm(n, sd=0.05)
```

Approximate f using an increasing quadratic spline with 7 equally spaced knots in $(0, 1)$ and find $\hat{f}$: the least squared estimator of f. Find the ISE.

Sol.

```
## define a function for parameter transform
b_to_a.fun <- function(b){
 b1 <- c(0, b[-1])
 a <- b[1]+cumsum(b1^2)
 return(a)
}

## define a function to find the least square fit given the data
get.fit <- function(x,y, knotlist, deg=2, bknots =c(0,1)){
 bx <- bs(x, knots=knotlist, deg=deg, Boundary.knots=bknots, intercept=TRUE)
 #define rss as a function of parameter b
 rss <- function(b){
  a <- b_to_a.fun(b)
  residual <- y - as.numeric(bx%*% a)
  return( sum(residual^2) )
 }
 nb <- length(knotlist)+deg+1
 b0 <- rep( mean(y), nb)
 #b0 is an initial value of b
 opt <- optim(b0, rss)
 a.hat <- b_to_a.fun(opt$par)
 #a.hat: vector of the estimated coefficients of B-spline basis functions
 fhat <- function(u){
  bu <- bs(u, knots=knotlist, deg=deg, Boundary.knots=bknots, intercept=TRUE)
  ans <- bu %*% a.hat
  return(ans[,1])
  }
 return(fhat)
}
```

```
#compute and plot fhat
fhat <- get.fit(x,y,(1:7)/8)
plot(x,y)
curve(fhat, 0, 1, add=TRUE, col=2)
curve(f, 0,1, add=TRUE, col=3)

#compute ISE
g <- function(x){ (fhat(x)-f(x))^2 }
integrate(g,0,1)$value
```

- **Fact 2** *For a quadratic spline function* $f_c = \sum_{j=1}^J c_j B_j$*, where* $c = (c_1, \ldots, c_J)^T$ *and* $B_1, \ldots, B_J$ *are B-spline basis functions with knots* $\xi_1, \ldots, \xi_K$ *and boundary knots* a *and* b*,* f_c *is increasing if and only if* $f_c'(a) \geq 0$*,* $f_c'(\xi_1) \geq 0, \ldots, f_c'(\xi_K) \geq 0$*,* $f_c'(b) \geq 0$*.*

- Fitting the isotonic regression model based on quadratic spline approximation using quadratic programming. According to Fact 2, the constrained optimization problem in (1) can be simplied to

$$\text{minimize} \sum_{i=1}^{n}(Y_i - \sum_{j=1}^{J} c_j B_j(X_i))^2$$

subject to

$$\sum_{j=1}^{J} c_j B_j'(\xi_k) \geq 0$$

for $k = 0, \ldots, K+1$, where $\xi_0 = a$ and $\xi_{K+1} = b$. The above constrained problem is a quadratic programming problem and can be solved using the function `solve.QP` in the R package "quadprog". To see this, let B be the $n \times J$ matrix whose (i,j)-th element is $B_j(X_i)$, let A_0 be the $(K+2) \times J$ matrix whose (i,j)-th element is $B_j'(\xi_{i-1})$ and let c be the $J \times 1$ vector whose j-th element is c_j, then the above constrained problem is to find c so that

$$-2y^T Bc + c^T B^T Bc$$

is minimized subject to $A_0 c \geq 0$, where $y = (Y_1, \ldots, Y_n)^T$ is an $n \times 1$ vector.

- The function `solve.QP` is used for solving the problem of minimizing $(-d^T b + 0.5 b^T D b)$ with the constraints $A^T b \geq b_0$. Let `Dmat, dvec, Amat, bvec` denote D, d, A, b_0 respectively, then

```
solve.QP(Dmat, dvec, Amat, bvec, meq=0)$solution
```

  gives the vector b that minimizes $(-d^T b + 0.5 b^T D b)$ with the constraints $A^T b \geq b_0$. Setting `meq=m` means the first m constraints are equalities.

- Example 3. Generate $Y_i = f(X_i) + \varepsilon_i$ with $f(x) = (x - 0.5)_+^2$ for $x \in (-\infty, \infty)$ as follows:

```
set.seed(1)
n <- 1000
x <- seq(0, 1, length=n)
f <- function(x){ ans <- (x-0.5)^2; ans[x<0.5] <- 0; return(ans) }
y <- f(x)+rnorm(n, sd=0.05)
```

  Approximate f using an increasing quadratic spline with 7 equally spaced knots in $(0, 1)$ and find $\hat{f}$: the least squared estimator of f by formulating the optimization problem as a quadratic programming problem. Find the ISE.

  Sol. Let B be the $n \times J$ matrix whose (i, j)-th element is $B_j(X_i)$

```
get.fit <- function(x,y, knotlist, deg=2, bknots =c(0,1)){
 B <- bs(x, knots=knotlist, deg=deg, Boundary.knots=bknots, intercept=TRUE)
 Dmat <- t(B)%*% B
 dvec <- as.numeric( y %*% B)
 xi <- c(bknots[1], knotlist, bknots[2])
 Amat <- t(dbs(xi, knotlist, bknots, m=deg+1, der=1))
 bvec <- rep(0, length(xi))
 a.hat <- solve.QP(Dmat, dvec, Amat, bvec)$solution
 #a.hat: vector of the estimated coefficients of B-spline basis functions
 fhat <- function(u){
  bu <- bs(u, knots=knotlist, deg=deg, Boundary.knots=bknots, intercept=TRUE)
  ans <- bu %*% a.hat
  return(ans[,1])
  }
 return(fhat)
}
```

```
#compute and plot fhat
fhat <- get.fit(x,y,(1:7)/8)
plot(x,y)
curve(fhat, 0, 1, add=TRUE, col=2)
curve(f, 0,1, add=TRUE, col=3)

#compute ISE
g <- function(x){ (fhat(x)-f(x))^2 }
integrate(g,0,1)$value
```

- Exercise 1. Generate $Y_i = f(X_i) + \varepsilon_i$ with $f(x) = e^{-x} I_{(-\infty,0.5)}(x) + e^{-0.5} I_{[0.5,\infty)}(x)$ for $x \in (-\infty, \infty)$ as follows:

```
set.seed(1)
n <- 1000
x <- seq(0, 1, length=n)
f <- function(x){ ans <- exp(-x); ans[x>=0.5] <- exp(-0.5); return(ans) }
y <- f(x)+rnorm(n, sd=0.05)
```

  Approximate f using an decreasing quadratic spline with 7 equally spaced knots in $(0, 1)$ and find $\hat{f}$: the least squared estimator of f by formulating the optimization problem as a quadratic programming problem. Find the ISE.

- Exercise 2. Generate $Y_i = f(X_i) + \varepsilon_i$ with $f(x) = e^{-x^2/2}/\sqrt{2\pi}$ for $x \in [-3, 3]$ as follows:

```
set.seed(1)
n <- 1000
x <- seq(-3, 3, length=n)
f <- function(x){ dnorm(x, 0, 1) }
y <- f(x)+rnorm(n, sd=0.05)
```

  Approximate f using a quadratic spline with 7 equally spaced knots in $(-3, 3)$ that is increasing on $[-3, 0]$ and decreasing on $[0, 3]$. Find $\hat{f}$: the least squared estimator of f by formulating the optimization problem as a quadratic programming problem. Find the ISE.