Regression using tensor basis functions

- Regression. Suppose that we observe (X_i, Y_i): $1 \le i \le n$, where

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

and ε_i's are IID errors with mean zero and variance σ^2. The problem of interest in regression is to estimate f based on (X_i, Y_i)'s.

- Approximation approach. Choose a set of functions $\{f_j\}_{j=1}^J$ so that

$$f(x) \approx \sum_{j=1}^J a_j f_j(x), \tag{1}$$

then

$$\hat{f} = \sum_{j=1}^J \hat{a}_j f_j, \tag{2}$$

where $\hat{a}_j$'s are the least square estimators for a_js. That is, $\hat{a}_j$'s are the a_js so that

$$\sum_{i=1}^n \left(Y_i - \sum_{j=1}^J a_j f_j(X_i) \right)^2 \tag{3}$$

is minimized.

- Suppose that f is a function on $I_1 \times \cdots \times I_d$, where $I_1, \ldots, I_d$ are d intervals in $(-\infty, \infty)$. Suppose that $\{\phi_{j,1}, \ldots, \phi_{j,m_j}\}$: $j = 1, \ldots, d$ are d sets of basis functions on intervals $I_1, \ldots, I_d$ respectively. Let

$$\Lambda = \{(i_1, \ldots, i_d) : i_j \in \{1, \ldots, m_j\} \text{ for } j = 1, \ldots, d\},$$

and for $(i_1, \ldots, i_d) \in \Lambda$, define $f_{i_1,\ldots,i_d}$ as

$$f_{i_1,\ldots,i_d}(x_1, \ldots, x_d) = \phi_{1,i_1}(x_1) \cdots \phi_{d,i_d}(x_d) \text{ for } (x_1, \ldots, x_d) \in I_1 \times \cdots \times I_d,$$

then the $\prod_{j=1}^d m_j$ functions $f_{i_1,\ldots,i_d}$: $(i_1, \ldots, i_d) \in \Lambda$ are the tensor product basis functions based on the d sets of basis functions $\{\phi_{j,1}, \ldots, \phi_{j,m_j}\}$: $j = 1, \ldots, d$, we can approximate f using a linear combination of $f_{i_1,\ldots,i_d}$: $(i_1, \ldots, i_d) \in \Lambda$. Since the approximation is of the form in (1), the least square estimator $\hat{f}$ in (2) can be obtained. Below we consider the case where the d sets of basis functions are same for simplicity.

- Example 1. Generate data according to the regression model $Y_i = f(X_i) + \varepsilon_i$ for $i = 1, \ldots, 1000$ as follows.

```
set.seed(1)
f <- function(x){ dnorm(x[,1]-0.5, sd=0.2)*dnorm(x[,2]-0.5, sd=0.2) }
n <- 1000
X <- matrix(runif(n*2), n,2)
y <- f(X)  + rnorm(n,sd=0.4)
```

Approximate f using tensor basis functions constructed based on the univariate basis functions $\{1, \cos(2\pi kx), \sin(2\pi kx)\colon\ k = 1,\ 2,\ 3\ \}$ and then use (2) to find $\hat{f}$. Find the ISE for $\hat{f}$.

```
#####define functions
### bx.trigo(x,m) compute univariate basis functions
###  1, sin(k*pi*x) and cos(k*pi*x)   for k=1,..., m.
###  x is a vector.
bx.trigo <- function(x, m){
  n <- length(x)
  b.trigo <- matrix(0, n, 2*m)
  for (k in 1:m){
    b.trigo[ ,k] <- cos(2*pi*k*x)
    b.trigo[ ,k+m] <- sin(2*pi*k*x)
  }
 return(cbind(rep(1,n), b.trigo))
}
### bx.tensor: compute tensor basis functions at x, x can be a matrix
bx.tensor <- function(x, m, bx.uni){
 if ( is.null(dim(x)) ) { return( bx.uni(x,m)  ) }
 n <- dim(x)[1]
 d <- dim(x)[2]
 mat.list <- vector("list", d)
 for (i in 1:d){   mat.list[[i]] <-   bx.uni(x[,i],m)  }
 n.basis1 <- dim(mat.list[[1]])[2]
 n.basisd <- n.basis1^d
 v <- vector("list", d)
 for (i in 1:d){   v[[i]] <- 1:n.basis1  }
 ind.mat <- as.matrix(expand.grid(v))
 bx <- matrix(0, n, n.basisd)
 for (j in 1:n.basisd) {
  ind <- ind.mat[j,]
  b.prod <- rep(1, n)
  for (k in 1:d) { b.prod <- b.prod*mat.list[[k]][,ind[k]] }
  bx[, j] <- b.prod
 }
 return(bx)
}
### get.fhat: compute the least square estimator for f
get.fhat <- function(data.x, data.y, m, bx.uni){
 bx.data <- bx.tensor(data.x, m, bx.uni)
 coef.hat <- lm(data.y~bx.data-1)$coef
 fhat <- function(x){
   bx.x <-  bx.tensor(x, m, bx.uni)
   return( as.numeric( bx.x %*% coef.hat )  )
  }
 return(fhat)
}
### generate data and compute fhat
set.seed(1)
```

```
f <- function(x){ dnorm(x[,1]-0.5, sd=0.2)*dnorm(x[,2]-0.5, sd=0.2) }
n <- 1000
X <- matrix(runif(n*2), n,2)
y <- f(X)  + rnorm(n,sd=0.4)
fhat <- get.fhat(X, y, 3, bx.trigo)

### compute ISE using monte carlo integration based on 10000 monte carlo samples
set.seed(2)
n.mc <- 10000
u <- matrix( runif(n.mc*2), n.mc, 2)
mean( (fhat(u) - f(u))^2 )
#approximate ISE: 0.008655918


#####plot f and fhat
x <- seq(0, 1, length= 31)
y <- seq(0, 1, length= 21)
xy <- as.matrix(expand.grid(x,y))
z <- matrix(f(xy), nrow=length(x))
res <- persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
require(grDevices)
ind <- seq(1,31*21,by=4)
points(trans3d(xy[ind,1], xy[ind,2],fhat(xy[ind,]), pmat = res), col = 4, pch = 16)
```

- Exercise 1. Generate data according to the regression model $Y_i = f(X_i) + \varepsilon_i$ for $i = 1, \ldots, 1000$ as in Example 1:

  ```
  set.seed(1)
  f <- function(x){ dnorm(x[,1]-0.5, sd=0.2)*dnorm(x[,2]-0.5, sd=0.2) }
  n <- 1000
  X <- matrix(runif(n*2), n,2)
  y <- f(X)  + rnorm(n,sd=0.4)
  ```

  Approximate f using tensor basis functions constructed based on the univariate B-spline basis functions on $[0, 1]$ with degree 3 and k equally space knots $1/(k+1), \ldots, k/(k+1)$, where k is chosen so that the number of univariate B-spline basis functions is 7. Use (2) to find $\hat{f}$. Find the ISE for $\hat{f}$ using monte carlo integration based on 10000 monte carlo samples.

- Additive regression model. For a regression function f on I^d, where $I \subset (-\infty, \infty)$, if there exist univariate functions $f_1, \ldots, f_d$ such that

  $$f(x_1, \ldots, x_d) = f_1(x_1) + f_2(x_2) + \cdots f_d(x_d) \text{ for } (x_1, \ldots, x_d) \in I^d,$$

  then we say that f is additive. In such case, each f_j can be approximated using univariate basis functions, so the total number of basis functions is much less than that based on tensor basis functions.

- Example 2. Generate data according to the regression model $Y_i = f(X_i) + \varepsilon_i$ for $i = 1, \ldots, 1000$ as follows.

```
set.seed(1)
f <- function(x){ dnorm(x[,1]-0.5, sd=0.2)+dnorm(x[,2]-0.5, sd=0.2) }
n <- 1000
X <- matrix(runif(n*2), n,2)
y <- f(X)  + rnorm(n,sd=0.4)
```

Note that $f(x_1, x_2) = f_1(x_1) + f_2(x_2)$. Approximate f so that f_1 and f_2 are approximated using the univariate basis functions $\{1, \cos(2\pi kx), \sin(2\pi kx)$: $k = 1, 2, 3$ $\}$ and then use (2) to find $\hat{f}$. Find the ISE for $\hat{f}$.

```
####define function for generating univariate basis functions
 bx.trigo <- function(x, m){
  n <- length(x)
  b.trigo <- matrix(0, n, 2*m)
  for (k in 1:m){
    b.trigo[ ,k] <- cos(2*pi*k*x)
    b.trigo[ ,k+m] <- sin(2*pi*k*x)
  }
 return(cbind(rep(1,n), b.trigo))
}
#####generate data
set.seed(1)
f <- function(x){ dnorm(x[,1]-0.5, sd=0.2)+dnorm(x[,2]-0.5, sd=0.2) }
n <- 1000
X <- matrix(runif(n*2), n,2)
y <- f(X)  + rnorm(n,sd=0.4)

#### compute fhat
bx <- cbind(bx.trigo(X[,1], 3), bx.trigo(X[,2], 3))[,-1]
y.lm <- lm(y~bx-1)
fhat <- function(u){
  bx.u <- cbind(bx.trigo(u[,1], 3), bx.trigo(u[,2], 3))[,-1]
  return( as.numeric(bx.u %*% y.lm$coef) )
}

#####compute ISE
set.seed(2)
n.mc <- 10000
u <- matrix( runif(n.mc*2), n.mc, 2)
mean( (fhat(u) - f(u))^2 )
#approximate ISE: 0.002077141

#### compare with the fhat obtained using tensor basis functions
fhat <- get.fhat(X, y, 3, bx.trigo)
set.seed(2)
n.mc <- 10000
u <- matrix( runif(n.mc*2), n.mc, 2)
mean( (fhat(u) - f(u))^2 )
#approximate ISE:  0.008608629
```

- Exercise 2. Write a R function that computes $\hat{f}$ based on data X_i, Y_i: $i = 1, \ldots, n$, where f is assumed to be additive ($f(x_1, \ldots, x_d) = f_1(x_1) + \cdots + f_d(x_d)$) and each f_j is approximated using univariate B-spline basis functions on $[0, 1]$ with degree 3 and k equally space knots $1/(k+1)$, …, $k/(k+1)$. The input variables are

  - `data.y`: the vector $(Y_1, \ldots, Y_n)$.
  - `data.x`: the $n \times d$ matrix whose i-th row is X_i for $i = 1, \ldots, n$
  - `k`: the number of inner knots in $(0, 1)$ for univariate B-spline basis functions.

  and the output is $\hat{f}$.

- Exercise 3. Generate data as follows.

```
set.seed(1)
f <- function(x){
  dnorm(x[,1]-0.5, sd=0.2) + dnorm(x[,2]-0.5, sd=0.2) + dnorm(x[,3]-0.5, sd=0.2)
}
n <- 1000
X <- matrix(runif(n*3), n,3)
y <- f(X) + rnorm(n,sd=0.4)
```

  Use your R function in Exercise 2 to compute $\hat{f}$ with $k = 3$ and its ISE. Also compute the ISE for $\hat{f}$ computed based on tensor B-spline basis functions, where the univariate basis functions are B-spline basis functions on $[0, 1]$ with degree 3 and 3 equally space knots $1/4$, …, $3/4$. Which ISE is smalller?