非参估计作业

汪利军

June 9, 2017

Contents

1 第一题

$\{Y_t\} \sim U[-3,3]$，进行核密度估计

- 选择不同的核函数进行估计，$K_1(x)$ 为高斯核，$K_2(x) = 0.5I(|x| \leq 1)$
- 通过 MSE 和 MISE 选择窗宽
- 画图 $\sqrt{nh}(\hat{f}(x) - f(x))$

1.1 估计

首先生成 n 个 $U(-3,3)$ 的随机数据。

```
n = 1000
set.seed(12345)
y = runif(n, -3, 3)
```

自己编写核函数估计的函数

```
mykde(y, kernel, h, reflect, y.range)
```

其中参数 “y” 是需要进行核密度估计的数据，“kernel” 可以选择高斯核函数（“gaussian”）或者均匀核函数（“uniform”）,“h” 为窗宽，“reflect” 表示是否通过反射处理边缘数据，如果需要进行反射处理，则需要输入参数 “y.range” 来确定对称轴。具体实现细节如下:

```
mykde <- function(y, kernel = "gaussian", h = 0.1, reflect = FALSE, y.range=NULL )
{
  n = 1000
  x = seq(-3, 3, length.out = 1000)
  x.left = x[1:250]
  x.middle = x[251:750]
  x.right = x[751:1000]
  if (kernel == "gaussian")
  {
    kernel.gaussian <- function(x)
    {
      return(exp(-x^2/2)/sqrt(2*pi))
    }
    if (reflect)
    {
```

```
    fhat.left = sapply(x.left,
      function(x) sum(kernel.gaussian((y-x)/h) +
                         kernel.gaussian((2*y.range[1]-y-x)/h))/(n*h))
    fhat.middle = sapply(x.middle,
      function(x) sum(kernel.gaussian((y-x)/h))/(n*h))
    fhat.right = sapply(x.right,
      function(x) sum(kernel.gaussian((y-x)/h) +
                         kernel.gaussian((2*y.range[2]-y-x)/h))/(n*h))
    fhat = c(fhat.left, fhat.middle, fhat.right)
  }
  else
    fhat = sapply(x, function(x) sum(kernel.gaussian((y-x)/h))/(n*h))
}
else if (kernel == "uniform")
{
  kernel.uniform <- function(x)
  {
    x[abs(x) < 1] = 1
    x[abs(x) > 1] = 0
    return(0.5*x)
  }
  if (reflect)
  {
    fhat.left = sapply(x.left,
      function(x) sum(kernel.uniform((y-x)/h) +
                         kernel.uniform((2*y.range[1]-y-x)/h))/(n*h))
    fhat.middle = sapply(x.middle,
```

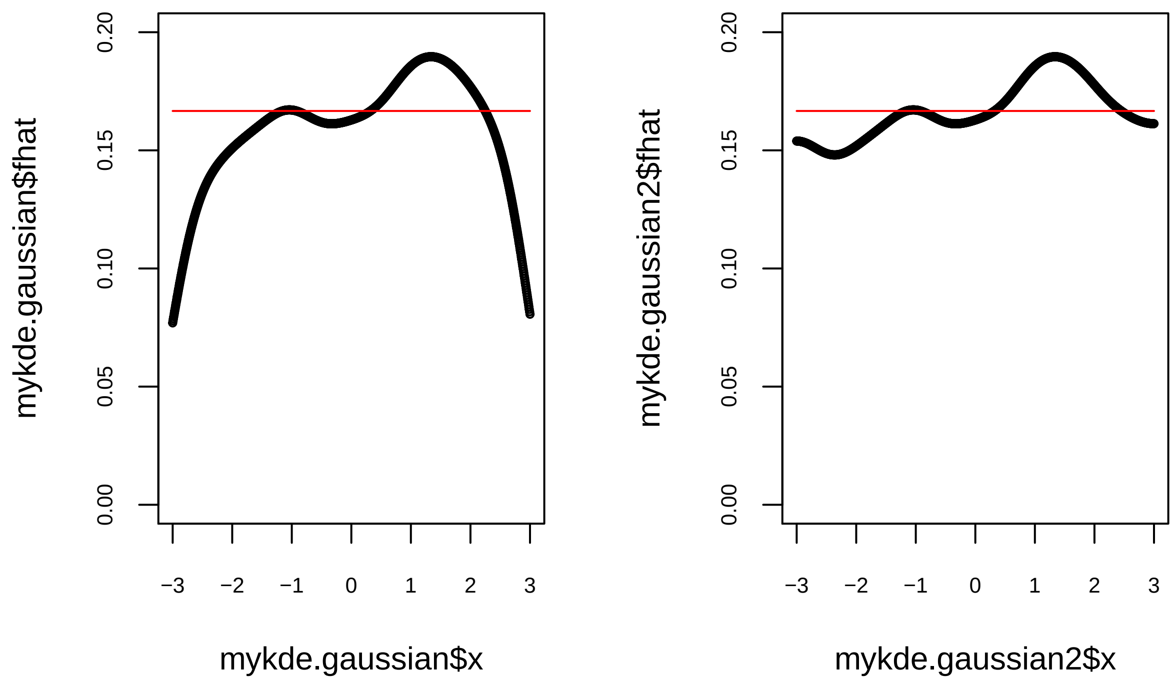
```
        function(x) sum(kernel.uniform((y-x)/h))/(n*h))
      fhat.right = sapply(x.right,
        function(x) sum(kernel.uniform((y-x)/h) +
                          kernel.uniform((2*y.range[2]-y-x)/h))/(n*h))
      fhat = c(fhat.left, fhat.middle, fhat.right)
    }
    else
      fhat = sapply(x, function(x) sum(kernel.uniform((y-x)/h))/(n*h))
  }
  else
    cast(paste0("Can not support ", kernel,
                "kernel, please change to gaussian kernel or uniform kernel.."))
  res = list(x = x,
             fhat = fhat)
  return(res)
}
```

取窗宽 $h = 0.395$，在考虑边缘效应和不考虑边缘效应时进行估计。对于反射的具体处理如下，若 $y \in [a, b]$。则对较小的 y（从小到大排序，位于前 25%）取关于 $y = a$ 的对称点，对较大的 y（从小到大排序，位于后 25%）取关于 $y = b$ 的对称点，即

$$
\begin{aligned}
f^*(x) &= \frac{1}{nh}\sum_{t=1}^{n}[K(\tfrac{y_t-x}{h}) + K(\tfrac{2a-y_t-x}{h})], \qquad & x \in [a, a+\tfrac{1}{4}(b-a)] \\
f^*(x) &= \frac{1}{nh}\sum_{t=1}^{n}[K(\tfrac{y_t-x}{h}) + K(\tfrac{2b-y_t-x}{h})], \qquad & x \in [a+\tfrac{3}{4}(b-a), b]
\end{aligned}
$$

```
## 高斯核
mykde.gaussian = mykde(y, kernel = "gaussian", h = 0.395)
mykde.gaussian2 = mykde(y, kernel = "gaussian", h = 0.395,
                        reflect = TRUE, y.range = c(-3,3))
```

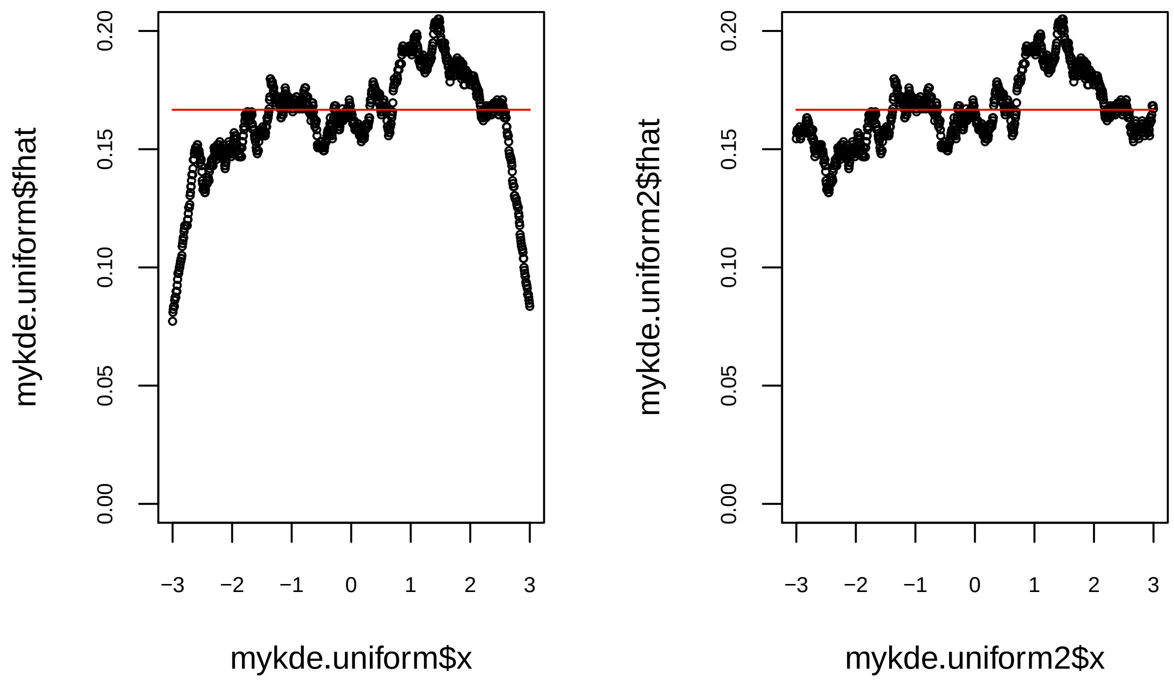
```r
par(mfrow = c(1, 2))
plot(mykde.gaussian$x, mykde.gaussian$fhat, ylim = c(0, 1/5),
     cex.axis = .7, cex = 0.5)
segments(-3,1/6,3,1/6,col = "red")
plot(mykde.gaussian2$x, mykde.gaussian2$fhat, ylim = c(0, 1/5),
     cex.axis = .7, cex = 0.5)
segments(-3,1/6,3,1/6,col = "red")
```



左图是不考虑边缘效应的核密度估计，而右图是通过反射法来考虑边缘效应进行核密度估计。从高斯核密度估计的这两种图象可以看出，反射处理能够有效减轻边缘效应带来的核密度估计的影响。

```r
## 均匀核
mykde.uniform = mykde(y, kernel = "uniform", h = 0.395)
mykde.uniform2 = mykde(y, kernel = "uniform", h = 0.395,
                       reflect = TRUE, y.range = c(-3,3))
par(mfrow = c(1, 2))
```

```r
plot(mykde.uniform$x, mykde.uniform$fhat, ylim = c(0, 1/5),
     cex.axis = .7, cex = 0.5)
segments(-3,1/6,3,1/6,col = "red")
plot(mykde.uniform2$x, mykde.uniform2$fhat, ylim = c(0, 1/5),
     cex.axis = .7, cex = 0.5)
segments(-3,1/6,3,1/6,col = "red")
```



从这两张图象可以看出，反射处理能够有效减轻边缘效应带来的核密度估计的影响。从均匀核密度估计的这两张图象可以看出，反射处理能够有效减轻边缘效应带来的核密度估计的影响。

1.2 确定窗宽

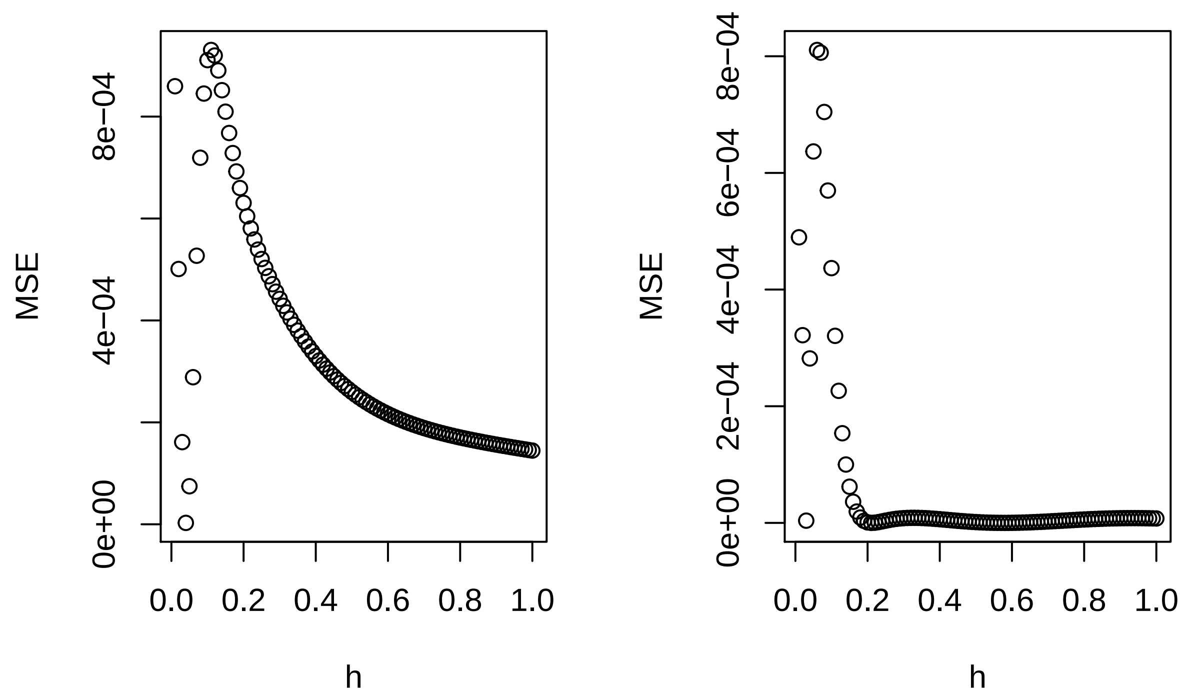
下面选择高斯核函数，并采用反射方法处理边缘效应。

1.2.1 MSE

```
h = seq(0.01, 1, length.out = 100)
fhat = sapply(h, function(x) {res = mykde(y, h = x,
                                   reflect = TRUE, y.range = c(-3,3)); res$fhat})
fhat.mse = apply(fhat, 2, function(x) (x-1/6)^2)
fhat.mse.min = apply(fhat.mse, 1, function(x) which.min(x))
```

以 y_{123}, y_{523} 为例，做出 $E(\hat{f}(x) - f(x))^2$ 关于 h 的图象

```
par(mfrow = c(1,2))
plot(h, fhat.mse[123, ], ylab = "MSE", xlab = "h")
plot(h, fhat.mse[523, ], ylab = "MSE", xlab = "h")
```



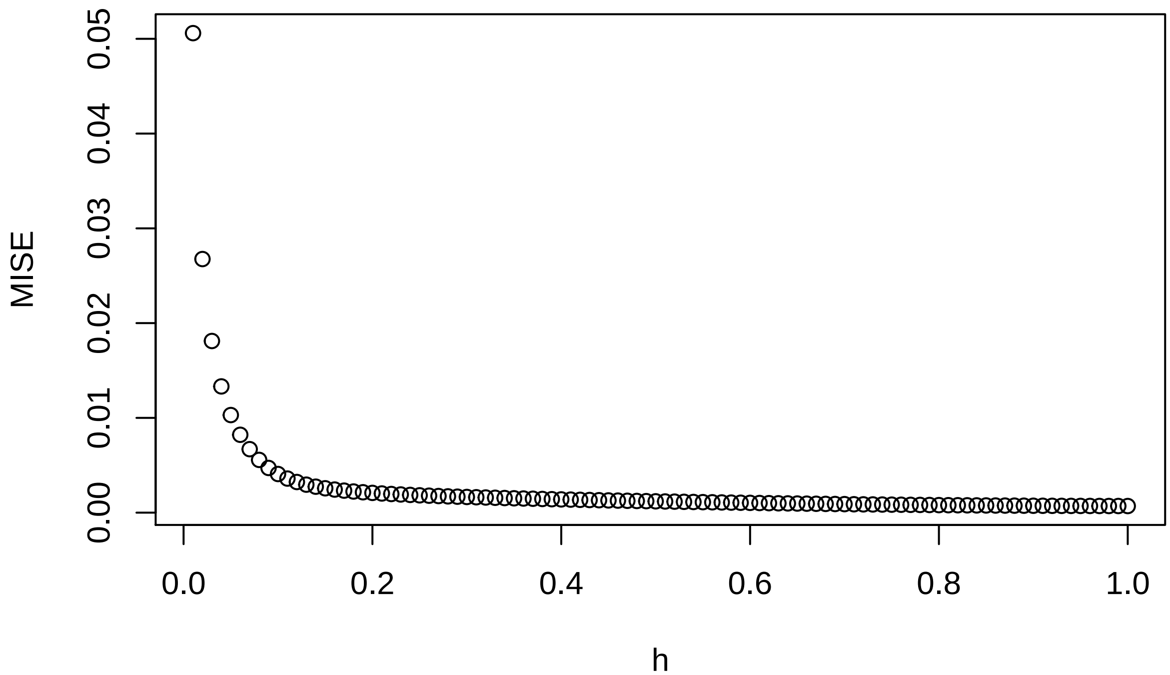
从图象可以看出 MSE 的变化趋势，当然对于不同的点其变化趋势不同，对于每个点我们选择使得 MSE 最小的那个 h, 这样选出来的 h 如下（只显示了前 20 个 x 对应的 h）

```r
h[fhat.mse.min[1:20]]
```

```
##  [1] 0.19 0.19 0.07 0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.18 0.18 0.18 0.18
## [15] 0.17 0.17 0.16 0.16 0.15 0.15
```

1.2.2 MISE

```r
fhat.mise = colSums(fhat.mse)/ncol(fhat.mse)
plot(h, fhat.mise, ylab = "MISE", xlab = "h")
```



从图象看起来，MISE 是随着 h 的增大而降低的（在给定的 h 的范围内），这与上课老师讲的最优 h 在 $O(h^{-1/5})$ 处取得似乎存在矛盾。但注意到当 $f(x)=1/6$ 时

$$Bias = \frac{h^2}{2} f''(x) \int_{-A}^{A} z^2 K(z)dz + o(h^2) = o(h^2)$$

则 MSE 主要依赖方差项，

$$Var(\hat{f}(x)) \approx \frac{1}{nh} \int_{-A}^{A} K^2(z)dz \cdot f(x)$$

对于方差项，h 越大，则方差越小，从而 MSE（MISE）随着 h 的增大而降低，如果从这个角度看，图象中表现出的 MISE 随着 h 的增大而降低是合理的。

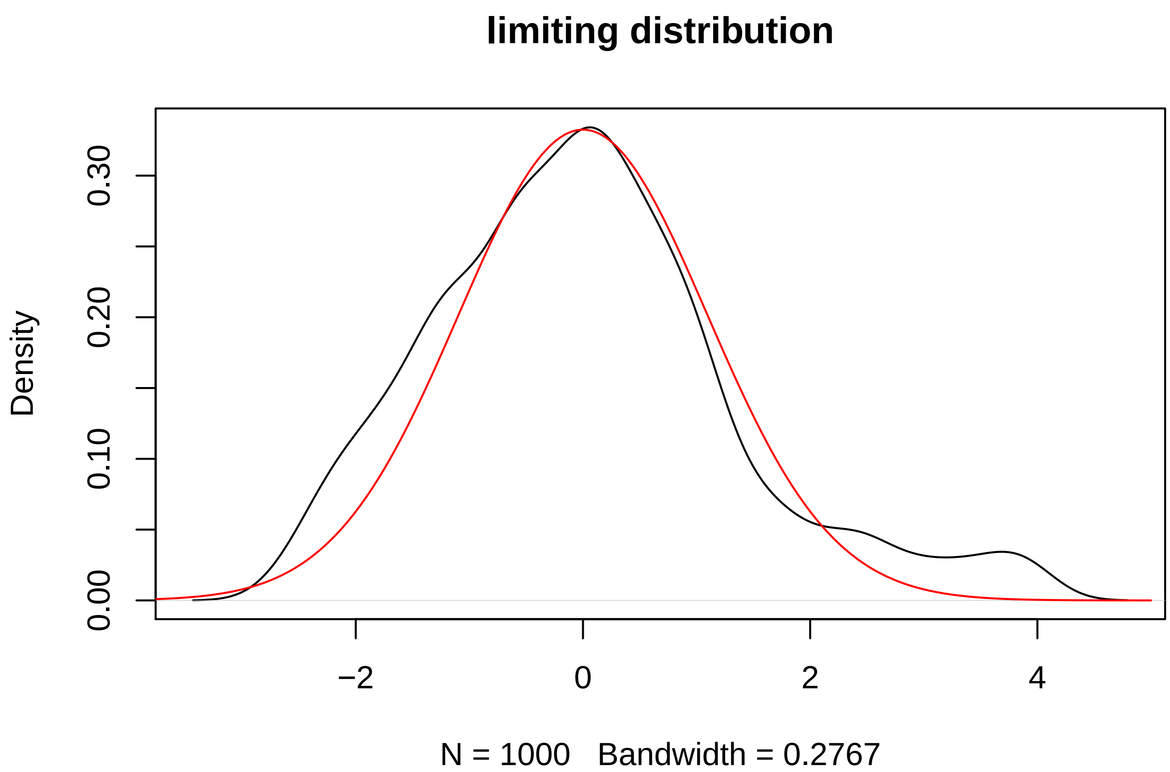
1.3 渐近分布

取 $h = 0.1$，采用高斯核函数进行核密度估计，考虑

$$\sqrt{nh}(\hat{f}(x) - f(x))$$

的渐进分布

```
n = 10000
set.seed(12345)
y = runif(n, -3, 3)
hh = 0.1
res = mykde(y, kernel = "gaussian", h = hh, reflect = TRUE, y.range = c(-3,3))
fhat = res$fhat
plot(density(sqrt(n*hh)*(fhat-mean(fhat))), main = "limiting distribution")
# normal
x = seq(-5, 5, length.out = 1000)
sigmahat = 1.2
p = 1/sqrt(2*pi*sigmahat^2)*exp(-x^2/(sigmahat*2))
lines(x, p, col = "red")
```



图中黑线是 $\sqrt{nh}(\hat{f}(x) - f(x))$ 在 $n = 10000, h = 0.1$ 情形下的密度曲线，红色曲线是 $\sigma = 1.2, \mu = 0$ 的正态分布密度曲线，可以看出两者近似程度还是相当高的，与极限分布为正态的结论是一致的。

2 第二题

$$Y_t = m(X_t) + \varepsilon_t$$

其中

$$m(x) = 2sin\pi x \qquad x \in [0, 2]$$

ε_t 任取，模拟时取 $X_t \sim U[0, 2]$

对 $\hat{m}(x)$ 进行局部线性估计，计算 $\hat{m}(x)$ 的 MSE 和 MISE.

2.1 理论推导

由泰勒展开有

$$m(x) = m(x_0) + (x - x_0)m'(x_0) + o(|x - x_0|), \qquad x \to x_0$$

则有

$$Y_t = m(x_t) = \alpha + \beta(x_t - x_0) + \varepsilon_t$$

则局部最小二乘估计为

$$(\hat{\alpha}(x_0), \hat{\beta}(x_0)) = \underset{\alpha(x_0),\beta(x_0)}{arg\ min} \sum_{t=1}^{n}(y_t - \alpha - \beta(x_t - x_0))^2 K(\frac{x_t - x_0}{h})$$

记

$$X = \begin{pmatrix} 1 & x_1 - x_0 \\ 1 & x_2 - x_0 \\ \cdot & \cdots \\ 1 & x_n - x_0 \end{pmatrix} \quad W = \begin{pmatrix} K(\frac{x_1-x_0}{h}) & & O \\ & \ddots & \\ O & & K(\frac{x_n-x_0}{h}) \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

并记 $\boldsymbol{\beta} = (\alpha, \beta)'$

则局部最小二乘估计写成矩阵形式为

$$\boldsymbol{\beta}(\hat{x_0}) = \underset{\boldsymbol{\beta}(x_0)}{arg\ min}\ (Y - X\boldsymbol{\beta})^T W (Y - X\boldsymbol{\beta})$$

即为加权最小二乘估计，

$$\hat{\boldsymbol{\beta}}(x_0) = (X'WX)^{-1}(XWY)$$

且

$$\hat{m}(x_0) = \alpha(x_0)$$

在 "lm()" 函数中指定权重 "weights=W" 便可以求解。

2.2 编程求解

选择高斯噪声，产生 n 个数据

```
n = 1000
set.seed(1234)
xt = runif(n, 0, 2)
set.seed(4321)
et = rnorm(n)
yt = 2*sin(pi*xt) + et
#yt = xt + et
```

编写 myloess 函数得到局部最小二乘的估计结果

```
myloess(xt, yt, h, ngrid)
```

其中，"xt" 和 "yt" 分别是数据点的横纵坐标，"h" 为窗宽，"n.grid" 是网格剖分的格数，具体函数实现细节如下

```
myloess <- function(xt, yt, h, ngrid)
{
  x.grid = seq(0, 2, length.out = ngrid)
  kernel.gaussian <- function(x)
  {
      return(exp(-x^2/2)/sqrt(2*pi))
  }
```

```
kernel.uniform <- function(x)
{
    x[abs(x) < 1] = 1
    x[abs(x) > 1] = 0
    return(0.5*x)
}
weight <- function(x)
{
  return(sapply(xt, function(xx) kernel.gaussian((xx-x)/h)))
}
w.grid = sapply(x.grid, weight)
alpha.grid = numeric(ngrid)
beta.grid = numeric(ngrid)
y.grid = numeric(ngrid)
for(i in 1:ngrid)
{
  lm.x = xt - x.grid[i]
  lm.fit = lm(yt ~ lm.x, w = w.grid[,i])
  lm.coef = coef(lm.fit)
  alpha.grid[i] = lm.coef[[1]]
  beta.grid[i] = lm.coef[[2]]
  y.grid[i] = alpha.grid[i]
  #y.grid[i] = alpha.grid[i] + beta.grid[i]*(x.grid[i]-x.grid[i])
  #y.grid[i] = predict(lm.fit, data.frame(lm.x = x.grid[i]))
}
res = list(alpha = alpha.grid,
           beta = beta.grid,
```

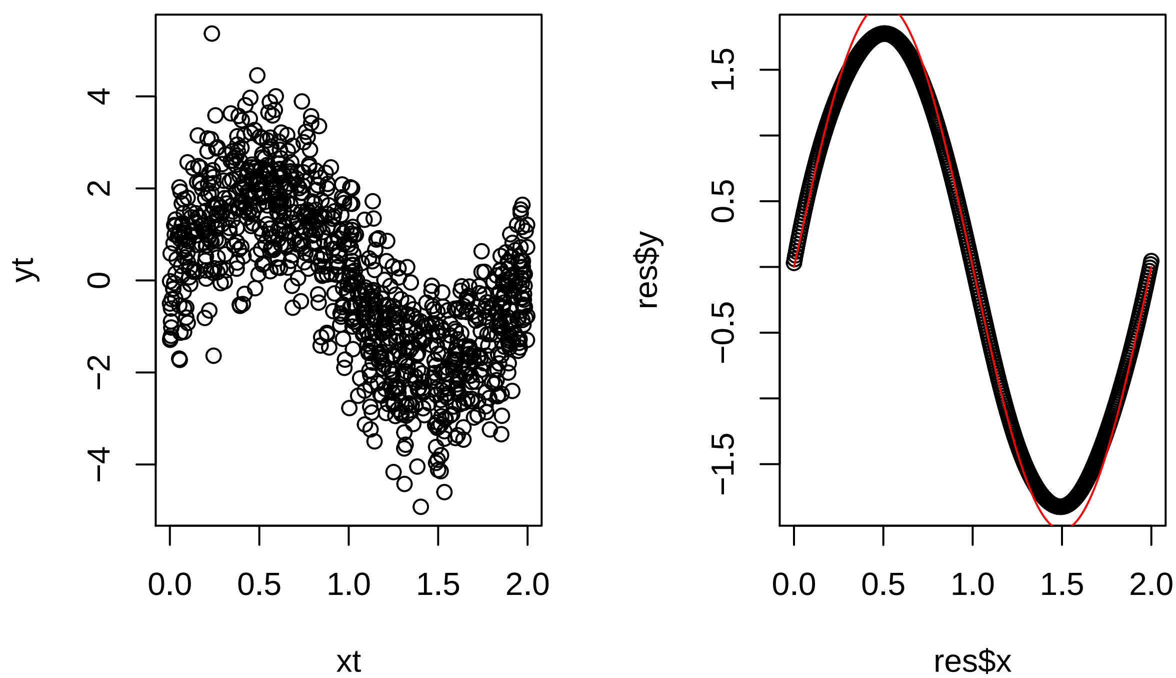
```r
              x = x.grid,
              y = y.grid)
  return(res)
}
```

选取窗宽 $h = 0.15, ngrid = 512$ 进行模拟，得到下面结果

```r
res = myloess(xt, yt, h = 0.15, ngrid = 512)
```

作出核密度估计拟合后的结果，与原始结果进行比较，可以看出局部线性估计可以很好地得到 y 与 x 原始的关系。

```r
par(mfrow=c(1,2))
plot(xt, yt)
plot(res$x, res$y)
lines(res$x, 2*sin(pi*res$x), col = "red")
```



右图中红色线条为 $y = 2sin\pi x$ 的曲线，可以看出局部线性估计在极值处的估计偏小，即 $x = 0.5, 1.5$ 处，其余地方与真实的差距不是很大。

2.3 MSE 和 MISE

$$MSE(\hat{m}(x)) = E(\hat{m}(x) - m(x))^2$$

$$MISE(\hat{m}(x)) = E\int_R (\hat{m}(x) - m(x))^2 dx = \frac{1}{N}\sum_{i=1}^{N}(\hat{m}(x_i) - m(x_i))$$

编写下列代码计算 MSE 和 MISE

```r
# MSE
mx.mse = (res$y - 2*sin(pi*res$x))^2
summary(mx.mse)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 1.000e-08 1.984e-03 1.186e-02 1.725e-02 3.173e-02 5.144e-02
```

```r
# MISE
mx.mise = mean(mx.mse)
mx.mise
```

```
## [1] 0.0172489
```

计算出所有点的 MSE，其四分位数分布结果如上所示，其均值即为 MISE，即 MISE=0.0172489。