FUNGSI-FUNGSI KERNEL PADA METODE REGRESI NONPARAMETRIK DAN APLIKASINYA PADA *PRIEST RIVER EXPERIMENTAL FOREST'S* DATA

Siana Halim, Indriati Bisono
Fakultas Teknologi Industri, Jurusan Teknik Industri, Universitas Kristen Petra, Surabaya
E-mail: {halim,mlindri}@petra.ac.id

ABSTRAK

Dalam paper ini akan diberikan beberapa model untuk mengestimasi fungsi regresi bila sebuah data Y diberikan. Pembuatan fungsi – fungsi estimasti tersebut juga diberikan dengan menggunakan freeware statistik R dan juga diberikan beberapa trik dalam pemrograman. Model-model ini diaplikasikan pada *Priest River Experimental Forest's* data.

Kata kunci: regresi nonparametrik, *smoothing*, kernel, R.

ABSTRACT

In this paper we discuss some models for estimating the regression function, provided the data Y is given. The programming of these function are presented as well as the "tricks" in the R- programming, a statistical freeware. We applied these models to The Priest River Experimental Forest's data.

***Keywords**: nonparametric regression, smoothing, kernel, R.*

1. PENDAHULUAN

Tujuan dasar dalam sebuah analisa regresi adalah untuk mempelajari bagaimana respon sebuah peubah *Y* terhadap perubahan yang terjadi pada peubah lain yaitu *X*. Hubungan antara *X* dan *Y* dapat dituliskan sebagai berikut:

$$Y = r(X) + \varepsilon \quad (1)$$

dimana r adalah fungsi matematik, yang disebut sebagai fungsi regresi dan ε adalah error yang mengijinkan terjadinya deviasi dari hubungan yang murni deterministik. Pada aplikasi, kita mengumpulakan data $(X_1,Y_1),\ldots,(X_n,Y_n)$ yang berisi informasi tentang fungsi *r*. Dari data-data ini kita dapat menduga ataupun mengestimasi fungsi *r* tersebut.

Jika pengetahuan kita tentang fungsi *r* ini minim, maka estimasi terhadap fungsi *r* ini dapat didekati secara nonparametrik. Agar pendekatan nonparametrik ini menghasilkan estimasi terhadap *r* yang masuk akal, maka hal yang harus diperhatikan adalah asumsi bahwa *r* memiliki derajat kemulusan. Biasanya kontinuitas dari *r* merupakan syarat yang cukup untuk menjamin sebuah estimator akan konvergen pada *r* yang sesungguhnya bila jumlah data bertambah tanpa batas.

Sebagai bandingan dari metode nonparametrik tentunya adalah metode parametrik, yang mendominasi statistika klasik. Andaikan peubah *X* diketahui berada pada selang [0,1]. Contoh sederhana dari model parametrik untuk *r* pada Persamaan (1), adalah persamaan garis lurus,

$$r(x) = \theta_0 + \theta_1 x\,,\ 0 \le x \le 1$$

dimana θ_0 dan θ_1 adalah konstanta yang tidak diketahui. Lebih umum lagi r dapat dinyatakan sebagai kombinasi linear sebagai berikut,

$$r(x) = \sum_{i=0}^{p} \theta_i r_i(x) \quad 0 \le x \le 1$$

di mana $r_0,...,r_p$ adalah fungsi-fungsi yang diketahui dan $\theta_0,...,\theta_p$ adalah konstanta yang tidak diketahui.

Jika asumsi kita terhadap sebuah model parametrik dibenarkan, fungsi regresi dapat diestimasi dengan cara yang lebih efisien daripada dengan menggunakan sebuah metode nonparametric. Namun demikian jika asumsi terhadap model parametric ini salah, maka hasilnya akan memberikan kesimpulan yang salah terhadap fungsi regresi.

Dalam paper ini akan dipresentasikan, beberapa model dari regresi nonparametrik, beserta aplikasinya pada data *Priest River Experimental Forest's* dengan menggunakan software R.

2. IDE DASAR DARI *SMOOTHING*

Tujuan dari *smoothing* adalah untuk membuang variabilitas dari data yang tidak memiliki effek sehingga ciri-ciri dari data akan tampak lebih jelas. *Smoothing* telah menjadi sinonim dengan metode-metode nonparametrik yang digunakan untuk mengestimasi fungsi-fungsi. Beberapa metode smothing dalam bermacam-macam konteks dapat dijumpai di Priestley (1981), Silverman(1986), Eubank (1988), Haerdle (1990), Hart(1997).

Jika diberikan data observasi $Y_1,...,Y_n$ pada titik-titik desain yang tetap (*fixed design points*) $x_1,...,x_n$, secara berturutan (agar lebih mudah, kita andaikan saja $0<x_1< .... < x_n < 1$). Asumsikan data memiliki model sebagai berikut:

$$Y_j = r(x_j) + \varepsilon_j, \; j = 1,...,n \qquad (2)$$

dimana *r* adalah sebuah fungsi yang didefinisikan pada selang [0,1] dan $\varepsilon_1,...,\varepsilon_n$ adalah peubah acak yang merepresentasikan error. Biasanya kita mengasumsikan $E(\varepsilon_i) = 0$ dan $Var(\varepsilon_i) = \sigma^2$, $i = 1,..,n$. Tujuan dari data analist adalah menduga fungsi regresi r pada tiap *x* di [0,1]. Untuk itu dalam subbab ini akan diberikan beberapa metode untuk mengestimasi fungsi regresi ini dan cara menuliskannya dalam program R[1].

2.1 Rata-rata lokal (*Local Averaging*)

Rata-rata local merupakan cara yang paling sederhana. Misalkan kita ingin mengestimasi fungsi *r(x)* untuk beberapa $x \in [0,1]$. Jika *r* adalah fungsi yang kontinu, maka nilai-nilai fungsi pada x_i yang dekat dengan *x* seharusnya akan cukup dekat dengan *r(x)*. Hal ini memberikan usulan bahwa merata-rata nilai Y_i yang bersesuaian dengan x_i yang dekat dengan *x* akan menghasilkan estimator tak bias untuk fungsi *r(x)*.

[1] Program hanya akan diberikan pada beberapa metode saja, untuk metode yang analog akan diberikan satu program saja.

Contoh 1

Misalkan kita memiliki 50 data titik, yang disimulasikan sebagai berikut

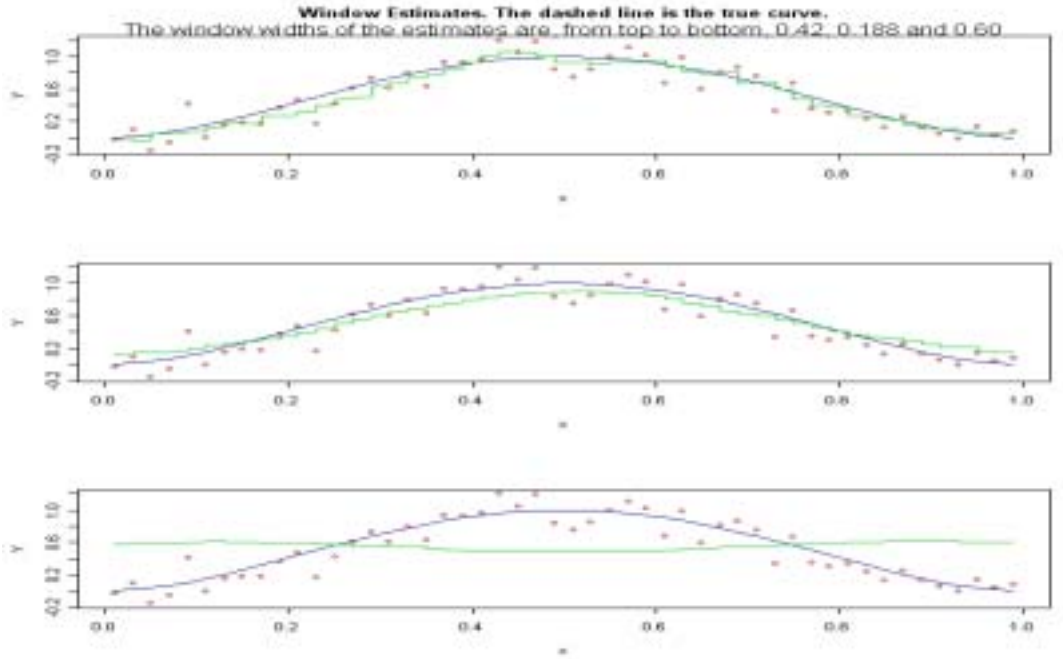
$$Y_j = r(x_j) + \varepsilon_j, \; j = 1,\ldots,50$$

di mana

$r(x_j) = (1-(2x-1)^2)^2$, $0 \le x \le 1$

$x_j=(j-0.5)/50, j = 1,\ldots,50$, dan $\varepsilon_j \sim N(0,(0.125)^2)$.

Untuk tiap x, perhatikan selang $[x-h,x+h]$, dimana h adalah bilangan positif yang cukup kecil nilainya. Rata-ratalah pasangan (x_j, Y_j) yang berada pada selang tersebut, dan hitung rata-rata untuk seluruh Y_j yang lain. Nilai rata-rata ini merupakan estimasi dari $r(x)$. Program dengan menggunakan R dapat dilihat pada Lampiran.



Gambar 1. Estimasi dengan rata-rata lokal, dengan nilai *h* dari atas ke bawah adalah *h = 0.42, h = 0.188* dan *h = 0.60*. Garis lurus adalah *r(x)* yang sebenarnya, titik-titik adalah nilai Y_j dan garis terputus-putus adalah estimasi dengan menggunakan rata-rata lokal.

Dari Gambar 1, dapat kita lihat bila nilai h terlalu kecil maka estimasi yang didapat akan sama dengan Y_j, tetapi bila h terlalu besar maka estimasi yang kita peroleh terlalu mulus. Tentu saja dalam kenyataan, metode ini tidak pernah digunakan. Diberikan di sini hanya untuk memberikan gambaran sederhana tentang regresi nonparametrik

2.2 *Kernel Smoothing*

Pada *kernel smoothing* rerata sederhana di atas digantikan dengan jumlahan berbobot, biasanya bobot yang lebih besar diberikan pada Y_j yang nilai x_j nya mendekati titik estimasi x. Di sini akan diberikan tiga macam kernel smoothing, yaitu

2.2.1 *Nadaraya Watson estimate*

$$\hat{r}_h^{NW}(x) = \frac{\sum_{j=1}^{n} Y_j K((x - x_j)/h)}{\sum_{j=1}^{n} K((x - x_j)/h)}, \quad 0 \le x \le 1 \tag{3}$$

di mana K adalah fungsi yang disebut sebagai kernel, sedangkan h adalah kuantitas yang disebut sebagai *bandwidth* atau *smoothing parameter*, dan yang mengkontrol kemulusan (*smoothness*) dari $\hat{r}_h^{NW}$. Beberapa tipe dari fungsi kernel $K \equiv K_R$ adalah:

$$K_R(u) = \frac{1}{2} I_{(-1,1)}(u)$$

dimana I_A merupakan fungsi indikator untuk himpunan A, yaitu,

$$I_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

Kernel K_R disebut sebagai *rectangular kernel*. Pilihan K yang umum adalah kernel Gaussian, yaitu

$$K_G(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

Tipe kernel lain yang juga sering digunakan adalah kernel Epanechnikov,

$$K_E(u) = 0.75(1 - u^2), \quad |u| < 1$$

2.2.2 Priestley – Chao estimate

$$\hat{r}_h^{PC}(x) = \frac{1}{h} \sum_{i=1}^{n} (x_i - x_{i-1}) Y_i K\left(\frac{x - x_i}{h}\right) \tag{4}$$

2.2.3 Gasser – Muller estimate

$$\hat{r}_h^{GM}(x) = \frac{1}{h} \sum_{i=1}^{n} Y_i \int_{s_{i-1}}^{s_i} K\left(\frac{x - u}{h}\right) du \tag{5}$$

di mana $s_0 = 0$, $s_i = (x_i + x_{i+1})/2$, $i = 1,\ldots,n-1$ dan $s_n = 1$.

2.3 Bandwidth Optimum

Telah diketahui secara umum, bahwa permasalahan utama pada *kernel smoothing* bukan terletak pada pemilihan kernel tetapi pada pemilihan *bandwidth* (lihat Tabel 1. Silverman, B.W (1986). Pemilihan *bandwidth* optimum lebih ditekankan pada penyeimbangan antara bias dan variance, seperti terlihat pada Gambar 1. Satu perumusan masalah yang dapat memperlihatkan hubungan antara bias dan variance adalah *mean square error* – MSE (lihat pada Lemma 1), karena itu dengan meminimumkan MSE maka permasalah antara bias dan variance di atas dapat diminimumkan juga.

Lemma 1: MSE *f(x)* dapat diuraikan sebagai jumlahan antara bias kuadrat dan variance dari *f(x)*

Bukti

$$\text{MSE } f(x) = E[f(x) - \hat{f}(x)]^2$$
$$= E[f(x) - Ef(x) + Ef(x) - \hat{f}(x)]^2$$
$$= E[f(x) - Ef(x)]^2 + 2E[f(x) - Ef(x)][Ef(x) - \hat{f}(x)] + E[Ef(x) - \hat{f}(x)]^2$$
$$= \text{Variance}(f(x)) + \text{Bias}(f(x))^2 \quad \square$$

Bandwidth optimum untuk Kernel *Gasser Muller* adalah

$$h_n = \left(\frac{\sigma^2 J_k}{f(x)[r''(x)]^2 \sigma_k^4} \right)^{1/5} n^{-1/5} \tag{6}$$

dimana $J_k = \int_{-1}^{1} K^2(u)du$, $\sigma_k^2 = \int_{-1}^{1} u^2 k(u)du$ adalah konstanta yang tergantung nilai dari *K*. Penurunan persamaan di atas dapat dilihat pada (Haerdle, 1990).

3. DATA

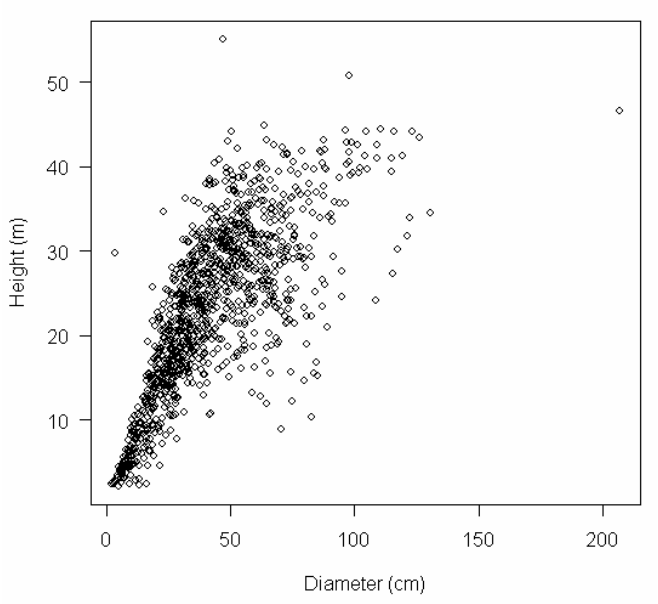
Priest River Experimental Forest's data (Pref data) adalah data penelitian hutan di Northern Idaho, USA yang dilaksanakan pada Juli-Agustus 2000. Hutan seluas 2530 hektar tersebut dibagi menjadi 9 strata yang terdiri atas 3 level ketinggian dan 3 level *solar insolation*. Pada setiap strata ditentukan 4 *cluster* yang dipilih secara acak. Observasi dilakukan pada setiap *cluster* dengan memilih beberapa pohon secara acak dan mengukur diameter pohon di ketinggian 1.3m. Hanya 85% dari pohon yang terpilih yang juga diukur tingginya. Selain itu dicatat pula *species* dari masing-masing pohon. Jelas bahwa Pref data ini mempunyai struktur hirarkis, yakni pohon terletak di dalam *cluster* yang terletak didalam strata tertentu. Namun dalam tulisan ini faktor *cluster* dan strata tidak diperhatikan.

Grafik dari tinggi versus diameter dari pohon yang di-sample memperlihatkan relasi linier antara tinggi dan diameter (lihat Gambar 2). Dari analisa data awal ternyata ada 2 variabel (*fixed effect*) yang berpengaruh pada pemodelan tinggi pohon; yakni diameter dan *species*. Namun, pemodelan dengan metode nonparametrik di atas digunakan untuk data dengan satu variabel saja. Oleh karena itu, metode-metode di atas diaplikasikan pada data dari salah satu *species* saja.

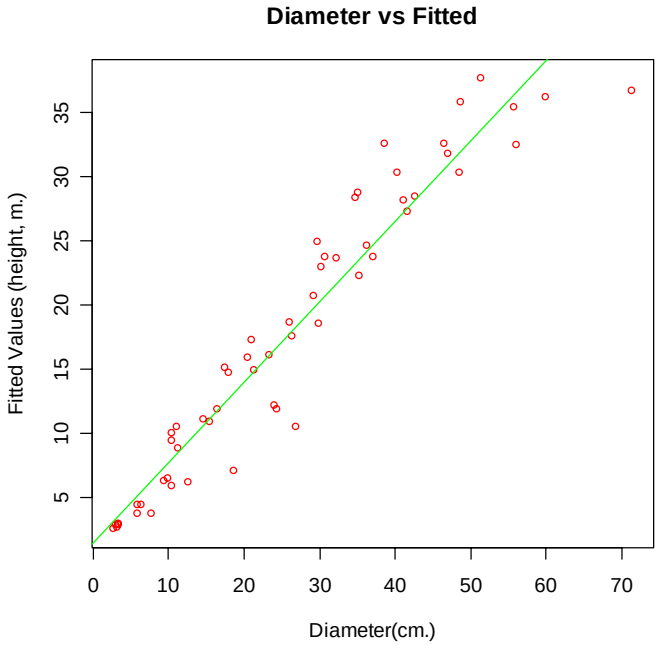
4. PENGOLAHAN DATA DAN ANALISA

Fungsi kernel yang dipakai adalah Epanechnikov kernel, dengan alasan nilai integralnya dapat dihitung secara analitik. Namun demikian kita harus memperhatikan bahwa *support* dari kernel ini berada pada selang [-1,1], sehingga pada program kita harus memisahkan apakah sebuah nilai s_i berada pada $(-\infty, 1]$, $[-1,1]$, $[-1, \infty)$, $(-\infty, -1)$ atau $(1, \infty)$. Dalam makalah ini akan ditunjukkan tiga metode, yaitu metode linear regressi klasik, dan metode nonparametrik dengan menggunkan dua kernel, Nadaraya-Watson dan Gasser-Mueller. Plot dari ketiga metode ini dapat dilihat pada Gambar 2 – Gambar 5. Sedangkan analisa pemilihan metode yang terbaik, dilakukan dengan cara sederhana, yaitu pada pengukuran standard error, lihat pada Tabel 1. Metode Gasser

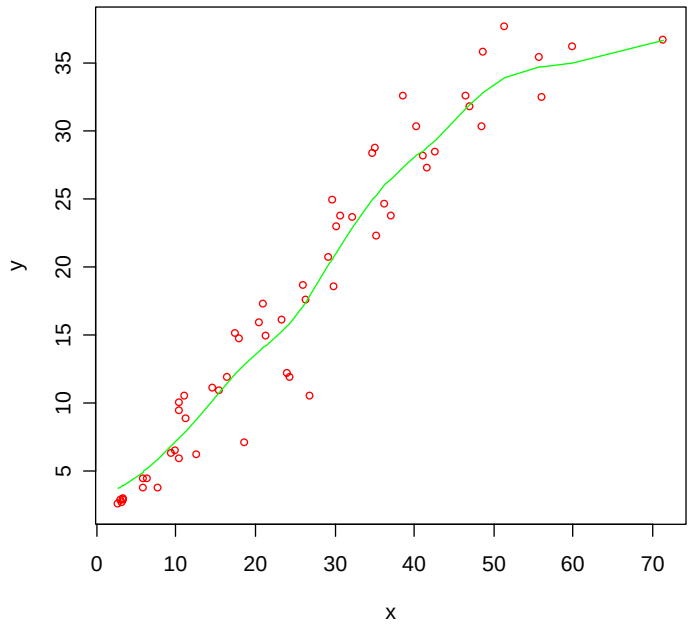
Mueller memberikan standard error yang terkecil. Standard error ini dihitung dengan mengabaikan satu nalai terakhir, hal ini karena adanya kelemahan pada metode Gasser Mueller ini pada *boundary*. Pada *boundary*, nilai x_i yang bisa dihitung pada Persamaan (5) lebih sedikit dibandingkan bila x_i tersebut berada di tengah (Hart, 1997).



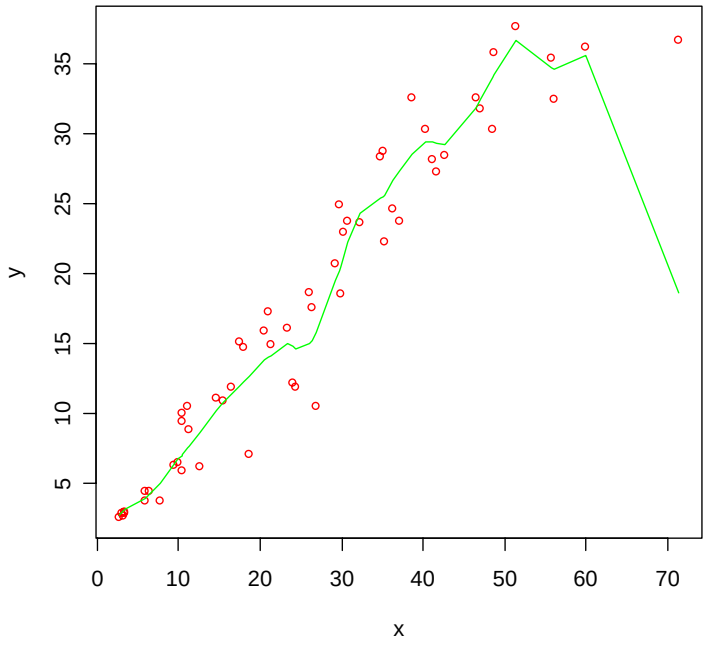
Gambar 2. Plot Tinggi (m) Versus Diameter (cm) Pohon dari Pref Data



Gambar 3. Linear Regressi



Gambar 4. Nadaraya Watson dengan Menggunakan *h* Optimum



Gambar 5. Gasser Mueller dengan Menggunakan *h* Optimum, Tanpa Perbaikan

Tabel 1. Perbandingan Metode berdasarkana pada nilai standard error

Metode	Standard error
Least Square method	3.083
Nadaraya Watson	2.455473
Gassermuller	2.397201

5. KESIMPULAN DAN CATATAN

Pada tulisan di atas telah dijelaskan beberapa macam kernel, dan cara menuliskannya dengan menggunakan Program R, namun demikian beberapa hal belum dijawab, yaitu masalah pada *boundary*. Hal ini terlihat jelas pada Gambar 5, pada bagian ujung kurva yang dibangun dengan pendekatan Gasser Mueller, kurva menjadi terlalu bias. Untuk itu biasanya digunakan koreksi pada regresi Nonparametrik ini. Pendekatan yang digunakan untuk memperbaiki masalah di *boundary* dapat dilihat, misalnya, Gasser dan Mueller (1979), Mueller (1987).

DAFTAR PUSTAKA

Eubank, R.L., 1988, *Spline smoothing and Nonparametric Regression*, Marcel Dekker, New York..

Gasser, T. and Mueller, H. G., 1979, Kernel estimation of regression functions, in Gasser and Rosenblatt (eds), *Smoothing Techniques for Curve Estimation*, Springer Verlag, Heidelberg.

Haerdle, W, 1990, *Introduction to Nonparametric Regression*, http://www.quantlet.de.

Hart, J.D, 1997, *Nonparametric Smoothing and Lack-of-Fit Test*, Spinger, New York.

Mueller, H. G., 1987, Weighted local regression and kernel methods for nonparametric curve_tting, *Journal of the American Statistical Association* 82: 231{8.

Silverman, B.W., 1986, *Density Estimation for statistics and Data Analysis*, Chapman & Hall, London.

LAMPIRAN:

Program 1

```
window <- function(h,x,y)
{ n <- length(x)

  f  <- c()
  for(i  in 1:n)
  { count1 <- 0
    count2 <- 0
    for(j in 1:n)
    { if(x[i]-h<= [j] & x[j] <=
x[i]+h)
      { count1 <- count1 +1
        count2 <- count2 + y[j]
        if (count1 == 0) temp <-
y[j]
      }
    }
    if (count1 != 0)
      f <-
append(f,count2/count1)
    else
      f <- append(f,temp)
   }

return(f)
}
```

Program 2

```
NW <- function(h,x,y)
{ n <- length(x)
  f <- c()
  for(i in 1:n)
  { x1 <- (x-x[i])/h
    K1 <- sum(KG(x1))
    K2 <- t(y)%*% KG(x1)
    f <- append(f,K2/K1)
  }
  return(f)
}

KG <- function(x)
{ kern <- exp(-
0.5*x^2)/sqrt(2*pi)
return(kern)
}

KEpach <- function(x)
{ if (abs(x) < sqrt(5))
   KEpach <- 0.75*(1-
0.2*x^2)/sqrt(5)
 else
   KEpach <- 0
 return(KEpach)
}
```

Program 3

```
GM <- function(h,x,y)
{ n <- length(x)
  f <- c()
  s <- x
  x[n+1] <- (71-0.5)/n
  for (i in 1:n)
    s[i] <- (x[i] + x[i+1])/2
  for(i in 1:n)
  { count <- 0
    for(j in 1:n)
    { if ((j-1)!=0)  a  <- (x[i]-
s[j-1])/h
      else    a    <- x[i]/h
      b    <- (x[i]-s[j])/h
    if (abs(a) <= 1 & abs(b)
<= 1)
       count<-count+
y[j]*(IntEpach(a)-
IntEpach(b))
    else  if (abs(a) <= 1 & b <
-1)
         count <- count +
       y[j]*(IntEpach(a)-
IntEpach(-1))
    else  if (abs(b) <= 1 & a >
1)
         count<-count+
     y[j]*(IntEpach(1)-
IntEpach(b))
    else  if (b < -1 & a > 1)
         count <- count + y[j]}
   f <- append(f,count) }
  return(f)}

IntEpach <- function(x)
{ if (abs(x) <= 1)
    KEint <- 0.75*x -
0.25*x^3
  else  KEint <- 0
  return(KEint)}
```