NON PARAMETRIC REGRESSION

DR. A. WAITITU

Table of Contents

COURSE OUTLINE

1. General non-parametric regression models in both fixed and stochastic design
2. Estimation of regression functions using common smoothing techniques, Kernel NN (Nearest Neighbor), spline and local polynomial
3. Properties of estimators
4. Choice of smoothing parameters, measures of estimation quality, rates of convergence
5. Bandwidth selection by cross-validation.
6. Asymptotic distribution of the estimates, boundary kernels and boot-strap method.
7. Orthogonal series expansion and wavelets: Fourier and other orthogonal series, density and regression estimates and windowed Fourier transform.
8. Neural networks: from perception to non-linear neuron, neural network estimates and their properties and network specification.

◀◀ ▶▶ ◀ ▶ ◀ Doc Doc ▶ Back Close

1. ESTIMATION OF REGRESSION FUNCTIONS USING SMOOTHING TECHNIQUES

The term smoothing can be defined as the approximation of the mean response function $M(x)$ in the regression relationship $y = m(x) + e$ given the data set (X, Y) where X is matrix e.g. $n\ X\ p$, Y is the vector.

Quote:

If m is believed to be smooth, then the observations at X_i near X should contain information about the value of m at x. Thus it should be possible to use smoothing like a local average of the data X to construct an estimation of $m(x)$. REubank (1988, P.7)

Every smoothing level is of the form $\hat{M}(x) = \frac{1}{n}\sum_{i=1}^{n} W_{ij} Y_j$ where W_{ij} are weights.
NB: $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \bar{Y} = \frac{1}{n}\sum Y_i$ where $W_{ij} = 1$
The regression estimator $\hat{M}(x)$ is called a *smoother* while the outcome of the smoothing procedure is called the *smooth*.

The major smoothing techniques are kernel, splines, K-NN(K-Nearest Neighbour) orthogonal series and local polynomial.

1.1. Kernel Smoothing

Simple approach to presenting the weight sequence $\{W_{ij}(x)\}_{i=1}^{n}$ is to describe the shape of the weight function $W_{ij}(x)$ by a density function with a scale parameter that adjust the size and the form of the weights near x. Such a density function is referred to as *Kernel (K)*.

A Kernel type smoother is a type of local average smoother that, for each target point x_i in predictor space calculates a weighted average space $\hat{M}(x_i)$ $(i.e.)$ $\hat{Y}_i$ of the observations of the target points, that is:

$\hat{Y}_i = \hat{M}(x_i) = \sum_{j=1}^{n} W_{ij} Y_j$ where $W_{ij} = \dfrac{K\left(\frac{x_i - x_j}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h}\right)}$

$\hat{Y}_i = \hat{M}(x_i) = \dfrac{\sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h}\right) Y_j}{\sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h}\right)}$ $i = 1, 2, \cdots, n$

$$\hat{Y}_i = \hat{M}(x_i) = \frac{\sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h}\right) Y_j}{\sum_{j=1}^{n} K\left(\frac{x_i - x_j}{h}\right)} i = 1, 2, \cdots, n \tag{1}$$

Equation 1 is called the *Nadaraya Watson Estimator*.

The parametric h in the equation of weights is the bandwidth parameter which determines how large a neighborhood of the target point is used to calculate local average.

A very large bandwidth generates an over-fitted curve while a very small bandwidth generates a wigglier curve. An optimal bandwidth generates a smooth curve. Therefore the choice of bandwidth is very crucial in Kernel smoothing.

Several Kernel functions exists:

Kernel	Function
Epanechnikov	$\frac{3}{4}\left(-u^2 + 1\right) I(\mid u \mid \leq 1)$
Gauss/ Normal	$(2\Pi)^{\frac{-1}{2}}\ exp\left(\frac{-u^2}{2}\right)$
Uniform	$\frac{1}{2} I(\mid u \mid \leq 1)$
Triangular	$(1 - \mid u \mid) I(\mid u \mid \leq 1)$
Quartic	$\frac{5}{6}\left(1 - u^2\right)^2 I(\mid u \mid \leq 1)$

where $u = K\left(\frac{x_i - x_j}{h}\right)$

Note:

What matters when fitting data is the choice of h and not the type of Kernel.

◀◀ ▶▶
◀ ▶
◀ Doc Doc ▶
Back Close

The Kernel function $K(u)$ has the following properties:

- $\sharp$ $K(u) \geq 0 \; \forall \; u$ (it has positive density)
- $\sharp$ $\int_{-\infty}^{\infty} K(u)\,du = 1$ because it is a pdf and the weight add upto 1
- $\sharp$ $K(-u) = K(u) \; \forall \; u$ (They are symmetrical)
- $\sharp$ $\int_{-\infty}^{\infty} uK(u)\,du = 1$

The first two properties are those of probability density function and the third property implies that the Kernel density is symmetrical. Equation 1 on the preceding page is called *Nadaraya Watson estimator.*

Kernel Smoothing in R

Nadaraya Watson estimation in R is done as follows:
ksmooth(X,Y), kernel="distribution",bandwidth=h,range.x=rang(X), n-points=max(100,length(X),x.points)
where
X-input x values
Y-input y values
kernel-the kernel to be used
bandwidth- value of bandwidth to be used
range.x- range of points to be covered in output (optional)
n.points- the number of points of which we evaluate the smooth fit
x.points- points at which to evaluate a smooth fit n points are chosen uniformly to cover a range x if x points are missing(optional)

EXERCISE 1. Consider the following data representing the speed and distance covered by different cars.

Speed	4	4	7	7	8	9	10	10	10	11	12	12	12	12
Distance	2	10	4	22	16	10	18	26	34	17	14	20	24	88

Speed	13	13	13	13	14	14	14	15	15	15	16	16	17	17
Distance	26	34	34	46	26	36	80	20	26	54	32	40	32	40

1. Plot a scatter diagram of distance. (y axis)against speed (x axis)
2. In plot 1 above add filled Nadaraya Watson fitted estimate/ smooth using a bandwidth of 1.5 the Gauss Normal Kernel.
3. In plot 1 and 2 above superimpose the fitted Nadaraya Watson estimates using a bandwidth of 6. Comment on the effect of the bandwidth and through trial and error method determine the optimal bandwidth for the data.
4. Repeat 1-3 above using box density.

Remarks on Kernel Smoothing

In Kernel smoothing values of $Y_i$such that $X_j$be close to $X_i$get relatively heavy weights while values of $Y_j$such that $X_j$is far from $X_i$gets more or zero weight. The value parameter h determines the width of $K\left(\frac{x_i - x_j}{h}\right)$ and hence controls the size of the region around $X_i$for which $Y_i$receives relatively large weights.

Since bias increases and variance decreases with increasing bandwidth h, selection of h is a compromise between bias and variance in order to achieve small squared errors. In practice, this is usually done by trial and error.

1.2. K-Nearest Neighbor Estimates (K-NN)

Unlike Kernel estimates, K-NN is a weighted average in a varying neighborhood. The neighborhood is defined through those X variables which are among the K nearest neighbors of X in Euclidean distance (linear real distance). The K-NN smoother is defined as:

$$\hat{M}_k(x) = \frac{1}{n}\sum_{i=1}^{n} W_{ki}(x)\,Y_j \tag{2}$$

where $\{W_{ki}(x)\}^n$ is a weight sequence defined through a set of indices

$J_x = \Big\{ i \quad x_i\; is\; one\; of\; the\; K-Nearest\; observations\; to\; x$

The K-NN weight sequence defined is then constructed using $J_x$instead of indexes of neighboring observations as follows:

$$W_{ki}(x) = \left\{ \begin{array}{c} \frac{1}{k}\,;\; if\; i \in J_x \\ 0\,;\; otherwise \end{array} \right\}$$

Example. Let $\{(X_i, Y_i)\}_{i=1}^{5}$ be $\{(1,5),(7,12),(3,1),(2,0),(5,4)\}$. Compute (K-NN estimate) $\hat{M}_k(x)$ (*i.e.*) for x=4 and k=3.

Solution:
$J_x = J_4 = (2,3,5)\;\; Nearest\; observation\; to\; 4$
therefore

$$\begin{aligned}
W_{k1}(4) &= W_{31}(4) = 0 \\
W_{k2}(4) &= W_{31}(4) = 0 \\
W_{k3}(4) &= W_{31}(4) = \frac{1}{3} \\
W_{k4}(4) &= W_{31}(4) = \frac{1}{3} \\
W_{k5}(4) &= W_{31}(4) = \frac{1}{3}
\end{aligned}$$

$\hat{M}_k(x) = \sum_{1=1}^{n} W_{ki}(x)Y_i$
$\hat{M}_3(4) = \sum_{1=1}^{3} W_{ki}(x)Y_i$
$\hat{M}_3(4) = \left(\frac{1}{3}*1\right) + \left(\frac{1}{3}*0\right) + \left(\frac{1}{3}*4\right) = \frac{5}{3}$
similarly
$\hat{M}_3(1) = \left(\frac{1}{3}*0\right) + \left(\frac{1}{3}*1\right) + \left(\frac{1}{3}*4\right) = \frac{5}{3}$
$\hat{M}_3(2) = \left(\frac{1}{3}*5\right) + \left(\frac{1}{3}*1\right) + \left(\frac{1}{3}*4\right) = \frac{10}{3}$
$\hat{M}_3(3) = \left(\frac{1}{3}*5\right) + \left(\frac{1}{3}*0\right) + \left(\frac{1}{3}*4\right) = \frac{9}{3}$
$\hat{M}_3(5) = \left(\frac{1}{3}*0\right) + \left(\frac{1}{3}*1\right) + \left(\frac{1}{3}*12\right) = \frac{13}{3}$
$\hat{M}_3(7) = \left(\frac{1}{3}*0\right) + \left(\frac{1}{3}*1\right) + \left(\frac{1}{3}*4\right) = \frac{5}{3}$
$\hat{M}_3(6) = \left(\frac{1}{3}*1\right) + \left(\frac{1}{3}*4\right) + \left(\frac{1}{3}*12\right) = \frac{17}{3}$
if k=4
$\hat{M}_4(1) = \frac{1}{4}(12+1+0+4) = \frac{17}{4}$
$\hat{M}_4(2) = \frac{1}{4}(12+1+0+4) = \frac{17}{4}$
$\vdots$
$\hat{M}_4(7) = \frac{1}{4}(5+1+0+4) = \frac{10}{4}$
Without loss of generality we allow $X_i \in J_x$. Then if k=n=5 we have:
$\hat{M}_5(1) = \frac{1}{5}(5+12+1+0+4) = \frac{22}{5}$
$\hat{M}_5(2) = \frac{1}{5}(5+12+1+0+4) = \frac{22}{5}$

$\vdots$
$\hat{M}_5(7)=\frac{1}{5}(5+12+1+0+4)=\frac{22}{5}$ □

This implies that if $k=n$, then we use the observations and $\hat{M}(x)$ simply becomes the sample average of Y_i, that is, $\hat{M}(x)$ produces a perfectly flat estimated mean function. In this case the variance is relatively small since we are using a lot of data to find average. However, if the time mean function $M(x)$is not actually constant then $\hat{M}(x)$ will be biased for many particular values of x.

If k=1, we have:
$\hat{M}_1(1)=\frac{1}{1}(5)=5$
$\hat{M}_1(2)=\frac{1}{1}(0)=0$
$\hat{M}_1(3)=\frac{1}{1}(1)=1$
$\vdots$
$\hat{M}_1(7)=\frac{1}{1}(12)=12$

This implies that when $k=1$, the observations are reproduced at X_i so that for an x between two adjacent predictable variables a step function is obtained with a jump in the middle between the two observations. However for $k=1$, one just uses one observation in sample average. The bias is relatively very small since we are using observations with x_i very close to x. This is at the cost of high variance since we are using very few observations, therefore, k is the turning parameter that optimizes the trade-off between bias and variance.

Note that:
$bias^2=\left(E\left[\hat{M}(x)\right]-M(x)\right)^2$ and
$Variance=E\left[\left(\hat{M}(x)-E(\hat{M}(x))\right)^2\right]$

Drawback

K-NN is not always used because it does not generalize well the particularly in higher dimensions. The method is very strongly affected by high dimensions (curse of dimensionality)

1.3. Local Polynomial Regression

Recall that with parametric polynomial model in one variable is given by:

$$M(x)=B_0+B_1x+B_2x^2+\dots B_px^p \tag{3}$$

Model 3 is called the p^{th}order model. In particular a 2^{nd} order parametric polynomial model in one variable is given by:

$$M(x)=B_0+B_1x+B_2x^2$$

and for two variables is given by :

$$M(x)=B_0+B_1x_1+B_2x_2+B_{11}x_1^2+B_{22}x_2^2+B_{12}x_1x_2$$

Remarks

- ♯ Polynomial models are useful in situations where the analyst knows that the curvilinear effects are present in the true response function.
- ♯ Polynomial models are also useful as approximating functions to unknown and possibly very complex non-linear relationship.
- ♯ Polynomial models can be Taylor series expansion of the unknown function.

Local polynomial regression is also called *Locally Weighted Regression* (LOWESS).

LOWESS uses the data from a neighborhood around the specific location, so that the neighborhood is defined as a span, which is the function of the total points used to form the neighborhood.

E.g. a span of 0.5 indicates that the closest half of the total data points is used as the neighborhood. The LOWESS procedure then uses the points in the neighborhood to generate a weighted least squares estimates of the specific response.

NB: The weights are based on the distance of the points used in the estimation from the specific location of interest.

- **Simple Regression (Univariate case)**

The model is

$$Y_i = f(x_i) + e_i$$

Evaluating the regression function at a particular x-value, say x_0 (note that we will finally fit the model at a representative range of values of x, or sample at the end observations, x_i, i= $1, 2.........n$)

We perform a p^{th}order weighted least squares polynomial regression of Y on x as follows:

The model

$$Y_i = a + b_1(x_i - x_0) + b_2(x_i - x_0)^2 + ...b_p(x_i - x_0)^p + e \tag{4}$$

- **The LOWESS Procedure**

1. Take a point, say x_0. Find the K-NNs of x_0, which constitute a neighborhood N(x_0). The number of neighbors, K, is specified as a percentage of the total number of points to be spanned.

2. Calculate the largest distance between x_0 and another point in the neighborhood as follows :
$$\triangle x_0 = Max_{N(x_0)} \mid x_0 - x_i \mid$$

3. Assign weights to each point in $N(x_0)$ using the tri-cube weight function
$$W = \left(\frac{\mid x_0 - x_i \mid}{\triangle x_0}\right) \ where \ W(u) = \begin{cases} (1-u^3)^3 & ; \ 0 \le u \le 1 \\ 0 & ; \ otherwise \end{cases}$$

4. Calculate the weighted Least square fit of $Y_j$in the neighborhood $N(x_0)$. Take the fitted value,
$\hat{Y}_0$=S(x_0)=a
$\hat{Y}_i$=arg min$\sum_{i=1}^{n}(Y_i - a - b_1(x_1 - x_0) - bp(x_i - x_0)^p)^2 W_i$
for the i^{th} predictor value

5. Repeat for each predictor value

Remarks

♯ The larger the span the smoother the results.

♯ The larger the order of the local regression p, the more flexible the smooth.

Example. Fit a simple local linear regression model to the following data (Windmill data) Velocity-x, Output-y

1. Read y in data. Plot(x,)
2. Use fuction LOWESS (. . .=F=. . .)where F is the span (with max of 1) to fit the model.
3. Superimpose

Serial Number	Velocity	Temperature	Output
1	2.5	18	0.1
2	2.8	17	0.5
3	3.0	19	0.7
4	3.2	16.5	0.5
5	3.8	20	1.1
6	3.9	20.5	1.2
7	4.0	20	1.1
8	4.1	21	1.15
9	4.8	21.5	1.6
10	5.0	21	1.6
11	5.5	21	1.6
12	6.0	21.5	1.6
13	6.2	20	1.7
14	6.3	22	1.8
15	6.4	22.5	1.9
16	7.0	21	1.2
17	7.5	23	2.0
18	8.0	24	2.2
19	8.2	23	2.2
20	8.5	23.5	1.9
21	9.0	24.5	2.2
22	9.4	25	2.0
23	9.5	25	2.2
24	10.0	24.5	1.9
25	10.2	25	2.0
26	10.3	24.5	1.9

Solution:
$wmill <-read.table(''c:input\backslash lowess.text'', header=TRUE)$
$x <-wmill\&velocity$
$y <-wmill\&output$
$plot(x,y)$
$lowess(x,y,0.5)$
$simpose <-lowess(x,y,0.5)$
$lines(simpose)$
$Thentryforvariousvaluesof F(maximumis1)$
$alternatively$
$serialno <-c(1:26)$
$velocity <-c()$
$temperature <-c()$
$output <-c()$
$plot(x,y)$
$lowess(x,y,0.5)$
$k <-lowess(x,y,f=0.5)$
$lines(k)$ □

- **Multiple Regression**

$$\begin{aligned} Y_i &= f(x_i) + e_i \\ &= f(x_{i1},\, x_{i2} \ldots x_{ik}) + e_i \end{aligned}$$

- **Fitting a Local Polynomial Regression Curve**

1. Define a multivariate neighborhood around a focal point $x_0' = (x_{01},\, x_{02},\, \ldots,\, x_{0k})$ so that

$$D(x_i, x_0) = \sqrt{\sum_{j=1}^{k} (Z_{ij} - Z_{0j})^2}$$

where $Z_{ij}$are the standard predictors;$Z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$ $\bar{x}_j$is the mean of the j^{th} predictor and s_j is the standard deviation.

2. Determine the scaled distance $W_i = W\left[\frac{\triangle(x_i, x_0)}{h}\right]$where $W(\cdot)$is suitable weight function such as the tri-cube.

3. Perform a weighted regression of Y on the $x's$ e.g. a local linear fit takes the following form

$$Y_i = a + b_1(x_{i1} - x_{01}) + b_2(x_{i2} - x_{02})^2 + \ldots .. b_k(x_{ik} - x_{0k}) + e_i$$

Note that $\hat{Y}_0$=a

The procedure is repeated for representative combinations of predictor values to build up a picture of the regression surface.

- **Local Regression Polynomial in R**

Fit a polynomial regression to the windmill data, that is, output (Y), Velocity (X_1), Temperature (X_2), Span (0.5), Degree. Use function Lowess, then plot.

Y<-wmill&output
X_1<-wmill&velocity
X_2<-wmill&temperature
multiplereg<-loess(formula=Y~$X_1 + X_2$,span= 0.5,degree=2)
velocity.inc<-seq(min(X_1), max(X_2), ln 26)
temp.inc<-seq(min(X_1), max(X_2), ln 26)
m=velocity.inc, a=temp.inc
newdata<-expand.grid($X_1 = m,\ X_2 = 9$)
fit.output<-matrix(predict(multiplereg, new.data), 26, 26)
multiplot<-persp(velocity.inc, temp.inc, fit.output, theta=45, phi=35, main="Multiple Regression", xlab="velocity", ylab="temperature", zlab="fitted output", sub="Example1")

2. ASYMPTOTIC DISTRIBUTION OF THE ESTIMATES, BOUNDARY KERNELS AND BOOT-STRAP METHOD

2.1. Boundary Kernel

Any smoothing method becomes less accurate near the boundary of the observation interval. This is because at the boundary fewer observations can be averaged and thus variance or bias can be affected. In particular Kernel weights become asymmetric as x approaches the boundary weights. However the "boundary effect" is not present for x in the interior of the observation interval.

The small and moderate the sample size, a significant population of the observation interval can be affected by the boundary behavior.

A solution to boundary effect was suggested by Rice (1984) and uses the generalized Jack Knives technique.

Consider the fixed design error model with kernel having support [-1,1]. Let us have the following kernel estimator.

$M_h\left(x\right)$which has expectation to

$\int_{x-\frac{1}{h}}^{\frac{x}{h}} K\left(u\right)m\left(x-uh\right)du + On^{-1}h^{-1}$

as $nh \to \infty$. In the middle of the observation interval, there is no problem since for h $\frac{x}{h} \leq 1$ and $\frac{x-1}{h} \leq -1$. h is small.

Now let $x = ph \leq 1-h$

Then by a Taylor series expansion, the expected value of $\hat{M}_h\left(x\right)$ can be approximated by

$$\int_{-1}^{\rho} K\left(u\right)du - hm^{'}\left(x\right)\int_{-1}^{\rho} uK\left(u\right)du + \frac{1}{2}h^2m^{''}\left(x\right)\int_{-1}^{\rho} K\left(u\right)du \tag{5}$$

$$= m\left(x\right)W_k\left(0,\rho\right) - hm^{'}\left(x\right)W_k\left(1,\rho\right) + \frac{1}{2}h^2m^{''}\left(x\right)W_k\left(2,\rho\right) \tag{6}$$

Note that if $\rho \geq 1$, $W_k\left(0,\rho\right)=1$, $W_k\left(1,\rho\right)=0$, $W_k\left(2,\rho\right)=d_k$, $d_k$is a constant

In this case, one has the bias of expansion for the Prestley-char estimator. To solve the problem of boundary Rice (1984b) defines a kernel depending on the relative location of x expressed through the parameter ρ. Under this method asymptotic unbiasedness is achieved for the kernel $K_p\left(\cdot\right) = \frac{K(\cdot)}{W_k(0,\rho)}$

If x is away from the left boundary, that is, $\rho \geq 1$ then the approximate biased is given by the third term equation 5 second term will reduce to zero and $m\left(x\right)$ for the first term.

If $\rho > 1$ the second term is of dominant order $O\left(h\right)$and thus the bias is of lower order at the boundary than in the center of the interval. The generalized Jack Knife technique allows one to eliminate the lower order terms. Let $\hat{M}_{n,p}(.)$ to be kernel estimate with K_p and let $\hat{M}_h^J\left(x\right) = \left(1-R\right)\hat{M}_{h,p^{(x)}}\left(x\right) + R\hat{M}_{h,p^{(x)}}\left(x\right)$ be the Jack Knife estimator of $m\left(x\right)$which is a linear combination of kernel smoother with bandwidth h and αh. (NB: At boundary the band width should be small)

From the biased expansion of equations 5 and 6 the leading biased term with $\hat{M}_h^J(x)$ can be eliminated if R is given as

$$R = \frac{W_k\left(1,\rho\right)\diagup W_k\left(0,\rho\right)}{\alpha W_k\left(1,\frac{\rho}{\alpha}\right)\diagup W_k\left(0,\frac{\rho}{\alpha}\right) - W_k\left(1,\rho\right)\diagup W_k\left(0,\rho\right)} \tag{7}$$

where the numerator and the denominator are all constants. In effect the Jack Knife estimator is using Kernel estimator

$$K_p^J\left(u\right) = \left(1-R\right)K\left(u\right) - \frac{R}{J^\alpha}K\left(\frac{u}{\alpha}\right) \tag{8}$$

where R and $\alpha$and thus K_p^Jdepend on ρ. Therefore $K_p^J(u)$ is boundary kernel. A good choice of α is $2-\rho$

Quiz

1. Use any boundary kernel to fit a regression curve to the windmill data.
2. Superimpose on the above curve the ordinary regression curve

3. BANDWIDTH SELECTION BY CROSS VALIDATION

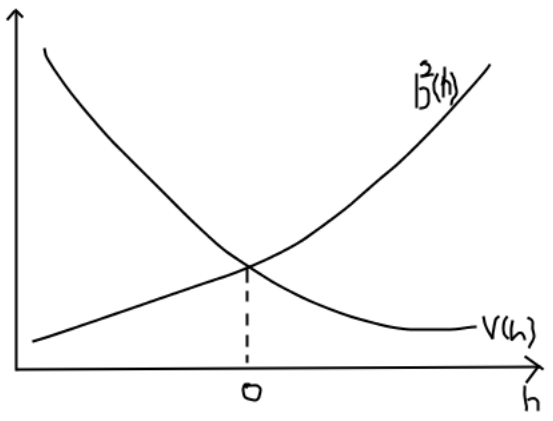
The accuracy of Kernel smoother as estimators of $M(x)$ or $M^{'}(x)$ is a function of $K(\cdot)$ which is the kernel and the bandwidth h. However, the accuracy mainly depends on the smoothing parameter h. The following distances can be used to optimize quadratic error measures for the regression curve and its derivatives.

$$\begin{aligned} A(h) &= \frac{1}{n}\sum_{i=1}^{n}\left(\hat{M}_h(x_j) - M(x_j)\right)^2 W(x_j) \\ B(h) &= \int\left(\hat{M}_h(x) - m(x)\right)^2 W(x) f(x)\,dx \\ C(h) &= E\left(A(h) \mid x_1, \cdots, x_n\right) \end{aligned}$$

where $W(x)$ is a non-negative weight function e.g. $W(x) = I\left(\mid x - \frac{1}{2} \mid < 0.4\right)$ so that the form of the above distance is determined by a variance component which decreases in h and a $bias^2$ component which increases in h.

Example. Using C(h), Show that the $bias^2$ is
$b^2(h) = bias^2 = \frac{1}{n}\sum_{j=1}^{n}\left[\frac{1}{n}\sum W_{hi}(x_j)m(x_i) - m(x_j)\right]^2 \mathrm{W}(x_j)$
and variance component is
$V(h) = \frac{1}{n}\sum\left[\frac{1}{n^2}\sum W_{hi}(x_j) - \sigma^2(x_j)\right]W(x_j)$
where W_{hi} are the Nadaraya-Watson weights

Solution: The relationship between $b^2(h)$ *and* $V(h)$ □



The decreasing curve $V(h)$ shows that $V(h)$ is roughly proportional to h^{-1}

Remark

A good start for guessing optimal h is $h_0 = n^{-\frac{1}{5}}$

The basic idea behind selecting an optimal bandwidth is to estimate the averaged squared error (ASE) given by:

$$\begin{aligned} ASE &= A(h) = \frac{1}{n}\sum_{i=1}^{n}\left[m^2(x_j) + M_h^2(x_j) - 2\hat{M}_h(x_j)m(x_j)\right]w(x_j) \\ &= \frac{1}{n}\sum_{i=1}^{n}m^2(x_j)W(x_j) + \frac{1}{n}\sum_{i=1}^{n}M_h^2(x_j)W(x_j) - 2\frac{1}{n}\sum_{i=1}^{n}\hat{M}_h(x_j)m(x_j)w(x_j) \quad (9) \end{aligned}$$

In order to estimate ASE in equation9, the first term on the right hand side is independent of the smoothing parameter h. The second term can be computed entirely from the data. If the third term could be estimated, and if it vanished faster than h tends to 0, then a device for selecting the bandwidth could be established.

The cross validation method is based on regression smoothens in which one , say the j^{th} observation is left out. This method is also called the Leave-One-Out method.

Let

$$m_{hij} = \frac{1}{n} \sum_{i \neq j}^{n} W_{hj}(x_j) Y_i \tag{10}$$

From the modified smoother equation10 one forms the cross validation function given by:

$$CV(h) = \frac{1}{n} \sum_{j=1}^{n} \left[Y_i - \hat{M}_{hij}(x_j) \right]^2 W(x_j) \tag{11}$$

so that$\hat{h} = argmin\ (CV(h))$

Equation 11 is an estimate of the third term in equation9

Note

The reason why Cross Validation works is that the cross product term from $CV(h)$ is given by:

$$CV_2(h) = -2n^{-1} \sum_{j=1}^{n} e_j \left[n^{-1} \sum_{i \neq j}^{n} Wh_i(x_j) Y_i - m(x_j) \right] W(x_j) \tag{12}$$

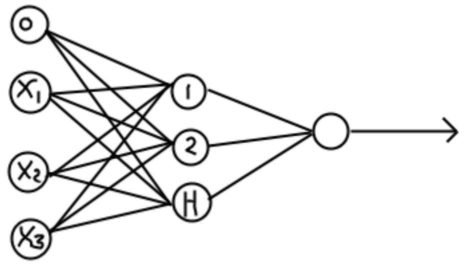
Cross product term - Elements along the diagonal and has expectation zero.

Determine the optimal bandwidth for the velocity (windmill data) using the cross-validation technique.

Use the bandwidth above to fit a smooth function to the velocity data.

4. ARTIFICIAL NEURAL NETWORK (ANN)

ANN is a parallel connection of a set of nodes called neurodes (weights). We define an ANN with an input larger, hidden layer and an output layer.



The above input-output map, we have d input nodes, one layer of H hidden nodes and an activation function $\psi(x)$. The input at hidden layer nodes are connected by weights W_{hj} for $h \in (1, \cdots, H)$ and $j \in (0, \cdots, d)$ where W_{h0} is the bias for the i^{th} hidden node. The hidden and output layers are connected by weights α_h for $h \in (0, \cdots, H)$. Considering an input vector $x = (x_1, x_2, \cdots, x_d) \in \mathbb{R}^d$ the input $V_h(x)$ to the h^{th} hidden node is the value

$$V_h(x) = W_{h0} + \sum W_{hj} x_j \tag{13}$$

for example, for $h = 2$ $V_2(x) = W_{20} + \sum W_{2j} x_j$
The output $\phi h(x)$ of the h^{th} hidden node is the value

$$\phi h(x) = \varphi(V_h(x)) \tag{14}$$

The net input to the output node is the value

$$\mathbb{Z}(x) = \alpha_0 + \sum \alpha_n \phi_n(x) \tag{15}$$

Finally the output $Z(x)$ of the net is the value

$$Z(x) = \psi(\mathbb{Z}(x)) \tag{16}$$

The connection/ weights are adjusted through training. There exists two training paradigms: Non Supervised and Supervised training. We discuss and later apply supervised learning. The Supervised training of a neural network requires the following:

A sample of d input vectors , $X = (x_1, x_2, \cdots, x_d)$ of size n and an associated output $Y = (y_1, y_2, \cdots, y_n)$

The selection of an initial weight set.

We need a repetitive metor to update the current weights to optimize the input output map.

4.1. Stopping Rule

There are two methods used to train a neural network, namely the maximum likelihood estimator (MLE) method and the sum of squared errors (SSE) method. We discuss and apply the SSE method.

- **The Sum of Squared Errors Method**

The SSE is used to train faced forward networks. In this methods the weights are adjusted in such a way that the SSE between the targets y and the goal of output Z is minimized.

The SSE is defined as:

$$\begin{aligned} S^2(x_i;\theta) &= \sum (Y_i - Z(x_i;\theta))^2 \\ &= \sum (Y_i - Z(x_i;W,\alpha))^2 \\ &= \sum \left(Y_i - \psi\left(\alpha_0 + \sum \alpha_n \psi\left(W_{h0} + \sum W_{hj}x_j\right)\right)\right)^2 \end{aligned} \tag{17}$$

which is an activation function
There exists two activation functions
The Unipolar activation function
It takes the form

$$\begin{aligned} \psi(x) &= \frac{exp(a(x-b))}{1+exp(a(x-b))} \\ &= \frac{1}{1+exp(-a(x-b))} \end{aligned} \tag{18}$$

so that

$$\psi(x) = \begin{cases} 1, & x \to \infty \\ 0, & x \to -\infty \end{cases} \tag{19}$$

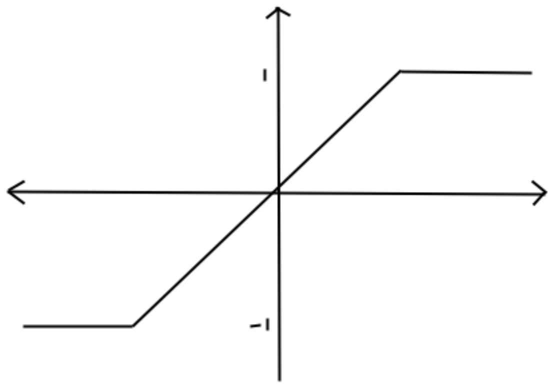
a is the learning rate and b is the bias. Since the unipolar activation function maps $\mathbb{R} \to [0,1]$, it is very practical to Bernoulli or Binomial data.
Bipolar activation function
It takes the form

$$\begin{aligned} \psi(x) &= 2\left\{\frac{1}{1+exp(-ax)}\right\} \\ &= tanh\left(\frac{ax}{2}\right) \end{aligned}$$

so that

$$\psi(x) = \begin{cases} 1, & x \to \infty \\ 0, & x \to 0 \\ -1, & x \to -\infty \end{cases}$$

and $\psi(x) + \psi(-x) = 0 \;\Rightarrow symmetrical$



Both the Unipolar and Bipolar sigmoids are continuously differentiable.
The other name for this sigmoid is logistic function.

4.2. Training of the Network

This step involves updating the weights until the function $S^2(x_i, \theta)$ is minimized. There are various methods of minimizing this function namely;

1. Back Propagation
2. Quasi Newton Method
3. Simulated Annealing Method

- **Back Propagation**

In this method and taking a unipolar $\psi(x)$, the weights are adjusted as follows:

$$W^{r+1} = W^r + \triangle W$$

$$\alpha^{r+1} = \alpha^r + \triangle\alpha \tag{20}$$

Taking individual weights, we have the r^{th} iteration weights as

$$\begin{aligned}\alpha_h^{r+1} &= \alpha_h^r - \lambda_1 \left\{ \frac{\partial S^2\left(x_i;\theta^{(r)}\right)}{\partial \alpha_h} \right\} \\ &= \alpha_h^r + \lambda_1 \left\{ Y_i - Z\left(x_i;\theta^{(r)}\right) \right\} Z\left(x_i;\theta^{(r)}\right) \left\{ 1 - Z\left(x_i;\theta^{(r)}\right) \right\} \phi_h(x_i)\end{aligned} \tag{21}$$

for $i = 1, \cdots, n$ and $h = 1, \cdots, H$
similarly

$$\begin{aligned}W_{hj}^{r+1} &= W_{hj}^r - \lambda_2 \left\{ \frac{\partial S^2\left(x_i;\theta^{(r)}\right)}{\partial W_{hj}} \right\} \\ &= W_{hj}^r + \lambda_2 \left\{ Y_i - Z\left(x_i;\theta^{(r)}\right) \right\} Z\left(x_i;\theta^{(r)}\right) \left\{ 1 - Z\left(x_i;\theta^{(r)}\right) \alpha_h^r \right\} \\ &\quad - \left\{ \phi_h(x_i)\left(1 - \phi_h(x_i)\right) \right\} x_j\end{aligned} \tag{22}$$

for $i = 1, \cdots, n$ and $h = 1, \cdots, H$ and $j = 0, \cdots, d$
and λ_1 and $\lambda_2$represent the step gain.

The weights are adjusted until the stopping criterion are met. Under this method, each weight is adjusted n, the the sample size, times each iteration. This means for I iterations, each weight is is adjusted I_n times. The method is therefore slow especially because I is normally large. The method is also not stable.

- **The Quasi-Newton Method**

This method was independently developed by four authors Broyden [4], [11], [13]. It is therefore referred to as the BFGS method.

The training starts by imputing an initial set of weights, θ^o. From θ^o, $S^2(X_i, \theta^o)$is determined.

From the principles of second-order Taylor operation, $S^2(X_1, \theta^1)$ can be found,

$$\begin{aligned}
A_o &= \frac{S^2(X_i,\theta^{0,1}+m_1,...,\theta^{0,p})-S^2(X_i,\theta^{0,1},...,\theta^{0,p})}{m_1}\\
&\vdots\\
&= \frac{S^2(X_i,\theta^{0,1},...,\theta^{0'p}+m_p,...,\theta^{0'p})-S^2(X_i,\theta^{0,1},...,\theta^{o,p})}{m_p}\\
&\vdots\\
&= \frac{S^2(X_i,\theta^{0,1},...,\theta^{0'p}+m_p)-S^2(X_i,\theta^{0,1},...,\theta^{0,1})}{m_p} \qquad (23)
\end{aligned}$$

where m_p= $max(\epsilon,\epsilon\theta^{0,p})$ with $\epsilon=10^{-6}$ for $p=1,...,p$ and $i=1,...n$.
The direct off-diagonal elements of the matrix B_o are evaluated as:

$$\begin{aligned}
\frac{\partial^2 S^{2\left(x_i;\theta^0\right)}}{\partial\theta^{0,j}\partial\theta^{0,k}} &= \frac{1}{m_j m_j}\{S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,j}+m_j,\cdots,\theta^{0,k}+m_k,\cdots,\theta^{0,p}\right)\\
&- S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,j},\cdots,\theta^{0,k}+m_k,\cdots,\theta^{0,p}\right)\\
&- S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,j}+m_j,\cdots,\theta^{0,k},\cdots,\theta^{0,p}\right)\\
&- S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,p}\right)\} \qquad (24)
\end{aligned}$$

for $j,\ k=1,\cdots,p$ and $i=1,\cdots,n$.
The direct diagonal elements of B_0 are evaluated as:

$$\begin{aligned}
\frac{\partial^2 S^{2\left(x_i;\theta^0\right)}}{\left(\partial\theta^{0,j}\right)^2} &= \frac{1}{m_j^2}\{S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,j}+m_j,\cdots,\theta^{0,p}\right)\\
&- 2S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,p}\right)\\
&+ S^2\left(x_i;\theta^{0,1},\cdots,\theta^{0,j}-m_j,\cdots,\theta^{0,p}\right)\} \qquad (25)
\end{aligned}$$

for $j,\ k=1,\cdots,p$ and $i=1,\cdots,n$.
$\theta^{'}$ is obtained by differentiating the right hand side of equation with respect to $\theta^{'}-\theta^0$ and equating the result to zero to get:

$$\theta^{'}=\theta^0-B_0^{-1}A_0 \qquad (26)$$

The quantity $B_0^{-1}A_0$ is called the direction of a change. It is a vector describing a segment of a path from iteration 0 to iteration 1. B_0 represents the "angle" of the direction, while A_0 represents the "size" of the direction.

The minimization then continuous from iteration 1 to 2 until the stopping criterion are met. In general, the iteration are given by:

$$\theta^{r+1}=\theta^r-B_r^{-1}A_r \qquad (27)$$

This method is very fast. However, the Hessian matrix B may become non-singular. This means that B_r^{-1} would be undefined. The BFGs method solves the problem by numerical approximating B_r.

The method first redefines equation as follows:

$$\theta^{r+1}=\theta^r-W_rB_r^{-1}A_r \qquad (28)$$

where W_r is called the step length and is found such that $S^2\left(x_i;\theta^r - W_r B_r^{-1} A_r\right) < S^2\left(x_i;\theta^r\right)$. This is done using various methods like step halving and golden section search.

In Step Halving, W_r is first set at 1 and the function $S^2\left(x_i;\theta^r - W_r B_r^{-1} A_r\right)$ tested for a decrease. If it fails W_r is decreased by $\frac{1}{2}$ and test carried out again. The process continues until a decrease in the function occurs. The final value of W_r is the required step length.

By letting:

$$a_r = \theta^{r+1} - \theta^r = -W_r B_r^{-1} A_r \tag{29}$$

represent the change in parameters in the r^{th} iteration and

$$b_r = A^{r+1} + A^r \tag{30}$$

the BFGs method has:

$$B_{r+1} a_r = b_r \tag{31}$$

Therfore $B_{r+1}$is the ratio of the change in the gradient to the change in the parameters. This is what is called the *Quasi Newton* condition.

The BFGs method solves equation (31) for B_{r+1} as

$$B_{r+1} = B_r + \frac{b_r b_r^t}{b_r^t a_r} - \frac{B_r a_r a_r^t B_r}{a_r^t B_r a_r} \tag{32}$$

where d_t represents the transpose of the vector d.

The BFGs update matrix B_r remains positive definite as long as $b_r^t a_r > 0$ and holds automatically since $S^2\left(x_i;\theta\right)$ is strictly convex.

Having defined all the necessary equations, we now summarize the BFGs algorithm:

1. Input the initial weights $\theta^0$and B_0, an identity matrix whose size is equal to the length of θ^0.
2. Set $\sigma_r = -B_r^{-1} A_r$
3. Compute the step length W_r and determine θ^{r+1} as $\theta^{r+1} = \theta^r + W_r \sigma_r$
4. Compute the values $a_r = \theta^{r+1} - \theta^r$ and $b_r = A^{r+1} - A^r$
5. Compute B_{r+1} as $B_{r+1} = B_r + \frac{b_r b_r^t}{b_r^t a_r} - \frac{B_r a_r a_r^t B_r}{a_r^t B_r a_r}$
6. Continue with the next r until termination criterion are satisfied.

5. EXERCISES

EXERCISE 2. Question one (a)

EXERCISE 3. Question one (d)

EXERCISE 4. One (b)

EXERCISE 5. one(c)

EXERCISE 6. one (e)

EXERCISE 7. one (f)

EXERCISE 8. Question 2(a)

EXERCISE 9. two (b)

Solutions to Exercises

Exercise 1. To be attempted by the students. Exercise 1

Exercise 2.

$$\begin{aligned}\psi(x) &= 2\left\{\frac{1}{1+exp(-ax)}\right\}-1\\ &= \frac{1-exp(-ax)}{1+exp(-ax)}\\ &= \frac{exp(ax/2)\{1-exp(-ax)\}}{exp(ax/2)\{1+exp(-ax)\}}\\ &= \frac{\{exp(ax/2)-exp(-ax/2)\}/2}{\{exp(ax/2)+exp(-ax/2)\}/2}\\ &= \frac{sinh(ax/2)}{cosh(ax/2)}\\ &= tanh(ax/2)\end{aligned}$$

$$\begin{cases}\psi(x)\to 1 & as\ x\to\infty\\ \psi(x)\to -1 & as\ x\to\infty\\ \psi(x)+\psi(-x)=0 & \end{cases}$$

Exercise 2

Exercise 3.
incomedata<-read.table("C:
Sta3107Data
salary.txt",header=TRUE)
par(mfrow=c(2,2))
plot(incomedata$age, incomedata$income, main= Neighborhood for x0=6, Span=13, xlab= "age", ylab= "income")
lines(lowess(incomedata$age, incomedata$income, f=0.5)) Exercise 3

Exercise 4.
Let $Y = m(x) + \epsilon$
Then, in parametric regression, the functional form of is known or assumed to be known while in nonparametric regression, the function form of is assumed to be unknown.
Exercise 4

Exercise 5.

$$\begin{aligned}\hat{M}_h(x) &= \frac{\frac{1}{n}\sum_{i=1}^{n}K\left(\frac{x-x_i}{h}\right)Y_i}{\frac{1}{n}\sum_{i=1}^{n}K\left(\frac{x-x_i}{h}\right)}\\ as\ h\to\infty \quad & K\left(\frac{x-x_i}{h}\right)\to K(0)\\ \hat{M}_h(x) &= \frac{\frac{1}{n}K(0)\sum_{i=1}^{n}Y_i}{\frac{1}{n}\,n\,K(0)}\\ &= \frac{1}{n}\sum_{i=1}^{n}Y_i\end{aligned}$$

Exercise 5

◀◀ ▶▶
◀ ▶
◀ Doc Doc ▶
Back Close

Exercise 6.
Let $Y = m(x) + \epsilon$
Then, in fixed design model, the X variables are controlled and non-stochastic. Y is therefore homoscedastic.
In random design model, both the response and the regressors are random. Exercise 6

Exercise 7.
Boundary Kernels:
Kernel weight become asymmetric as x approaches the boundary points. However, the boundary effect is not present for x as in the interval of the observation interval.
The generalised jackniffing technique can be used to solve the boundary effects problem.
Exercise 7

Exercise 8.

neuraldata <- read.csv(C:
waititus data
BANKDATAEXAM.csv, header= TRUE)
our.x<- data.frame(age, amount, salary, time, gender)
our.y<- data.frame(status)
our.x<- as.matrix(our.x)
our.y<- as.matrix(our.y) neural.nettt<-nnet(our.x,our.y,size=14,entropy=TRUE,range=0.1, maxit=10000)
fitted.values(neural.nettt)
newloans<-read.csv(C:waititusdata
newapplicants.csv,header=TRUE)
PredictedStatus<- predict(neural.nettt, newdata=newloans) Exercise 8

Exercise 9.
The cross validation (leave-one-out method) is based on regression smoothens in which one, say the j^{th}, observation is left out, that is,

$$\hat{M}_{h,j}(x_j) = n \sum_{i \neq j} w_{hi}(x_i) Y_i$$

so that the cross-validation function is given by

$$\begin{aligned} CV(h) &= \frac{1}{n}\sum_{i=1}^{n}\left(Y_j - \hat{M}_{h,j}(x_j)\right) w(x_j) \\ so\, that & \\ \hat{h} &= argmin\, CV(h) \end{aligned}$$

Exercise 9

◀◀ ▶▶
◀ ▶
◀ Doc Doc ▶
Back Close