# Lecture 13

# Nonlinear Systems - Newton's Method

## An Example

The LORAN (LOng RAnge Navigation) system calculates the position of a boat at sea using signals from fixed transmitters. From the time differences of the incoming signals, the boat obtains differences of distances to the transmitters. This leads to two equations each representing hyperbolas defined by the differences of distance of two points (foci). An example of such equations[1] are

$$\begin{aligned} \frac{x^2}{186^2} - \frac{y^2}{300^2 - 186^2} &= 1 \quad \text{and} \\ \frac{(y-500)^2}{279^2} - \frac{(x-300)^2}{500^2 - 279^2} &= 1\,. \end{aligned} \tag{13.1}$$

Solving two quadratic equations with two unknowns, would require solving a 4 degree polynomial equation. We could do this by hand, but for a navigational system to work well, it must do the calculations automatically and numerically. We note that the Global Positioning System (GPS) works on similar principles and must do similar computations.

## Vector Notation

In general, we can usually find solutions to a system of equations when the number of unknowns matches the number of equations. Thus, we wish to find solutions to systems that have the form

$$\begin{aligned} f_1(x_1, x_2, x_3, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, x_3, \ldots, x_n) &= 0 \\ f_3(x_1, x_2, x_3, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, x_3, \ldots, x_n) &= 0. \end{aligned} \tag{13.2}$$

For convenience we can think of $(x_1, x_2, x_3, \ldots, x_n)$ as a vector $\mathbf{x}$ and $(f_1, f_2, \ldots, f_n)$ as a vector-valued function $\mathbf{f}$. With this notation, we can write the system of equations (13.2) simply as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0},$$

[1] E. Johnston, J. Mathews, *Calculus*, Addison-Wesley, 2001

i.e. we wish to find a vector that makes the vector function equal to the zero vector.

As in Newton's method for one variable, we need to start with an initial guess $\mathbf{x}_0$. In theory, the more variables one has, the harder it is to find a good initial guess. In practice, this must be overcome by using physically reasonable assumptions about the possible values of a solution, i.e. take advantage of engineering knowledge of the problem. Once $\mathbf{x}_0$ is chosen, let

$$\Delta\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_0.$$

## Linear Approximation for Vector Functions

In the single variable case, Newton's method was derived by considering the linear approximation of the function $f$ at the initial guess $\mathbf{x}_0$. From Calculus, the following is the linear approximation of $\mathbf{f}$ at $\mathbf{x}_0$, for vectors and vector-valued functions:

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_0) + D\mathbf{f}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0).$$

Here $D\mathbf{f}(\mathbf{x}_0)$ is an $n \times n$ matrix whose entries are the various partial derivative of the components of $\mathbf{f}$, evaluated at $\mathbf{x}_0$. Specifically,

$$D\mathbf{f}(\mathbf{x}_0) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}_0) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}_0) & \frac{\partial f_1}{\partial x_3}(\mathbf{x}_0) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}_0) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}_0) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}_0) & \frac{\partial f_2}{\partial x_3}(\mathbf{x}_0) & \cdots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}_0) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}_0) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}_0) & \frac{\partial f_n}{\partial x_3}(\mathbf{x}_0) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}_0) \end{pmatrix}. \tag{13.3}$$

## Newton's Method

We wish to find $\mathbf{x}$ that makes $\mathbf{f}$ equal to the zero vectors, so let's choose $\mathbf{x}_1$ so that

$$\mathbf{f}(\mathbf{x}_0) + D\mathbf{f}(\mathbf{x}_0)(\mathbf{x}_1 - \mathbf{x}_0) = \mathbf{0}.$$

Since $D\mathbf{f}(\mathbf{x}_0)$ is a square matrix, we can solve this equation by

$$\mathbf{x}_1 = \mathbf{x}_0 - (D\mathbf{f}(\mathbf{x}_0))^{-1}\mathbf{f}(\mathbf{x}_0),$$

provided that the inverse exists. The formula is the vector equivalent of the Newton's method formula we learned before. However, in practice we never use the inverse of a matrix for computations, so we cannot use this formula directly. Rather, we can do the following. First solve the equation

$$D\mathbf{f}(\mathbf{x}_0)\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}_0). \tag{13.4}$$

Since $D\mathbf{f}(\mathbf{x}_0)$ is a known matrix and $-\mathbf{f}(\mathbf{x}_0)$ is a known vector, this equation is just a system of linear equations, which can be solved efficiently and accurately. Once we have the solution vector $\Delta\mathbf{x}$, we can obtain our improved estimate $\mathbf{x}_1$ by

$$\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}.$$

For subsequent steps, we have the following process:

- Solve $D\mathbf{f}(\mathbf{x}_i)\Delta\mathbf{x} = -\mathbf{f}(\mathbf{x}_i)$ for $\Delta\mathbf{x}$.
- Let $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$

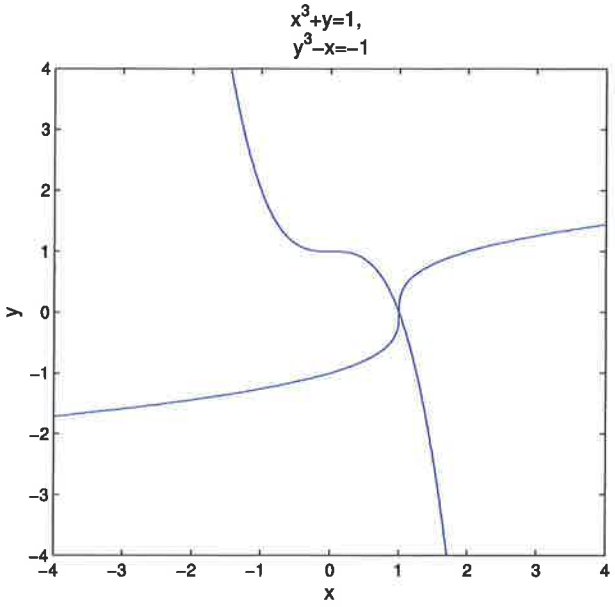



Figure 13.1: Graphs of the equations $x^3 + y = 1$ and $y^3 - x = -1$. There is one and only one intersection; at $(x, y) = (1, 0)$.

## An Experiment

We will solve the following set of equations:

$$\begin{aligned} x^3 + y &= 1 \\ y^3 - x &= -1. \end{aligned} \tag{13.5}$$

You can easily check that $(x, y) = (1, 0)$ is a solution of this system. By graphing both of the equations you can also see that $(1, 0)$ is the only solution (Figure 13.1).

We can put these equations into vector-function form (13.2) by letting $x_1 = x$, $x_2 = y$ and

$$\begin{aligned} f_1(x_1, x_2) &= x_1^3 + x_2 - 1 \\ f_2(x_1, x_2) &= x_2^3 - x_1 + 1. \end{aligned}$$

or

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1^3 + x_2 - 1 \\ x_2^3 - x_1 + 1 \end{pmatrix}.$$

Now that we have the equation in vector-function form, write the following script program:

```
format long;
f = @(x)[ x(1)^3+x(2)-1 ; x(2)^3-x(1)+1 ]
x = [.5;.5]
x = fsolve(f,x)
```

Save this program as **myfsolve.m** and run it. You will see that the internal MATLAB solving command **fsolve** approximates the solution, but only to about 7 decimal places. While that would be close enough for most applications, one would expect that we could do better on such a simple problem.

Next we will implement Newton's method for this problem. Modify your **myfsolve** program to:

```
% mymultnewton
format long;
n=8  % set some number of iterations, may need adjusting
f = @(x)[x(1)^3+x(2)-1 ; x(2)^3-x(1)+1] % the vector function
% the matrix of partial derivatives
Df = @(x)[3*x(1)^2,  1 ; -1,  3*x(2)^2]
x = [.5;.5]              % starting guess
for i = 1:n
      Dx = -Df(x)\f(x); % solve for increment
      x = x + Dx        % add on to get new guess
      f(x)              % see if f(x) is really zero
end
```

Save and run this program (as **mymultnewton**) and you will see that it finds the root exactly (to machine precision) in only 6 iterations. Why is this simple program able to do better than MATLAB's built-in program?

## Exercises

13.1 (a) Put the LORAN equations (13.1) into the function form (13.2).

(b) Construct the matrix of partial derivatives $D\mathbf{f}$ in (13.3).

(c) Adapt the **mymultnewton** program to find a solution for these equations. By trying different starting vectors, find at least three different solutions. (There are actually four solutions.) Think of at least one way that the navigational system could determine which solution is correct.

Computational Optimization and Applications, 17, 329–341, 2000


# Accurate Solution to Overdetermined Linear Equations with Errors Using $L_1$ Norm Minimization

J. BEN ROSEN jbrosen@cs.ucsd.edu
*Computer Science Department, University of Minnesota, Minneapolis, MN 55455, USA; Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093*

HAESUN PARK hpark@cs.umn.edu
*Computer Science Department, University of Minnesota, Minneapolis, MN 55455, USA*

JOHN GLICK glick@acusd.edu
*Department of Mathematics and Computer Science, University of San Diego, San Diego, CA 92110, USA*

LEI ZHANG lzhang@cs.umn.edu
*Computer Science Department, University of Minnesota, Minneapolis, MN 55455, USA*

*Received April 6, 1999; Accepted July 6, 1999*

**Abstract.** It has been known for many years that a robust solution to an overdetermined system of linear equations $Ax \approx b$ is obtained by minimizing the $L1$ norm of the residual error. A correct solution $x$ to the linear system can often be obtained in this way, in spite of large errors (outliers) in some elements of the $(m \times n)$ matrix $A$ and the data vector $b$. This is in contrast to a least squares solution, where even one large error will typically cause a large error in $x$. In this paper we give necessary and sufficient conditions that the correct solution is obtained when there are some errors in $A$ and $b$. Based on the sufficient condition, it is shown that if $k$ rows of $[A\ b]$ contain large errors, the correct solution is guaranteed if $(m-n)/n \geq 2k/\sigma$, where $\sigma > 0$, is a lower bound of singular values related to $A$. Since $m$ typically represents the number of measurements, this inequality shows how many data points are needed to guarantee a correct solution in the presence of large errors in some of the data. This inequality is, in fact, an upper bound, and computational results are presented, which show that the correct solution will be obtained, with high probability, for much smaller values of $m-n$.

**Keywords:** overdetermined linear systems, robust approximation, minimum error, zero error conditions, outliers, 1-norm, linear programming

## 1. Introduction

The benefits of using minimization of the $L_1$ norm in approximation problems have been known for many years [3–5, 10, 16]. The advantages are likely to be greatest when the problem data contains a relatively small number of large errors (outliers). Examples of applications where the $L_1$ norm minimization is used to advantage are described in [8, 9, 13, 14, 20]

We present an analysis for the linear, overdetermined system of equations

$$Ax \approx b, \tag{1}$$

It is also shown there that the condition is invariant with respect to the magnitude of the errors in $b$. This property of the $L_1$ norm minimization is compared to the approximation $\tilde{x}$ obtained from the solution to the least squares problem

$$\min_x \|Ax - b\|_2, \tag{6}$$

where the error in $\tilde{x}$ will typically be proportional to the errors in $b$.

In order to obtain more practically useful relationships, extensive computational tests were carried out. The results of these tests give an empirical formula and curves which show how many data points ($m$) are needed to give $x^* = x_c$ with any specified probability $P$, assuming that there are no more than $k$ rows with large errors. These results are presented in §3. The computational results show that for $k$ rows with errors, $k \leq 50$ and $n \leq 40$, (4) will give $x^* = x_c$ with probability $P \geq 0.995$ when $m - n \geq 22 + 2k$. Note that this value of $m - n$ depends linearly on $k$, as does the upper bound in Theorem 2.1, but it is substantially smaller, and therefore more useful in practice.

Given an optimal solution to (4), and the corresponding basis $B$, it is also of interest to determine what changes can be made in $b$ without changing the optimality of the basis $B$. This situation can be analyzed by a sensitivity analysis of the linear program corresponding to (4). This linear program is given by (31) with $w = x$ and $d = b$. Such an analysis is described in [10].

The use of the $L_1$ norm minimization has also been investigated for the more difficult parameter estimation problem

$$A(\alpha)x \approx b. \tag{7}$$

For this type of problem it is necessary to determine the parameter vector $\alpha$ as well as the vector $x$ when there are errors in the data vector $b$. Many signal identification problems can be formulated in this way [1, 11, 12, 18, 19, 21]. The minimization problem now becomes

$$\min_{\alpha,x} \|A(\alpha)x - b\|_1. \tag{8}$$

The theoretical results presented here for (4) are relevant for (8), but are not directly applicable. An iterative algorithm for the solution of (8) is described and analyzed in [18]. It has been applied to the identification of a complex signal with some large errors in the data. The computational results presented in [19] show that, as with the linear case (4), the solution obtained using the $L_1$ norm is very robust with respect to large data errors.

## 2. Conditions for correct solution

### 2.1. *Necessary and sufficient conditions*

In this section we investigate when the $L_1$ minimization (4) will give the correct solution $x_c$ when there may be large errors in some rows of $A$ and in some elements of $b$. A condition is formulated in terms of a scalar $\beta$, and a vector $d$. It is then shown that $\|d\|_1 \leq \beta$ is a

then we always get $w \neq 0$. This will always occur if more than $m - n$ elements of $d$ are nonzero.

Directly from (15), the minimization (13) is equivalent to

$$\min_B \|\hat{A}B^{-1}d_B - d_{NB}\|_1. \tag{16}$$

Let $\mathbf{S}$ denote the set of basis matrices $B$, such that $d_B \neq 0$, and define

$$\beta = \min_{B \in \mathbf{S}} \|\hat{A}B^{-1}d_B - d_{NB}\|_1. \tag{17}$$

If $\|d\|_1 < \beta$, then the minimum for (16) will be attained with $d_B = 0$. If $\|d\|_1 = \beta$, then we may have $d_B \neq 0$, while for $\|d\|_1 > \beta$, we will always have $d_B \neq 0$. This is summarized by the following conditions for $x^* = x_c$:

NC: A necessary condition that $w = 0$ and $x^* = x_c$ is that $\|d\|_1 \leq \beta$.
SC: A sufficient condition that $w = 0$ and $x^* = x_c$ is that $\|d\|_1 < \beta$.

We now show that, for any fixed number $k$ of errors, the number of data points $m$ can be chosen so as to guarantee that the correct solution $x_c$ is computed. To show this, we choose $m = (q + 1)n$, where $q$ is a positive integer. The matrix $A$ is then partitioned so that

$$A^T = \begin{pmatrix} B^T & A_1^T & A_2^T, \ldots, A_q^T \end{pmatrix}. \tag{18}$$

Also let

$$\begin{aligned} \hat{A}^T &= \begin{pmatrix} A_1^T & A_2^T, \ldots, A_q^T \end{pmatrix}, \\ C^T &= \begin{pmatrix} C_1^T & C_2^T, \ldots, C_q^T \end{pmatrix}, \end{aligned} \tag{19}$$

where $C_j = A_j B^{-1}$, $j = 1, \ldots, q$, are $(n \times n)$ matrices. Let $\sigma_j$ be the minimum singular value of $C_j$, when $B$ is determined by (17), and assume that $\sigma_j \geq \sigma > 0$, $j = 1, \ldots, q$. Note that if $A_j = B$, then $\sigma_j = 1$. Let $k$ elements of $d$ be nonzero, and for simplicity assume they have the values $\pm 1$.

**Theorem 2.1.** *A sufficient condition that $x^* = x_c$ is that*

$$\frac{m - n}{n} \geq \frac{2k}{\sigma}. \tag{20}$$

**Proof:** We have $\|d\|_1 = k$, and for $d_B \neq 0$, $\|d_B\|_2 \geq 1$, and $\|d_{NB}\|_1 \leq k - 1$. We consider $d \neq 0$, since otherwise $x^* = x_c$. From (17),

$$\begin{aligned} \beta &= \min_{B \in \mathbf{S}} \|Cd_B - d_{NB}\|_1 \\ &\geq \min_{B \in \mathbf{S}} \|Cd_B\|_1 - \max_{B \in \mathbf{S}} \|d_{NB}\|_1 \\ &\geq \sum_{j=1}^{q} \|C_j d_B\|_1 - k + 1. \end{aligned} \tag{21}$$

Let $\mathbf{B}$ denote the set of nonsingular $(n \times n)$ basis matrices $B$ selected from the rows of $A$. Corresponding to each $B \in \mathbf{B}$ are the partitions:

$$A = \begin{pmatrix} B \\ \hat{A} \end{pmatrix}, \qquad d = \begin{pmatrix} d_B \\ d_{NB} \end{pmatrix}, \quad \text{and} \quad \epsilon = \begin{pmatrix} \epsilon_B \\ \epsilon_{NB} \end{pmatrix}.$$

The solution to (24) or (25) will give a basis $B_M \in \mathbf{B}$ and corresponding $\hat{A}_M$, $d_B$ and $\epsilon_B$:

$$\begin{aligned} \min_w \|Aw - d - \epsilon\|_1 &= \min_{B \in \mathbf{B}} \|\hat{A}B^{-1}(d_B + \epsilon_B) - d_{NB} - \epsilon_{NB}\|_1 \\ &= \left\|\hat{A}_M B_M^{-1}(d_B + \epsilon_B) - d_{NB} - \epsilon_{NB}\right\|_1, \end{aligned} \tag{26}$$

with $w^* = B^{-1}(d_B + \epsilon_B)$. Also let $\mathbf{S} \subset \mathbf{B}$ denote the subset of basis matrices for which $d_B \neq 0$. Then the related minimization

$$\min_{B \in \mathbf{S}} \|\hat{A}B^{-1}(d_B + \epsilon_B) - d_{NB} - \epsilon_{NB}\|_1 = \left\|\hat{A}_S B_S^{-1}(d_B + \epsilon_B) - d_{NB} - \epsilon_{NB}\right\|_1 \tag{27}$$

gives the matrix $B_S^{-1}$ and corresponding $\hat{A}_S$. Now define the two quantities

$$\rho_M = \left\|\hat{A}_M B_M^{-1}\right\|_1 \quad \text{and} \quad \rho_S = \left\|\hat{A}_S B_S^{-1}\right\|_1.$$

Note that although the $\|\cdot\|_1$ given by (27) is greater than or equal to that given by (26), we may have $\rho_S < \rho_M$.

We want a sufficient condition that (26) gives $d_B = 0$, so that

$$w^* = B_M^{-1}\epsilon_B$$

and

$$\|w^*\|_1 \leq \left\|B_M^{-1}\right\|_1 n\delta. \tag{28}$$

This will be true if the minimum of (26) is obtained with $B \notin \mathbf{S}$, so that from (27) we want

$$\left\|\hat{A}_M B_M^{-1}\epsilon_B - d_{NB} - \epsilon_{NB}\right\|_1 < \left\|\hat{A}_S B_S^{-1}(d_B + \epsilon_B) - d_{NB} - \epsilon_{NB}\right\|_1 \tag{29}$$

The left side of (29) is overestimated by

$$\|d_{NB}\|_1 + \left\|\hat{A}_M B_M^{-1}\epsilon_B\right\|_1 + \|\epsilon_{NB}\|_1 \leq \|d\|_1 + \rho_M n\delta + (m - n)\delta.$$

The right side of (29) is underestimated by

$$\min_{B \in \mathbf{S}} \|\hat{A}B^{-1}d_B - d_{NB}\|_1 - \rho_S n\delta - (m - n)\delta,$$

with $m + n$ variables and $2m$ inequality constraints. The solution is given by $w = B^{-1}d_B$, with

$$\sum_{i=1}^{m} \gamma_i = \|Aw - d\|_1, \qquad \text{and} \quad \gamma_i \geq 0. \tag{32}$$

The desired result, $(x^* = x_c)$, therefore corresponds to $w = 0$.

For each fixed value of $(m, n, k)$ a total of 50 LPs were solved, each with a different matrix $A$. The probability $P$ that $w = 0$, corresponding to $(m, n, k)$, was then taken to be

$$P = P(m, n, k) = l/50, \tag{33}$$

where $l$ is the number of LPs with the solution $w = 0$.

For each fixed value of $n$ and $k$, 7 values of $m$ were chosen so that the corresponding 7 values of $l$ covered the range [0, 50]. For each of the values $n = 10, 20$, a total of 8 values of $k \in [5, 40]$ were used, and for $n = 40$, a total of 11 values of $k \in [1, 40]$ were used. This gave a total of 9450 LPs solved. The largest of these $(m = 140, n = 40)$ required approximately 10 seconds to solve on a Sun Sparcstation 4. The results obtained are summarized in figures 1–3. The curves presented in figures 1–3 show how the probability $P$

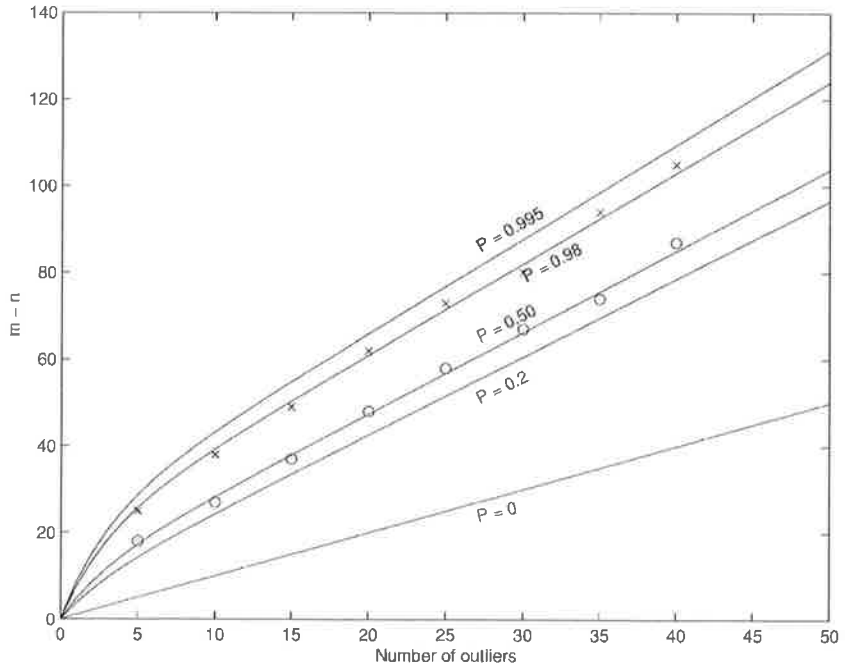



*Figure 1.* Probability of zero outliers in the basis for $n = 40$.

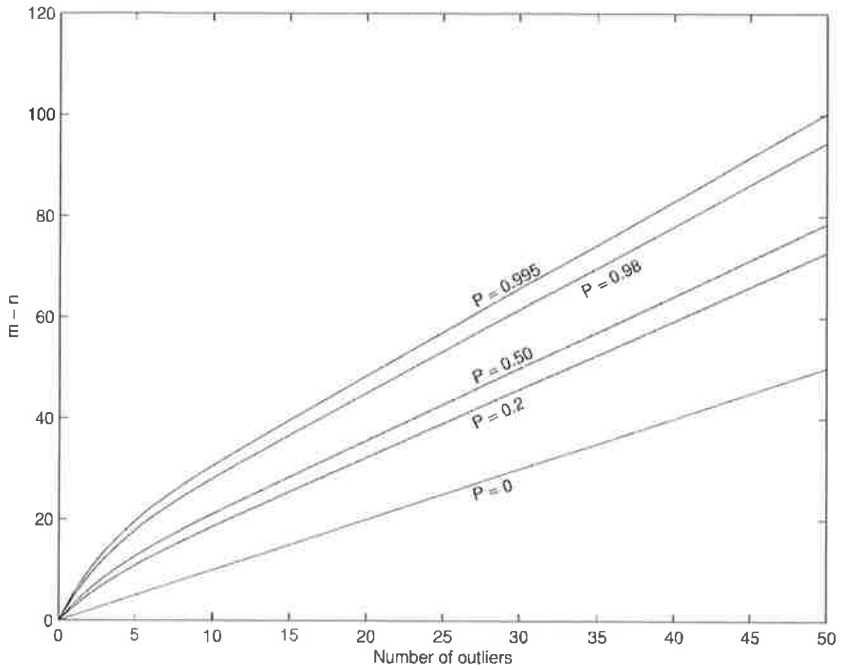



*Figure 3.* Probability of zero outliers in the basis for $n = 10$.

the computational results. Specifically, for any fixed $P$, $m - n$ is essentially linear in $k$, for $\gamma_3 k \geq 3$ or $k \geq 10$. Also, $m - n = 0$ for $k = 0$. Finally, $k \geq 5$ and $m - n \geq 22 + 2k$, gives $P \geq 0.995$.

The parameters $\gamma_1$ and $\gamma_2$ were determined by setting $P = 0.5$ and fitting the resulting linear function $m - n = \gamma_1 + \gamma_2 k$ to the data for $P = 0.5$ and $k \geq 0$. The values of $\Delta\gamma_1$ and $\Delta\gamma_2$ were then determined to give a best linear fit to the data for $P = 0.98$ and $k \geq 10$. Note that $\ln(\frac{P}{1-P}) \approx 3.89$ for $P = 0.98$, so that for $P = 0.98$ and $k \geq 10$ we have

$$m - n \approx \gamma_1 + \Delta\gamma_1 + (\gamma_2 + \Delta\gamma_2)k. \tag{35}$$

*Table 1.* Parameters for Eq. (34).

| | $n = 10$ | $n = 20$ | $n = 40$ |
|---|---|---|---|
| $\gamma_1$ | 7.0 | 8.0 | 9.8 |
| $\gamma_2$ | 1.43 | 1.61 | 1.88 |
| $\gamma_3$ | 0.32 | 0.32 | 0.32 |
| $\Delta\gamma_1$ | 5.0 | 6.2 | 9.2 |
| $\Delta\gamma_2$ | 0.22 | 0.22 | 0.22 |

6. A. Bjorck, Numerical Methods for Least Squares Problems, SIAM: Philadelphia, PA, 1996.
7. R. De Beer and D. Van Ormondt, "Analysis of NMR data using time-domain fitting procedures," in In-vivo Magnetic Resonance Spectroscopy I: Probeheads, Radiofrequency Pulses, Spectrum Analysis, NMR Basic Principles and Progress 26, M. Rudin (Ed.), Springer-Verlag: Berlin, 1992, pp. 201–248.
8. T.E. Dielman, "Least absolute value estimation in regression models: An annotated bibliography," Commun. Statistical-Theor. Meth., vol. 13, pp. 513–541, 1984.
9. T.E. Dielman and E.L. Rose, "Forecasting in least absolute value regression with autocorrelated errors: A small sample study," Int. J. Forecast, vol. 10, pp. 539–547, 1994.
10. J. Dupacova, "Robustness of $L_1$ regression in the light of linear programming," in $L_1$-statistical Analysis and Related Methods, Y. Dodge (Ed.), North-Holland: Amsterdam, 1992.
11. R. Kumaresan and D.W. Tufts, "Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise," IEEE Trans. on Acoust., Speech, and Signal Process., vol. 30, pp. 833–840, 1982.
12. L. Ljung, System Identification Theory for the User, Prentice-Hall: Englewood Cliffs, NJ, 1987.
13. P.W. Mielke, Jr and K.J. Berry, "Permutation-based multivariate regression analysis: The case for least sum of absolute deviations regression," Ann. Oper. Res., vol. 74, pp. 259–268, 1997.
14. S.C. Narula and J.F. Wellington, "The minimum sum of absolute errors regression: A state of the art survey," Int. Statist. Rev., vol. 50, pp. 317–326, 1982.
15. M.A. Rahman and K.B. Yu, "Total least squares approach for frequency estimation using linear prediction," IEEE Trans. Acous. Speech Signal Process., vol. 35, pp. 1440–1454, 1987.
16. J.R. Rice and J.S. White, "Norms for smoothing and estimation," SIAM Rev., vol. 6, pp. 243–256, 1964.
17. J.B. Rosen, H. Park, and J. Glick, "Total least norm formulation and solution for structured problems," SIAM J. Matrix Anal. Appl., vol. 17, pp. 110–128, 1996.
18. J.B. Rosen, H. Park, and J. Glick, "Structured total least norm for nonlinear problems," SIAM J. Matrix Anal. Appl., vol. 20, pp. 14–30, 1999.
19. J.B. Rosen, H. Park, and J. Glick, "Signal identification using a least $L_1$ norm algorithm," Optimization & Engineering, vol. 1, pp. 51–65, 2000.
20. P.J. Rousseeuw and A.M. Leroy, Robust Regression and Outlier Detection, John Wiley: New York, 1987.
21. S. Van Huffel, H. Park, and J.B. Rosen, "Formulation and solution of structured total least norm problems for parameter estimation," IEEE Trans. Signal Process., vol. 44, pp. 2464–2474, 1996.
22. S. Van Huffel and J. Vandewalle, The Total Least Squares Problem, Computational Aspects and Analysis, SIAM: Philadelphia, PA, 1991.

# Regression III: Advanced Methods

Bill Jacoby
*Michigan State University*

http://polisci.msu.edu/jacoby/icpsr/regress3

# Multiple Nonparametric Regression

- Formally, local polynomial regression is easily extended to multiple regression
- In practice, however, the ***Curse of Dimensionality*** gets in the way
- As the number of predictors increases, the number of points in a local neighbourhood for a particular focal point declines
  - In order to include a fixed number of cases in each local regression the span necessarily gets larger, and thus the regression becomes less local—as a result the bias of the estimate increases
- Moreover, because nonparametric regression has no parameter estimates, the model becomes difficult to interpret the more explanatory variables we include
  - Graphing the fitted values helps, but with more than two predictors we can't see the complete regression surface
- Despite these limitations, multiple regression with two explanatory variables can be useful

# Multiple Nonparametric Regression (2)

- Defining neighbourhoods for local polynomial multiple regression is a straightforward extension of the simple lowess case
- Again we use ***Weighted Least Squares***, regressing Y on the X's, emphasizing cases closest to the focal point
- For the local linear fit, the equation for a particular case is

$$Y_i = A + B_1(x_{i1} - x_{01}) + B_2(x_{i2} - x_{02}) + \cdots + B_k(x_{ik} - x_{ok}) + E_i$$

- As in local polynomial simple regression we choose a span $s$ that smooths the data
- Now, however, we must take the product of all the marginal weights in order to find the appropriate neighbourhood for each observations

$$w_i = W\left(\frac{D(\mathbf{x}_i, \mathbf{x}_0)}{h}\right)$$

- where W is the weight function, $D(x_i, x_0)$ is the distance in predictor space between the focal point $x_0$ and each $x_i$ and h is the window half-width
- The distance $D(x_i, x_0)$ can be calculated in several ways:
  - If the X's are standardized, we can use ***Euclidean distances*** (the method used in ***R)***:

$$D_E(\mathbf{x}_i, \mathbf{x}_0) = \sqrt{\sum_{j=1}^{k} (x_{ij} - x_{oj})^2}$$

- Alternatively we could use ***Generalized distances*:**

$$D_g(\mathbf{x}_i, \mathbf{x}_0) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_0)' \mathbf{V}^{-1} (\mathbf{x}_1 - \mathbf{x}_0)}$$

- Here V is the covariance matrix of the X's

# Local Polynomial Multiple Regression: Example: Inequality data

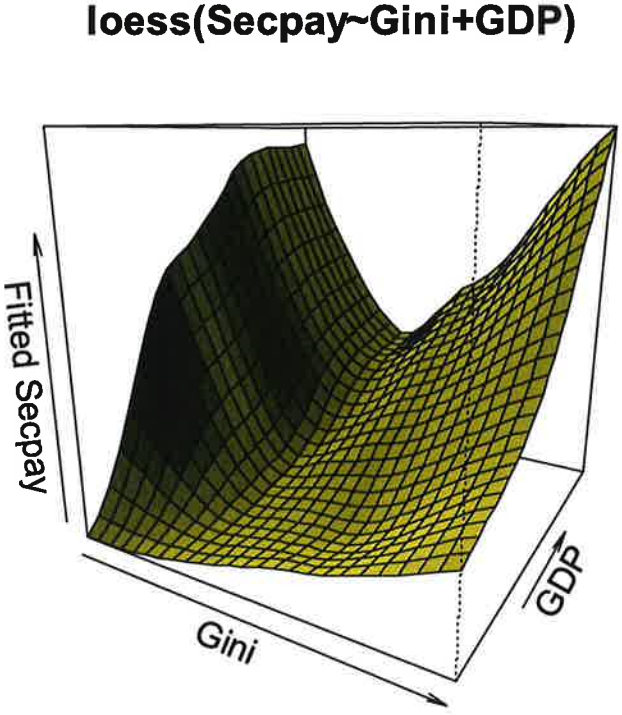
- Using the now familiar Inequality data we regress ***secpay*** on ***gini*** and ***gdp***, fitting the following model

```
library(modreg)
mod.loess<-loess(secpay~gini+gdp,
                 span=.8, degree=1)
```

- Since there are no parameter estimates, it is important to graph the regression surface
- The general tests used in the simple local polynomial case can also be used here, and can be extended to the incremental F-test for terms in the model
  - That is,we can test the impact of a term by comparing a model with it to a model without it

# Local Polynomial Multiple Regression: Example: Inequality data (2)

- The regression surface is clearly nonlinear, especially for ***gini***, but it is less obvious for ***GDP*** (in fact, for mid-values of ***gini*** the ***GDP*** plane appears flat)
- The next step is a type II incremental F-test for the terms in the model
- We could also test for nonlinearity using an F-test, comparing the RSS of this model with the RSS of the linear model



# Local Polynomial Multiple Regression: R-script for perspective plot

```
mod.loess<-loess(secpay~gini+gdp, span=.8, degree=1)
#Setting up "newdata" points for fitted values
#in order to make perspective plots
gini2<-seq(min(gini), max(gini), len=25)
gdp2<-seq(min(gdp),max(gdp), len=25)
data<-expand.grid(gini=gini2, gdp=gdp2)
#Fitted values for each model for the newdata
fit.mod<-matrix(predict(mod.loess, data), 25,25)
persp (gini2, gdp2, fit.mod,
theta=30, xlab='Gini', ylab='GDP', shade=.75, col="yellow",
zlab='Fitted Secpay',
main="loess(Secpay~Gini+GDP)")
```

# Statistical Significance Test for Terms

- The `anova` function allows comparison of two models fit with the `loess` function
- As we see below, there is no statistically significant difference between the model with GDP and the model without it

```
> anova(mod.loess, mod.loess2)
Model 1: loess(formula = secpay ~ gini + gdp,
               span = 0.8, degree = 1)
Model 2: loess(formula = secpay ~ gini,
               span = 0.8, degree = 1)

Analysis of Variance:   denominator df 19.08

     ENP    RSS F-value Pr(>F)
1 5.0400 0.2529
2 3.1400 0.2745  0.4016 0.7634
```

# Dynamic 3D perspective plot in R

```
#Multiple nonparametric regression
Model.lowess<-loess(secpay~gini+gdp, span=.8, degree=1)
gini2<-seq(min(gini), max(gini), len=35)
gdp2<-seq(min(gdp), max(gdp), len=35)
#dividing the range of gini and gdp
#into 25 values each
data<-expand.grid(gini=gini2, gdp=gdp2)
secpay.fitted<-matrix(predict(Model.lowess, data), 35, 35)
persp(gini2, gdp2, secpay.fitted, theta=35,
      ph=20, ticktype='detailed',
      xlab='Gini Coefficient', ylab='GDP', zlab='Attitudes')

#Dynamic 3D plots
library(djmrgl) #Requires R--SDI
persp3d(gini2, gdp2, secpay.fitted, main="Lowess", cex=1.5)
```

# Summary and Conclusions

- ***Nonparametric regression***, in particular lowess smoothing, provides a very useful tool for assessing linearity
  - All the same tests for OLS apply
  - We can see the linear model as nested within the nonparametric model and hence test for nonlinearity using an incremental F-test
- For complicated nonlinear relationships it can also provide a good summary of the data in graphical form
- The extension to multiple nonparametric regression is relatively straightforward, but it has limitations
  - Since there are no parameters it is difficult to comprehend relationships in more than three dimensions
  - The ***curse of dimensionality*** means that we need a larger span and hence our estimates are more biased

# Summary and Conclusions (2)

- ***Additive models***, however, attempt over come these problems by putting some restrictions on the model
- Wednesday's and Thursday's lecture will cover these models, and the more general ***Generalized Additive Models***

# *Local Polynomial Regression*

*John Hughes*

*October 2, 2013*

Recall that the nonparametric regression model is

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

where $f$ is the regression function and the $\varepsilon_i$ are errors such that $\mathbb{E}\varepsilon_i = 0$.

## *The Nadaraya-Watson Kernel Estimator*

The Nadaraya-Watson kernel estimator offers what is probably the simplest approach to nonparametric regression. The kernel estimator is an example of a linear smoother. The estimator is linear in the sense that it is given by a linear transformation of the response. Specifically, let $s(x) = (s_1(x), \ldots, s_n(x))'$, where

$$s_i(x) = \frac{w_i(x)}{\sum_{j=1}^n w_j(x)},$$

where $w_i(x) = K\{(x - x_i)/h\}$. Now, if $\mathbf{Y} = (Y_1, \ldots, Y_n)'$, the kernel estimator of $f(x)$ is

$$\begin{aligned}
\hat{f}_n(x) &= \mathbf{s}'(x)\mathbf{Y} \\
&= \sum_{i=1}^n s_i(x) Y_i \\
&= \sum_{i=1}^n \frac{w_i(x)}{\sum_{j=1}^n w_j(x)} Y_i \\
&= \sum_{i=1}^n \frac{K\{(x - x_i)/h\}}{\sum_{j=1}^n K\{(x - x_j)/h\}} Y_i.
\end{aligned}$$

This shows that $\hat{f}_n(x)$ is a weighted average of the observations, where the weights $s(x)$ are normalized kernel weights.

This formulation can easily be extended to handle a grid of estimation points $\mathbf{z} = (z_1, \ldots, z_{n_g})'$. Form the $n_g \times n$ matrix $\mathbf{S}$, the $k$th row of which is $\mathbf{s}'(z_k)$. Then

$$\hat{\mathbf{f}}_n(\mathbf{z}) = (\hat{f}_n(z_1), \ldots, \hat{f}_n(z_{n_g}))' = \mathbf{S}\mathbf{Y}.$$

The matrix $\mathbf{S}$ is called the smoothing matrix. It is analogous to the hat matrix from linear regression.

**Theorem 1** *The risk (assuming the $L^2$ loss) of the Nadaraya-Watson kernel estimator is*

$$R(\hat{f}_n, f) = \frac{h^4}{4}\left\{\int x^2 K(x)dx\right\}^2 \int \left\{\ddot{f}(x) + 2\dot{f}(x)\frac{\dot{g}(x)}{g(x)}\right\}^2 dx \quad (1)$$
$$+ \frac{\sigma^2 \int K^2(x)dx}{nh}\int \frac{1}{g(x)}dx + o(nh^{-1}) + o(h^4)$$

*as $h \to 0$ and $nh \to \infty$, where $g$ is the density from which the $x_i$ are drawn, and $\sigma^2 = \mathbb{V}\varepsilon_i$.*

If we set the derivative of (1) equal to zero and solve for $h$, we get the optimal bandwidth

$$h_{\text{opt}} = n^{-1/5}\left[\frac{\sigma^2 \int K^2(x)dx \int \frac{1}{g(x)}dx}{\left\{\int x^2 K(x)dx\right\}^2 \int \left\{\ddot{f}(x) + 2\dot{f}(x)\frac{\dot{g}(x)}{g(x)}\right\}^2 dx}\right]^{1/5},$$

which implies that $h_{\text{opt}} = O(n^{-1/5})$. If we plug $h_{\text{opt}}$ into (1), we see that the risk decreases at the rate $O(n^{-4/5})$. For most parametric models, the risk of the MLE decreases at the rate $O(n^{-1})$. The moral of this story is that we pay a price for using a nonparametric approach. We gain flexibility, but we may sacrifice statistical power to get it.

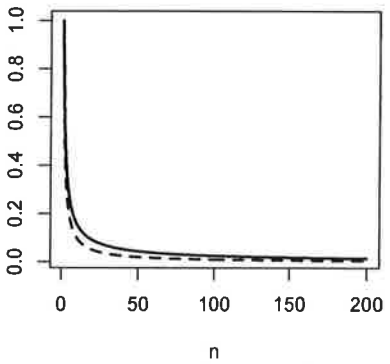



Figure 1: This figure illustrates the price we pay for adopting a nonparametric approach, as measured by the rate at which risk decreases with increasing sample size. The solid curve is $n^{-4/5}$. The dashed curve is $n^{-1}$.

## *Local Polynomial Regression*

A kernel estimator suffers from design bias (a bias that depends on the distribution of the $x_i$) and boundary bias (a bias near the endpoints of the $x_i$). These biases can be reduced by using local polynomial regression.

Consider choosing an estimator that minimizes $\sum_{i=1}^n (Y_i - \beta_0)^2$. Note that this is equivalent to minimizing the squared length of $\boldsymbol{Y} - \beta_0\mathbf{1}$, where 'length' is defined as the ordinary Euclidean norm

$$\|v\| = \sqrt{\sum_{i=1}^n v_i^2},$$

which is in turn defined in terms of the usual inner product, the dot product:

$$\langle u, v\rangle = u'v = \sum_{i=1}^n u_i v_i.$$

That is, $\|v\|^2 = \langle v, v\rangle = v'v$.

Recall that the solution to this estimation problem is $\hat{\beta}_0 = \bar{Y}$. The vector $\bar{Y}\mathbf{1}$ is the vector in span$\{\mathbf{1}\}$ that is closest to $\boldsymbol{Y}$ with respect to the ordinary norm. You may also recall that $\bar{Y}\mathbf{1}$ is the orthogonal

projection of $\mathbf{Y}$ onto span$\{\mathbf{1}\}$, where our notion of perpendicularity is given by the dot product: $\boldsymbol{u} \perp \boldsymbol{v}$ iff $\boldsymbol{u}'\boldsymbol{v} = 0$. To see this, observe that the orthogonal projection of $\mathbf{Y}$ onto span$\{\mathbf{1}\}$ is

$$\mathbf{1}(\mathbf{1}'\mathbf{1})^{-1}\mathbf{1}'\mathbf{Y}.$$

(This is just a special case of $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ from ordinary linear regression.) Now, $\mathbf{1}'\mathbf{1} = \sum_{i=1}^{n} 1 = n$, and $\mathbf{1}'\mathbf{Y} = \sum_{i=1}^{n} Y_i$. Thus

$$\mathbf{1}(\mathbf{1}'\mathbf{1})^{-1}\mathbf{1}'\mathbf{Y} = \mathbf{1}\frac{1}{n}\sum_{i=1}^{n} Y_i = \bar{Y}\mathbf{1}.$$

Now change the scenario slightly by changing the inner product from $\boldsymbol{u}'\boldsymbol{v}$ to

$$\boldsymbol{u}'\mathbf{W}_x\boldsymbol{v},$$

where $\mathbf{W}_x = \text{diag}\{w_i(x)\}$, with $w_i(x) = K\{(x - x_i)/h\}$. The analogous estimation problem is to minimize $\sum_{i=1}^{n} w_i(x)(Y_i - \beta_0)^2$, but now the relevant projection is orthogonal with respect to this new inner product. Hence,

$$\hat{\beta}_0 = (\mathbf{1}'\mathbf{W}_x\mathbf{1})^{-1}\mathbf{1}'\mathbf{W}_x\mathbf{Y}.$$

This implies that

$$\hat{f}_n(x) = \hat{\beta}_0 = \frac{\sum_{i=1}^{n} w_i(x)Y_i}{\sum_{i=1}^{n} w_i(x)},$$

the kernel estimator. And so we see that the kernel estimator results from introducing kernel weights in an intercept-only linear model. The weights 'localize' the estimator in the sense that more distant observations are down-weighted. Since the kernel estimator is local and uses only an intercept, the kernel estimator is sometimes called a locally constant estimator.

Local polynomial regression is based on the idea that we might improve the estimator by using a higher-order polynomial as a local approximation to $f$. Taylor's theorem tells us this is a sensible idea. According to Taylor's theorem,

$$\begin{aligned} f(x) &\approx f(z) + f^{(1)}(z)(z - x) + \frac{f^{(2)}(z)}{2!}(z - x)^2 + \cdots + \frac{f^{(p)}(z)}{p!}(z - x)^p \\ &= \beta_0 + \beta_1(z - x) + \beta_2(z - x)^2 + \cdots + \beta_p(z - x)^p \\ &\equiv P_x(z, \boldsymbol{\beta}) \end{aligned}$$

for $z$ in a neighborhood of $x$, where $f^{(m)}$ denotes the $m$th derivative of $f$. The kernel estimator takes $p = 0$. More generally, local polynomial regression of order $p$ minimizes

$$\sum_{i=1}^{n} w_i(x)\{Y_i - P_x(x_i, \boldsymbol{\beta})\}^2. \tag{2}$$

This yields the local estimate

$$\hat{f}_n(x) = P_x(x, \hat{\beta}) = \hat{\beta}_0(x).$$

Note that the minimizer of (2) is

$$\hat{\beta}(x) = (\mathbf{X}_x' \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x' \mathbf{W}_x Y,$$

where

$$\mathbf{X}_x = \begin{pmatrix} 1 & x - x_1 & (x_1 - x)^2 & \cdots & \frac{(x_1 - x)^p}{p!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x & (x_n - x)^2 & \cdots & \frac{(x_n - x)^p}{p!} \end{pmatrix}.$$

This implies that $\hat{f}_n(x) = \hat{\beta}_0(x)$ is the inner product of $Y$ with the first row of $(\mathbf{X}_x' \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x' \mathbf{W}_x$, and so $\hat{f}_n(x)$ is a linear smoother. The estimator has mean and variance

$$\mathbb{E}\hat{f}_n(x) = s(x)\mathbf{f}(x)$$
$$\mathbb{V}\hat{f}_n(x) = \sigma^2 \|s(x)\|^2,$$

where $s(x)$ is the first row of $(\mathbf{X}_x' \mathbf{W}_x \mathbf{X}_x)^{-1} \mathbf{X}_x' \mathbf{W}_x$ and $\mathbf{f}(x) = (f(x_1), \ldots, f(x_n))'$.

## *Why p Should Be Odd*

The case $p = 1$ is called local linear regression. Local linear regression eliminates design bias and alleviates boundary bias.

**Theorem 2** *Let $Y_i = f(X_i) + \sigma(X_i)\varepsilon_i$ for $i \in \{1, \ldots, n\}$ and $X_i \in [a, b]$. Assume that the $X_i$ were drawn from density $g$. Suppose that $g$ is positive; $g$, $\ddot{f}$, and $\sigma$ are continuous in a neighborhood of $x$; and $h \to 0$ and $nh \to \infty$. Let $x \in (a, b)$. Then the local constant estimator and the local linear estimator both have variance*

$$\frac{\sigma^2(x)}{g(x)nh} \int K^2(u)du + o\left(\frac{1}{nh}\right).$$

*The local constant estimator has bias*

$$h^2 \left(\frac{1}{2}\ddot{f}(x) + \frac{\dot{f}(x)\dot{g}(x)}{g(x)}\right) \int u^2 K(u)du + o(h^2),$$

*and the local linear estimator has bias*

$$h^2 \frac{1}{2}\ddot{f}(x) \int u^2 K(u)du + o(h^2).$$

*At the endpoints of $[a, b]$, the local constant estimator has bias of order $h$, and the local linear estimator has bias of order $h^2$.*

More generally, let $p$ be even. Then local polynomial regression of order $p + 1$ reduces design bias and boundary bias relative to local polynomial regression of order $p$, without increasing the variance.

## *Variance Estimation*

### *Homoscedasticity*

Until the previous theorem we had been assuming homoscedasticity, i.e., $Y_i = f(x_i) + \sigma\varepsilon_i$ for all $i$, where $\mathbb{V}\varepsilon_i = 1$. In this case, we can estimate $\sigma^2$ in a simple and familiar way, namely, as the sum of the squared residuals divided by the residual degrees of freedom. More specifically, the estimator is

$$\begin{aligned}\hat{\sigma}^2 &= \frac{\sum_{i=1}^n \{Y_i - \hat{f}_n(x_i)\}^2}{n - 2\nu + \tilde{\nu}} \\ &= \frac{e'e}{n - 2\nu + \tilde{\nu}} \\ &= \frac{\|e\|^2}{n - 2\nu + \tilde{\nu}}\end{aligned}$$

where $\nu = \text{tr}(\mathbf{S})$ and $\tilde{\nu} = \text{tr}(\mathbf{S}'\mathbf{S}) = \sum_i \|s(x_i)\|^2$. Recall that $\mathbf{S}$ is the smoothing matrix.

The estimator $\hat{\sigma}^2$ is consistent for $\sigma^2$. To see this, first observe that $e = Y - \mathbf{S}Y = (\mathbf{I} - \mathbf{S})Y$, which implies that

$$\hat{\sigma}^2 = \frac{Y'\mathbf{\Lambda}Y}{\text{tr}(\mathbf{\Lambda})},$$

where $\mathbf{\Lambda} = (\mathbf{I} - \mathbf{S})'(\mathbf{I} - \mathbf{S})$. A well-known fact about quadratic forms is

$$\mathbb{E}Y'\mathbf{A}Y = \text{tr}(\mathbf{A}\mathbf{\Sigma}) + \mu'\mathbf{A}\mu,$$

where $\mathbf{\Sigma} = \mathbb{V}Y$ and $\mu = \mathbb{E}Y$. Thus

$$\begin{aligned}\mathbb{E}\hat{\sigma}^2 &= \frac{\mathbb{E}Y'\mathbf{\Lambda}Y}{\text{tr}(\mathbf{\Lambda})} \\ &= \frac{\text{tr}(\mathbf{\Lambda}\sigma^2\mathbf{I})}{\text{tr}(\mathbf{\Lambda})} + \frac{\mathbf{f}'\mathbf{\Lambda}\mathbf{f}}{n - 2\nu + \tilde{\nu}} \\ &= \sigma^2 + \frac{\mathbf{f}'\mathbf{\Lambda}\mathbf{f}}{n - 2\nu + \tilde{\nu}}. \qquad (3)\end{aligned}$$

Under mild conditions, the second term in (3) will go to zero as $n \to \infty$.

The appearance of $n - 2\nu + \tilde{\nu}$ may seem mysterious, but this quantity is in fact analogous to the residual degrees of freedom $n - p$ in ordinary linear regression. In that setting, $n - p = \text{tr}(\mathbf{I} - \mathbf{H}) = \text{tr}\{(\mathbf{I} - \mathbf{H})'(\mathbf{I} - \mathbf{H})\}$, where $\mathbf{H}$ is the hat matrix. In the current setting,

$\mathbf{I} - \mathbf{H}$ is replaced by $\mathbf{I} - \mathbf{S}$, and we have

$$\begin{aligned}\operatorname{tr}\{(\mathbf{I}-\mathbf{S})'(\mathbf{I}-\mathbf{S})\} &= \operatorname{tr}(\mathbf{I}'\mathbf{I} - \mathbf{I}'\mathbf{S} - \mathbf{S}'\mathbf{I} + \mathbf{S}'\mathbf{S}) \\ &= \operatorname{tr}(\mathbf{I} - \mathbf{S} - \mathbf{S}' + \mathbf{S}'\mathbf{S}) \\ &= \operatorname{tr}(\mathbf{I}) - \operatorname{tr}(\mathbf{S}) - \operatorname{tr}(\mathbf{S}') + \operatorname{tr}(\mathbf{S}'\mathbf{S}) \\ &= n - 2\operatorname{tr}(\mathbf{S}) + \operatorname{tr}(\mathbf{S}'\mathbf{S}) \\ &= n - 2\nu + \tilde{\nu}.\end{aligned}$$

### *Heteroscedasticity*

Now suppose that $Y_i = f(x_i) + \sigma(x_i)\varepsilon_i$. Since this implies that $\sigma$ is a (presumably non-constant) function, estimating it requires a second regression. The second regression is for the model

$$\begin{aligned}Z_i &= \log\{Y_i - f(x_i)\}^2 \\ &= \log\sigma^2(x_i)\varepsilon_i^2 \\ &= \log\sigma^2(x_i) + \log\varepsilon_i^2 \\ &= \log\sigma^2(x_i) + \delta_i.\end{aligned}$$

This model suggests that we could estimate $\log\sigma^2(x)$ by doing a regression with the log squared residuals from the first regression as the response. Specifically, we do the following.

1. Estimate $f(x)$ to arrive at $\hat{f}_n(x)$.
2. Let $Z_i = \log\{Y_i - \hat{f}_n(x_i)\}^2$.
3. Regress the $Z_i$ on the $x_i$ to get an estimate $\hat{g}(x)$ of $\log\sigma^2(x)$.
4. Let $\hat{\sigma}^2(x) = \exp\hat{g}(x)$.

## *Confidence Bands*

We would of course like to construct confidence bands for $f$. A confidence interval for $f(x)$ usually has the form

$$\hat{f}_n(x) \pm c\,\mathrm{se}(x),$$

where $c > 0$ is a constant and $\mathrm{se}(x)$ is an estimate of the standard deviation of $\hat{f}_n(x)$. Perhaps counterintuitively, such a confidence interval is not truly an interval for $f(x)$, but is instead an interval for

$$\bar{f}_n(x) = \mathbb{E}\hat{f}_n(x) = \boldsymbol{s}(x)\mathbf{f}(\boldsymbol{x}).$$

This is because there is a bias that does not disappear as the sample size becomes large.

Let $s_n(x)$ be the standard deviation of $\hat{f}_n(x)$. Then

$$\begin{aligned}\frac{\hat{f}_n(x) - f(x)}{s_n(x)} &= \frac{\hat{f}_n(x) - \bar{f}_n(x)}{s_n(x)} + \frac{\bar{f}_n(x) - f(x)}{s_n(x)} \\ &= Z_n(x) + \frac{\text{bias}\{\hat{f}_n(x)\}}{\sqrt{\mathbb{V}\hat{f}_n(x)}}.\end{aligned}$$

Typically, $Z_n(x) \Rightarrow \mathcal{N}(0,1)$. In a nonparametric setting, the second term does not go to zero as the sample size increases. This means the bias is present in the limit, which implies that the resulting confidence interval is not centered around $f(x)$. We might respond to this by

1. accepting that our confidence interval is for $\bar{f}_n(x)$ rather than $f(x)$;
2. attempting to correct the bias by estimating the bias function $\bar{f}_n(x) - f(x)$; or
3. minimizing the bias by undersmoothing.

The second option is perhaps the most tempting but is considerably more difficult than estimating $f(x)$ since the bias involves $\ddot{f}(x)$. This fact makes the first and third options more appealing. Most people go with the first option because it is difficult to choose the right amount of undersmoothing.

*Pointwise Bands*

We can construct a pointwise band by invoking asymptotic normality or by using the bootstrap. In the former case, the interval is

$$\hat{f}_n(x) \pm \Phi^{-1}(1 - \alpha/2)\text{se}(x).$$

As for the bootstrap, how we should resample depends on whether we assume homoscedasticity. If we do assume constant variance, i.e., $\sigma(x) = \sigma$, the $k$th bootstrap dataset is

$$Y_i^{(k)} = \hat{f}_n(x_i) + e_i^{(k)} \qquad (i = 1, \ldots, n),$$

where $e^{(k)} = (e_1^{(k)}, \ldots, e_n^{(k)})'$ is a sample (with replacement) of size $n$ from the vector of residuals $e = (Y_1 - \hat{f}_n(x_1), \ldots, Y_n - \hat{f}_n(x_n))'$. The endpoints of the resulting interval at $x_i$ are the $\alpha/2$ and $1 - \alpha/2$ quantiles of the bootstrap sample $\hat{f}_n^{(1)}(x_i), \ldots, \hat{f}_n^{(b)}(x_i)$.

If we assume that $\sigma(x)$ is a non-constant function, we can still do a bootstrap, but we must modify the resampling procedure. Here is the algorithm in detail.

1. Estimate $\sigma(x_i)$ to arrive at $\hat{\sigma}(x_i)$ for $i \in \{1, \ldots, n\}$.

2. Studentize the vector of residuals $(Y_1 - \hat{f}_n(x_1), \ldots, Y_n - \hat{f}_n(x_n))'$ by dividing the $i$th element by $\hat{\sigma}(x_i)$:

$$e_i = \frac{Y_i - \hat{f}_n(x_i)}{\hat{\sigma}(x_i)}.$$

3. Compute the $k$th bootstrap dataset as

$$Y_i^{(k)} = \hat{f}_n(x_i) + \hat{\sigma}(x_i)\, e_i^{(k)} \qquad (i = 1, \ldots, n),$$

where $e^{(k)} = (e_1^{(k)}, \ldots, e_n^{(k)})'$ is a sample (with replacement) of size $n$ from the vector of Studentized residuals.

4. Compute $\hat{\mathbf{f}}_n^{(k)}(\boldsymbol{x}) = \mathbf{S}\boldsymbol{Y}^{(k)}$ for $k \in \{1, \ldots, b\}$.

5. The endpoints of the confidence interval at $x_i$ are again the $\alpha/2$ and $1 - \alpha/2$ quantiles of the bootstrap sample $\hat{f}_n^{(1)}(x_i), \ldots, \hat{f}_n^{(b)}(x_i)$.

*Simultaneous Bands*

To construct a simultaneous band we use the so-called tube formula. Suppose that $\sigma$ is known, and let $I(x)$ be an interval. Then

$$\begin{aligned}
\mathbb{P}\{\bar{f}_n(x) \in I(x) \text{ for some } x \in [a,b]\} &= \mathbb{P}\left(\max_{x\in[a,b]} \frac{|\hat{f}_n(x) - \bar{f}_n(x)|}{\sigma\|\boldsymbol{s}(x)\|} > c\right) \\
&= \mathbb{P}\left(\max_{x\in[a,b]} \frac{|\sum_i \varepsilon_i s_i(x)|}{\sigma\|\boldsymbol{s}(x)\|} > c\right) \\
&= \mathbb{P}\left(\max_{x\in[a,b]} |W(x)| > c\right),
\end{aligned}$$

where

$$\begin{aligned}
W(x) &= \sum_{i=1}^{n} Z_i T_i(x), \\
Z_i &= \varepsilon_i/\sigma \sim \mathcal{N}(0,1), \\
T_i(x) &= s_i(x)/\|\boldsymbol{s}(x)\|.
\end{aligned}$$

It turns out that

$$\mathbb{P}\left(\max_x |W(x)| > c\right) \approx 2\{1 - \Phi(c)\} + \frac{\kappa}{\pi}\exp(-c^2/2)$$

for large $c$, where

$$\kappa = \int_a^b \|\dot{\boldsymbol{T}}(x)\|dx,$$

where $\dot{\boldsymbol{T}}(x) = (\dot{T}_1(x), \ldots, \dot{T}_n(x))'$. Choosing $c$ to solve

$$2\{1 - \Phi(c)\} + \frac{\kappa}{\pi}\exp(-c^2/2) = \alpha$$

yields the desired band $\hat{f}_n(x) \pm c\,\mathrm{se}(x)$.

### *Choosing the Right Bandwidth*

We want to choose $h$ to minimize the risk

$$R(h) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\{\hat{f}_n(x_i) - f(x_i)\}^2\right).$$

Since $R(h)$ depends on the unknown function $f$, we will instead minimize an estimate $\hat{R}(h)$ of $R(h)$.

It might seem sensible to estimate $R(h)$ using

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{f}_n(x_i)\}^2,$$

the so-called training error. But this estimator is biased downward and usually leads to undersmoothing. A better risk estimator is the leave-one-out cross-validation score:

$$\mathrm{CV}(h) = \hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{f}_{-i}(x_i)\}^2$$

where $\hat{f}_{-i}(x_i)$ is the estimate obtained by leaving out the $i$th observation. Intuitively, we are asking, "How well can we predict $Y_i$ if we do not use $Y_i$ in the estimation procedure?" For linear smoothers, computing this score is not as burdensome as it may seem because we do not have to recompute the estimate with each observation left out. Instead, we have

$$\mathrm{CV}(h) = \hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{Y_i - \hat{f}_n(x_i)}{1 - S_{ii}}\right\}^2,$$

where $S_{ii}$ is the $i$th diagonal element of $\mathbf{S}$.

An alternative is the generalized cross-validation score:

$$\mathrm{GCV}(h) = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{Y_i - \hat{f}_n(x_i)}{1 - \frac{1}{n}\mathrm{tr}(\mathbf{S})}\right\}^2,$$

which replaces the $S_{ii}$ with their average. Usually CV and GCV lead to bandwidths that are close to one another.