Nonparametrics and Local Methods

Richard L. Sweeney

based on slides by Chris Conlon

Empirical Methods
Spring 2019

Section outline

Why nonparametrics?

- sometimes just interested in the distribution
- sometimes this is the first stage and we want to integrate
- sometimes want to do something semiparametric

In this section, we are interested in estimating the **density** $f(x)$ under minimal assumptions.

Let's start with the histogram

One of the more successful and popular uses of nonparametric methods is estimating the density or distribution function $f(x)$ or $F(x)$.

$$\hat{f}_{HIST}(x_0) = \frac{1}{N}\sum_{i=1}^{N} \frac{\mathbf{1}(x_0 - h < x_i < x_0 + h)}{2h}$$

- Divide the dataset into bins, count up fraction of observations in each bins

Kernel Estimation

Let's rewrite the histogram estimator

$$\hat{f}_{HIST}(x_0) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x_i - x_0}{h}\right)$$

Where $K(z) = \frac{1}{2} \cdot \mathbf{1}(|z| < 1)$

Density estimator interpretation

- for each observation, there is probability mass 1 to spread around
- use the function $K(\cdot)$ and smoothing parameter h to choose how to allocate this mass
- then, for any given x_0, sum over these functions that spread out mass, and normalize by dividing by N

Smooth Kernels

We call $K(\cdot)$ a Kernel function and h the bandwidth. We usually assume

(i) $K(z)$ is symmetric about 0 and continuous.

(ii) $\int K(z)dz = 1$, $\int zK(z)dz = 0$, $\int |K(z)|dz < \infty$.

(iii) Either (a) $K(z) = 0$ if $|z| \geq z_0$ for some z_0 or
(b) $|z|K(z) \to 0$ as $|z| \to \infty$.

(iv) $\int z^K(z)dz = \kappa$ where κ is a constant.

Smooth Kernels

We call $K(\cdot)$ a Kernel function and h the bandwidth. We usually assume

(i) $K(z)$ is symmetric about 0 and continuous.

(ii) $\int K(z)dz = 1$, $\int zK(z)dz = 0$, $\int |K(z)|dz < \infty$.

(iii) Either (a) $K(z) = 0$ if $|z| \geq z_0$ for some z_0 or
(b) $|z|K(z) \to 0$ as $|z| \to \infty$.

(iv) $\int z^K(z)dz = \kappa$ where κ is a constant.

Usually we choose a smooth, symmetric K. But a common nonsmooth choice: $K(x) = (|x| < 1/2)$ gives the *histogram* estimate.

Some Common Kernels

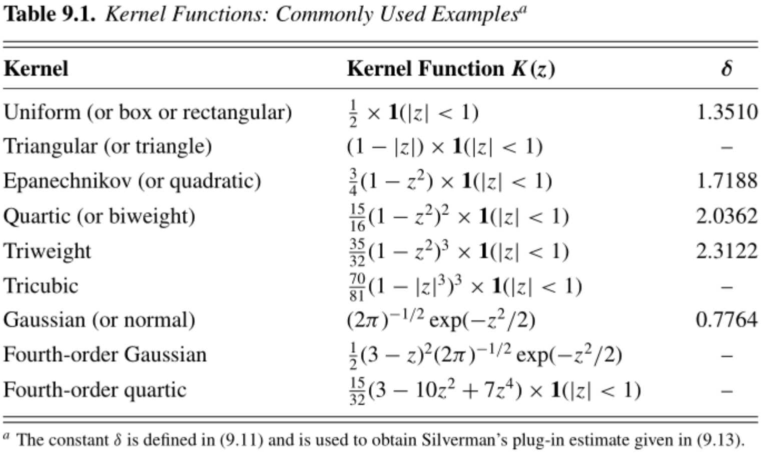


Table 9.1. *Kernel Functions: Commonly Used Examples*[a]

Kernel	**Kernel Function $K(z)$**	δ
Uniform (or box or rectangular)	$\frac{1}{2} \times \mathbf{1}(\lvert z\rvert < 1)$	1.3510
Triangular (or triangle)	$(1 - \lvert z\rvert) \times \mathbf{1}(\lvert z\rvert < 1)$	–
Epanechnikov (or quadratic)	$\frac{3}{4}(1 - z^2) \times \mathbf{1}(\lvert z\rvert < 1)$	1.7188
Quartic (or biweight)	$\frac{15}{16}(1 - z^2)^2 \times \mathbf{1}(\lvert z\rvert < 1)$	2.0362
Triweight	$\frac{35}{32}(1 - z^2)^3 \times \mathbf{1}(\lvert z\rvert < 1)$	2.3122
Tricubic	$\frac{70}{81}(1 - \lvert z\rvert^3)^3 \times \mathbf{1}(\lvert z\rvert < 1)$	–
Gaussian (or normal)	$(2\pi)^{-1/2} \exp(-z^2/2)$	0.7764
Fourth-order Gaussian	$\frac{1}{2}(3 - z)^2(2\pi)^{-1/2} \exp(-z^2/2)$	–
Fourth-order quartic	$\frac{15}{32}(3 - 10z^2 + 7z^4) \times \mathbf{1}(\lvert z\rvert < 1)$	–

[a] The constant δ is defined in (9.11) and is used to obtain Silverman's plug-in estimate given in (9.13).

Kernel Comparison

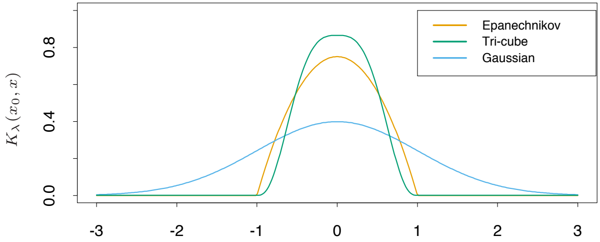


FIGURE 6.2. *A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to 1. The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov kernel has none. The Gaussian kernel is continuously differentiable, but has infinite support.*

Mean and Variance of $\hat{f}(x_0)$

Assume that the derivative of $f(x)$ exists and is bounded, and $\int zK(z)dz = 0$

Then the estimator has **bias**

$$b(x_0) = E\left[\hat{f}(x_0)\right] - f(x_0) = \frac{1}{2}h^2 f''(x_0)\int z^2K(z)dx$$

The **variance** of the estimator is

$$V\left[\hat{f}(x_0)\right] = \frac{1}{Nh}f(x_0)\int K(z)^2dz\left\{+o(\frac{1}{Nh})\right\}$$

So, unsurprisingly, the bias is *increasing* in h, and the variance is *decreasing* in h.

How to Choose h

- We want both bias and variance to be as small as possible, as usual.
- In parametric estimation, it is not a problem: they both go to zero as sample size increases.
- In nonparametric estimation reducing h reduces bias, but increases variance; how are we to make his trade off?
- Note that how we set h is going to be much more important than the choice of $K(\cdot)$

Mean Integrated Square Error

- Start with the *local* performance at x_0

$$MSE\left[\hat{f}(x_0)\right] = E\left[\left(\hat{f}(x_0) - f(x_0)\right)^2\right]$$

- Calculate the *integrated* (as opposed to expected) squared error

$$\int \left(\hat{f}(x) - f(x)\right)^2 dx = \int \text{bias}^2\left(\hat{f}(x)\right) + \text{var}\left(\hat{f}(x)\right) dx$$

- Simple approximate expression (symmetric order 2 kernels):

$$(\text{bias})^2 + \text{variance} = Ah^4 + B/nh$$

with $A = \int (f''(x))^2 \left(\int u^2 K\right)^2 /4$ and $B = f(x) \int K^2$

Optimal bandwidth

- The AMISE is
$$Ah^4 + B/nh$$
- Minimize by taking the FOC
$$h_n^* = \left(\frac{B}{4An}\right)^{1/5}$$
- bias and standard error are *both* in $n^{-2/5}$
- and the AMISE is $n^{-4/5}$—**not** $1/n$ as it is in parametric models.
- But: A and B both depend on K (known) and $f(y)$ (unknown), and especially "wiggliness" $\int (f'')^2$ (unknown, not easily estimated). Where do we go from here?

Optimal bandwidth

Can be shown that the optimal bandwidth is

$$h* = \delta \left(\int f''(x_0)^2 dx_0 \right)^{-0.2} N^{-0.2}$$

where δ depends on the kernel used (Silverman 1986) [these δ's are given in the kernel table]

Note the "optimal" kernel is Epanechnikov, although the difference is small.

Silverman's Rule of Thumb

- If f is normal with variance σ^2 (may not be a very appropriate benchmark!), the optimal bandwidth is

$$h_n^* = 1.06\sigma n^{-1/5}$$

- In practice, typically use **Silverman's plug-in estimate**:

$$h_n^* = 0.9 * \min(s, IQ/1.34) * n^{-1/5}$$

where IQ=interquartile distance

- Investigate changing it by a reasonable multiple.

Silverman's Rule of Thumb

- If f is normal with variance σ^2 (may not be a very appropriate benchmark!), the optimal bandwidth is

$$h_n^* = 1.06\sigma n^{-1/5}$$

- In practice, typically use **Silverman's plug-in estimate**:

$$h_n^* = 0.9 * \min(s, IQ/1.34) * n^{-1/5}$$

  where IQ=interquartile distance

- Investigate changing it by a reasonable multiple.

This tends to work pretty well. But can we do better?

Why not search for optimal h in our data?

- Know we want to minimize MISE.
- One option is to find the h that minimizes it *in sample*
  - Loop through increments of h
  - Calculate MISE
- Example: Old Faithful R data
  - Waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.
  - See R code in this folder.

Cross-validation

- General concept in the whole of nonparametrics: choose h to minimize a criterion $CV(h)$ that approximates

$$AMISE(h) = \int E(\hat{f}_n(x) - f(x))^2 dx.$$

- Usually programmed in metrics software. *If you can do it, do it on a subsample, and rescale.*
- CV tries to measure what the expected out of sample (OOS or EPE) prediction error of a new never seen before dataset.
- The main consideration is to prevent overfitting.
  - In sample fit is always going to be maximized by the most complicated model.
  - OOS fit might be a different story.
  - ie 1-NN might do really well in-sample, but with a new sample might perform badly.

Sample Splitting/Holdout Method and CV

Cross Validation is actually a more complicated version of sample splitting that is one of the organizing principles in machine learning literature.

Training Set	This is where you estimate parameter values.
Validation Set	This is where you choose a model- a bandwidth h or tuning parameter λ by computing the error.
Test Set	You are only allowed to look at this after you have chosen a model. Only Test Once: compute the error again on fresh data.

- Conventional approach is to allocate 50-80% to training and 10-20% to Validation and Test.
- Sometimes we don't have enough data to do this reliably.

Sample Splitting/Holdout Method

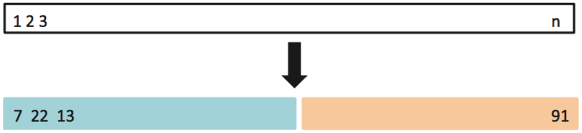


FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

Challenge with Sample Splitting

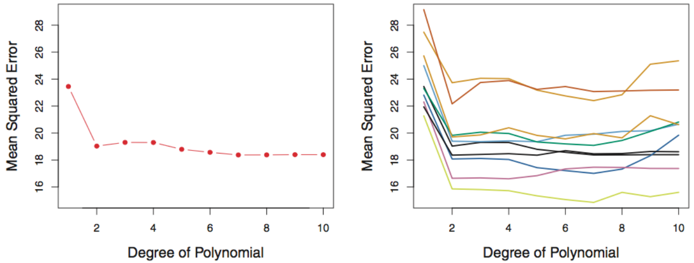


FIGURE 5.2. *The validation set approach was used on the* Auto *data set in order to estimate the test error that results from predicting* mpg *using polynomial functions of* horsepower. Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

k-fold Cross Validation

- Break the dataset into k equally sized “folds” (at random).
- Withhold $i = 1$ fold
  - Estimate the model parameters $\hat{\theta}^{(-i)}$ on the remaining $k-1$ folds
  - Predict $\hat{y}^{(-i)}$ using $\hat{\theta}^{(-i)}$ estimates for the ith fold (withheld data).
  - Compute $MSE_i = \frac{1}{k \cdot N} \sum_j (y_j^{(-i)} - \hat{y}_j^{(-i)})^2$.
  - Repeat for $i = 1, \ldots, k$.
- Construct $\widehat{MSE}_{k,CV} = \frac{1}{k} \sum_i MSE_i$

k-fold Cross Validation

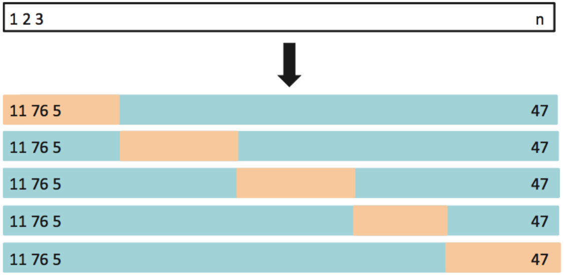


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

Leave One Out Cross Validation (LOOCV)

Same as k-fold but with $k = N$.

- Withhold a single observation i
- Estimate $\hat{\theta}_{(-i)}$.
- Predict $\hat{y}_i$ using $\hat{\theta}^{(-i)}$ estimates
- Compute $MSE_i = \frac{1}{N}\sum_j (y_i - \hat{y}_i(\hat{\theta}^{(-i)}))^2$.

Note: this requires estimating the model N times which can be costly.

LOOCV vs k-fold CV

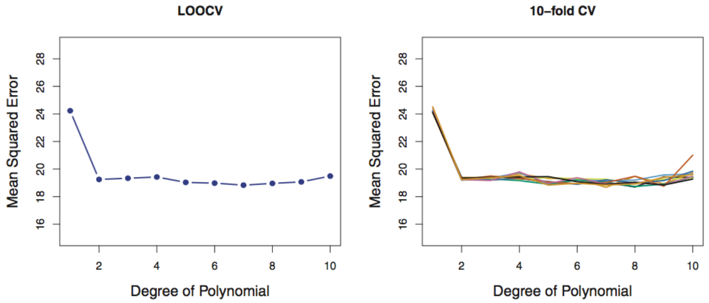


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: *10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.*

Cross Validation

- Main advantage of cross validation is that we use all of the data in both estimation and in validation.
    - For our purposes validation is mostly about choosing the right bandwidth or tuning parameter.
- We have much lower variance in our estimate of the OOS mean squared error.
    - Hopefully our bandwidth choice doesn't depend on randomness of splitting sample.

Test Data

- In Statistics/Machine learning there is a tradition to withhold 10% of the data as Test Data.
- This is completely new data that was not used in the CV procedure.
- The idea is to report the results using this test data because it most accurately simulates true OOS performance.
- We don't do much of this in economics.
  (Should we do more?)

Local Bandwidths

Local Bandwidths

If you only care about $f(y)$ at some given point, then

$$A = f''(y)^2 \left(\int u^2 K \right)^2 /4 \text{ and } B = f(y) \int K^2.$$

Local Bandwidths

If you only care about $f(y)$ at some given point, then

$$A = f''(y)^2 \left(\int u^2 K \right)^2 /4 \text{ and } B = f(y) \int K^2.$$

So in a low-density region, worry about variance and take h larger. In a curvy region, worry about bias and take h small.

Higher-Order Kernels

- K of order r iff $\int x^j K(x)dx = 0$ for $j < r$ and $\int x^r K(x)dx \neq 0$. Try $r > 2$?
- The beauty of it: bias in h^r if f is at least C^r. . . so AMISE can be reduced to $n^{-r/(2r+1)}$, almost $\sqrt{n}$-consistent if r is large.
- But gives wiggly (and sometimes negative) estimates $\rightarrow$ leave them to theorists.

Back to the CDF

Since now we have estimated the density with

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

Back to the CDF

Since now we have estimated the density with

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

a natural idea is to integrate; let $\mathcal{K}(x) = \int_{-\infty}^{x} K(t)dt$, try

$$\hat{F}_n(y) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{K}\left(\frac{y - y_i}{h}\right)$$

as a reasonable estimator of the cdf in y.

Back to the CDF

Since now we have estimated the density with

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

a natural idea is to integrate; let $\mathcal{K}(x) = \int_{-\infty}^{x} K(t)dt$, try

$$\hat{F}_n(y) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{K}\left(\frac{y - y_i}{h}\right)$$

as a reasonable estimator of the cdf in y.

Very reasonable indeed:

- when $n \longrightarrow \infty$ and h goes to zero (at rate $n^{-1/3}\dots$) it is consistent at rate $\sqrt{n}$
- it is nicely smooth
- by construction it accords well with the density estimator
- ... it is a much better choice than the empirical cdf.

Example application: Auctions

- Why auctions?
- Great introduction to structural approach. Arguably the most successful application.
- Auctions are an example of a game with assymetric information: participants know the primitives of the game, but do not know their rivals exact valuations.
- By imposing rationality/ profit maximization, we can recover the distribution of values.
- Can then run counterfatuals
- Example: Asker (AER 2008) - stamp cartel

For more detail, check out Chris Conlon's slides in this folder, or John Asker's PhD lecture notes.

Setup

Let's consider the first price sealed bid (FPSB) auction

- bidders have **private information** (or type) scalar rv X_i with realization x_i
- signals are informative: $dE[U|x]/dx > 0$
- given their signal, they make a bid b_i
- if its the highest bid, recieve utility $[U|x_i, x_{-i}] - b_i$
- else get 0
- note that if we assume values are **independent**, we get $E[U|x_i, x_{-i}] = E[U|x_i]$

This introduces a tradeoff in first price auctions: Increasing the bid increases the probability of winning; but reduces your net utility from the object.

How to bid?

Denote the equilibrium bid as B_i, with realizations b_i

Perfect Bayes Nash equilibrium:

$$max_{\tilde{b}} \left(E\left[U_i | X_i = x_i\right] - b, max_{j \in N_{-i}} B_j \leq \tilde{b} \right) \\ Pr\left(max_{j \in N_{-i}} B_j \leq \tilde{b} | X_i = x_i \right)$$

See Athey and Haile or Krishna for an accessible derivation.

Can find bid as a function of primitives

$$v(x_i, \mathbf{x_i}; N) = b_i + \frac{G_{M_i|B_i}(b_i|b_i; N)}{g_{M_i|B_i}(b_i|b_i; N)}$$

where $G_{M_i|B_i}$ and $g_{M_i|B_i}$ are the CDF and PDF of the max bids given b_i and N

- we are typically interested in the LHS
- RHS is stuff we can observe or compute
- nice linear structure makes this easy to work with
- Guerre, Perrigne and Vuong (2000): can leverage assumption of equilibrium best response and invertibility of b to recover v

Estimation strategy I

[Assumption: FPSB with symmetric IPV]

1. Leverage independence assumption:

$$\begin{aligned} G_{M_i|B_i}(m_i|b_i;N) &= G_{M_I|n} \\ &= Pr(max_{j\neq i}B_j \leq m_i|n) \end{aligned}$$

2. Value equation becomes

$$u = b + \frac{G_B\,(b|n)}{(n-1)g_B\,(b|n)}$$

   where G_B and g_B are now the marginal distribution of equilibrium bids and the densities in n bidder auctions

Estimation strategy II

3. can estimate G and g using kernels
4. now have

$$\hat{u} = b + \frac{\hat{G}_B\left(b|n\right)}{(n-1)\hat{g}_B\left(b|n\right)}$$

5. finally, can recover the distribution of values with another kernel

$$\hat{f}(u) = \frac{1}{T_n h_f} \sum_{T=1} \frac{1}{n_t} \sum_{i=1}^{n} K\left(\frac{u_i - \hat{u}_{it}}{h_f}\right)$$

Algoritm

- for each b_o, estimate $\hat{G}_B(b_0|n)$ and $\hat{g}_B(b_0|n)$ using **all the data**
- infer $\hat{u}(b_0)$
- estimate $\hat{f}$
- plot bids, adjust bandwidth etc
- run counterfactuals

Nonparametric Regression

- Often we're interested in $E\left[Y_i|X_i = x\right]$
- If X is discrete, can just average for each value
- But often times we want to smooth across values of X
    - Each bin could have small n. [Likely if X has many dimensions]
    - X could be continuous
- One option is to pick a parametric functional form $y = f(x)$. But often hard to think about how sensible these assumptions are (and what they impose on the economics of the problem)
- An alternative is to extend the concepts of nonparametric density estimation

A Fake Data Example

HTF 2.3 includes an example classification problem.

An (unknown) model maps a pair of inputs X_1 and X_2 into classes of either BLUE or ORANGE.

Training data: Imagine we have 100 points from each class.

OLS solution

$$\begin{aligned} Y = &\quad ORANGE \text{ if } Y* = x^T\hat{\beta} &> 0.5 \\ Y = &\quad BLUE \text{ if } Y* = x^T\hat{\beta} &\leq 0.5 \end{aligned}$$

Linear Probability Model

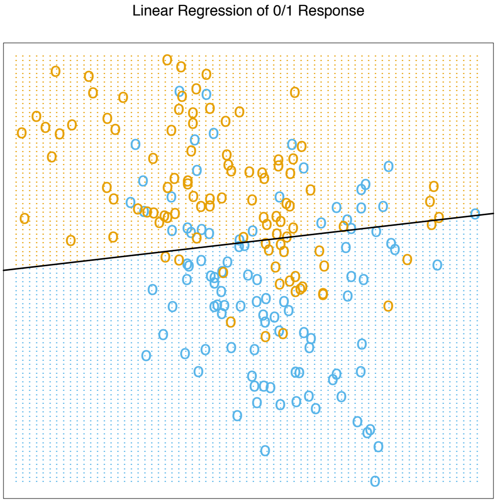


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE.

Is this the best we can do?

Consider two DGPs:

1. Draws from bivariate normal distribution with uncorrelated components but different means (2 overlapping types)
2. Mixture of 10 low variance (nearly point mass) normal distributions where the individual means were drawn from another normal distribution. (10 nearly distinct types).

In the case of 1, OLS is the best we can do.

In the case of 2, OLS will perform very poorly.

Alternative

- Lots of potential alternatives to our decision rule.
- A simple idea is to hold a majority vote of neighboring points

$$Y^* = \frac{1}{k} \sum_{x_{-i} \in N_k(x)} y_i$$

- How many parameters does this model have: None? One? k?
- Technically it has something like N/k.
- As $N \to \infty$ this means we have an infinite number of parameters! (This is a defining characteristic of non-parametrics).

15 Nearest Neighbor

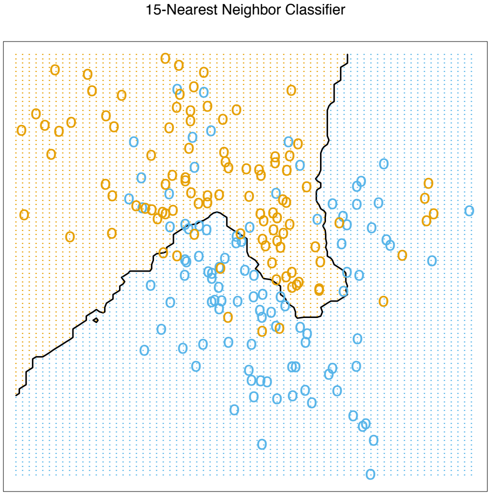


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

Extreme: 1 Nearest Neighbor

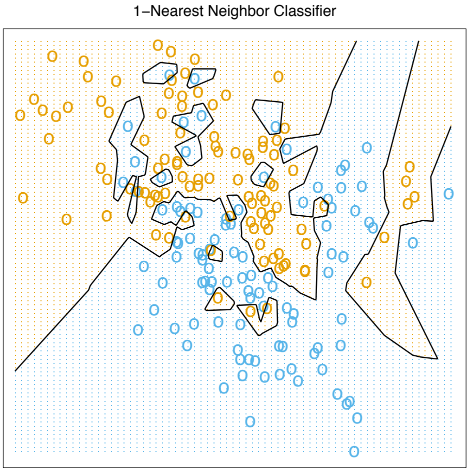


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Bias Variance Decomposition

We can decompose any estimator into two components

$$\underbrace{E[(y-\hat{f}(x))^2]}_{MSE} = \underbrace{\left(E[\hat{f}(x)-f(x)]\right)^2}_{Bias^2} + \underbrace{E\left[\left(\hat{f}(x)-E[\hat{f}(x)]\right]\right)^2}_{Variance}$$

- In general we face a tradeoff between bias and variance.
- In k-NN as k gets large we reduce the variance (each point has less influence) but we increase the bias since we start incorporating far away and potentially irrelevant information.
- In OLS we minimize the variance among unbiased estimators assuming that the true f is linear and using the entire dataset.

Big Data

- It used to be that if you had $N = 50$ observations then you had a lot of data.
- Those were the days of finite-sample adjusted t-statistics.
- Now we frequently have 1 million observations or more, why can't we use k-NN type methods everywhere?

Curse of Dimensionality

Take a unit hypercube in dimension p and we put another hypercube within it that captures a fraction of the observations r within the cube

- Since it corresponds to a fraction of the unit volume, r each edge will be $e_p(r) = r^{1/p}$.
- $e_{10}(0.01) = 0.63$ and $e_{10}(0.1) = 0.80$, so we need almost 80% of the data to cover 10% of the sample!
- If we choose a smaller r (include less in our average) we increase variance quite a bit without really reducing the required interval length substantially.

Curse of Dimensionality

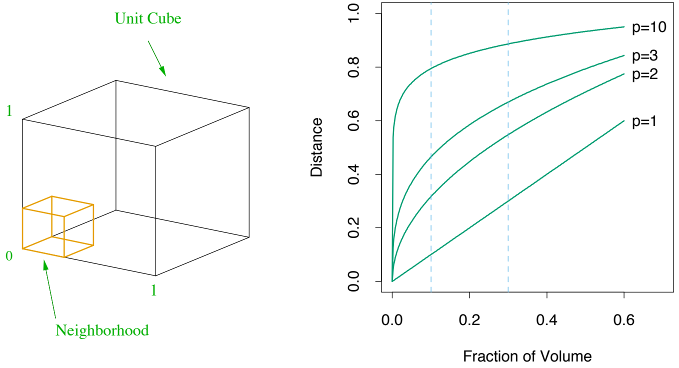


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

Curse of Dimensionality

Don't worry, it only gets worse:

$$d(p, N) = \left(1 - \left(\frac{1}{2}\right)^{1/N}\right)^{1/p}$$

- $d(p, N)$ is the distance from the origin to the closest point.
- $N = 500$ and $p = 10$ means $d = 0.52$ or that the closest point is closer to the boundary than the origin!
- Why is this a problem?
- In some dimension nearly every point is the closest point to the boundary – when we average over nearest neighbors we are extrapolating not interpolating.

Back to Bias - Variance

What minimizes MSE?

$$f(x_i) = E[Y_i|X_i]$$

- Seems simple enough (but we are back where we started).
- How do we compute the expectation ?
- k-NN tries to use local information to estimate conditional mean
- OLS uses entire dataset and adds structure $y = x\beta$ to the problem.
- A natural middleground point is to use a smoother that weights "close" observations more than "far" ones.

Common weights

Consider the following **local weighted average estimator**:

$$\hat{m}(x_0) = \sum_{i=1}^{N} w_{i0,h} y_i$$

where $w_{i0,h} = w(x_i, x_0, h)$ and $\sum_i w_{i0,h} = 1$.

Common weights

Consider the following **local weighted average estimator**:

$$\hat{m}(x_0) = \sum_{i=1}^{N} w_{i0,h} y_i$$

where $w_{i0,h} = w(x_i, x_0, h)$ and $\sum_i w_{i0,h} = 1$.

Rearranging, can see that OLS uses the following weights

$$\hat{m}_{OLS}(x_0) = \sum_{i=1}^{N} \{ \frac{1}{N} + \frac{(x_0 - \bar{x})(x_i - \bar{x})}{\sum_j (x_j - \bar{x})^2} \} y_i$$

CT note that these weights can actually *increase* with the distance between x_0 and x_i! (for example if $x_i > x_0 > \bar{x}$)

k-NN is simply a running average

$$\hat{m}_k(x_0) = \frac{1}{k}(y_{i-(k-1)/2} + ... + y_{i+(k-1)/2}$$

Can immediately see that this will not be great at the end points.

For the smallest and largest x, the average is one sided. This is the **boundary problem**.

Nonparametric Regression

Of course, we could also average all the observations within some bandwidth h

Nadaraya-Watson:

$$\hat{m}_n(x) = \frac{\sum_{i=1}^{n} y_i K\left(\frac{x-x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right)}.$$

where $K(\cdot)$ is a kernel weighting function as above.

Again, bias in h^2 and variance in $1/nh$ if $p_x = 1$.

Choosing h

Plug-in estimates work badly here.

In most cases leave one out cross-validation is feasible

Can be shown that this amounts to

$$\min_h CV(h) = \sum_{i=1}^{N} \left(\frac{y_i - \hat{m}(x_i)}{1 - \left[w_{ii,h} / \sum_j wji, h \right]} \right)^2$$

So not that hard: for each h, only need to compute one weighted average $\hat{m}(x_i)$ for each N.

Multivariate Kernels

Typically we are interested in more than one regressor

$$y_i = m(x_{1i}, ..., x_{ki})$$

the NW kernel estimator extends naturally to k dimensions

$$\hat{m}(\mathbf{x_0}) = \frac{\sum_{i=1}^{N} y_i K\left(\frac{\mathbf{x_1}-\mathbf{x_0}}{h}\right)}{\sum_{i=1}^{N} K\left(\frac{\mathbf{x_1}-\mathbf{x_0}}{h}\right)}.$$

where we just use a mutivariate kernel.

If you rescale by dividing by the standard deviation, you can even use a common bandwidth.

Silverman's Table

Silverman's Table

Silverman (1986 book) provides a table illustrating the difficulty of kernel estimation in high dimensions. To estimate the density at 0 of a $N(0,1)$ with a given accuracy, he reports:

Silverman's Table

Silverman (1986 book) provides a table illustrating the difficulty of kernel estimation in high dimensions. To estimate the density at 0 of a $N(0,1)$ with a given accuracy, he reports:

Dimensionality Required	Sample Size
1	4
2	19
5	786
7	10,700
10	842,000

Silverman's Table

Silverman (1986 book) provides a table illustrating the difficulty of kernel estimation in high dimensions. To estimate the density at 0 of a $N(0,1)$ with a given accuracy, he reports:

Dimensionality Required	Sample Size
1	4
2	19
5	786
7	10,700
10	842,000

Not to be taken lightly. . . in any case convergence with the optimal bandwidth is in $n^{-2/(4+p_y)}$ now—and Silverman's rule of thumb for choosing h_n^* must be adapted too.

Back to one dimesion I

We can interpret the standard kernel estimator as estimating a constant regressino function $g(x) = m$

where

$$\hat{m} = \text{arg min}_{m_0} \sum_{i=1}^{N} K\left(\frac{x_i - x_0}{h}\right) \cdot (y_i - m_0)^2$$

Taking the FOC, we see that this yields

$$\hat{g}(x) = \hat{m} = \sum_{i=1}^{N} K\left(\frac{x_i - x_0}{h}\right) \cdot y_i / \sum_{i=1}^{N} K\left(\frac{x_i - x_0}{h}\right)$$

Local Linear Regression I

This suggests other functions instead of constants (that is why we're interested in this regression in the first place!)

A natural option is **local linear regression**:

$$m(x) = \alpha + \beta(x - x_0)$$

where

$$(\hat{\alpha}, \hat{\beta}) = \text{arg min}_{m_0} \sum_{i=1}^{N} K\left(\frac{x_i - x_0}{h}\right) \cdot (y_i - \alpha + \beta(x_i - x_0))^2$$

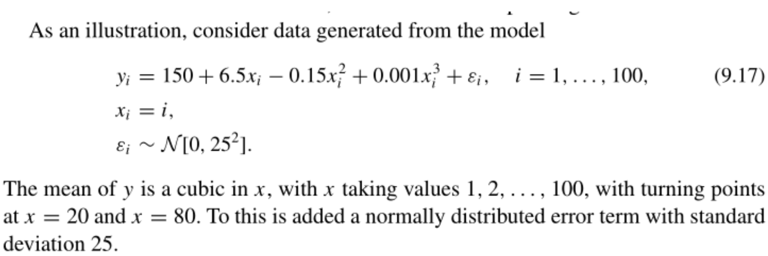
Local Linear Regression II

Advantages:

- the bias becomes 0 if the true $m(x)$ is linear.
- the coefficient of $(x - x_i)$ estimates $m'(x)$.
- behaves better in "almost empty" regions.

Disadvantages: hardly any, just do it! How?

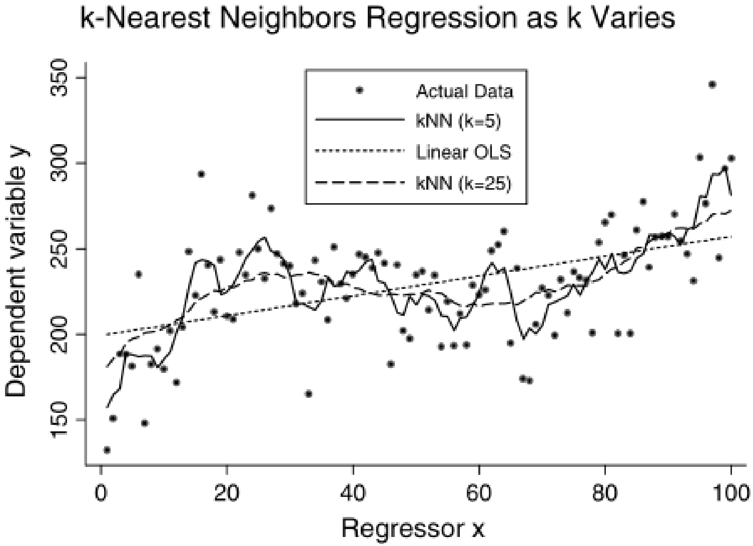
Example from Cameron and Trivedi


As an illustration, consider data generated from the model

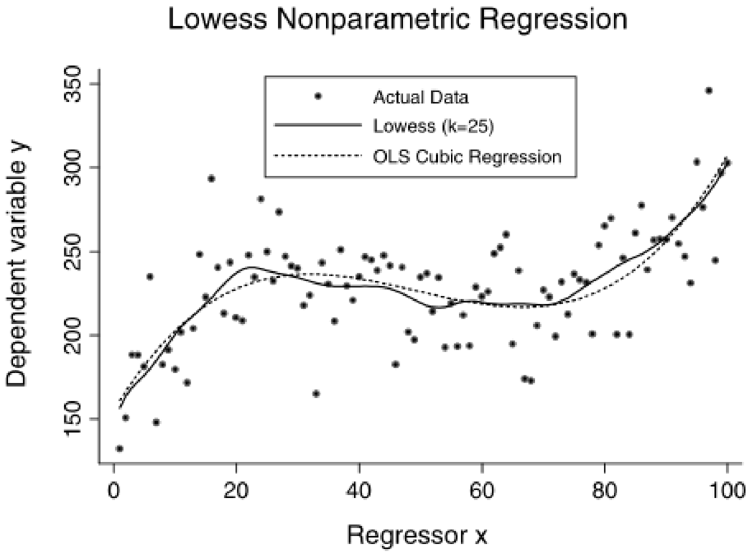
$$
\begin{aligned}
y_i &= 150 + 6.5x_i - 0.15x_i^2 + 0.001x_i^3 + \varepsilon_i, \quad i = 1, \ldots, 100, \qquad (9.17)\\
x_i &= i,\\
\varepsilon_i &\sim \mathcal{N}[0, 25^2].
\end{aligned}
$$

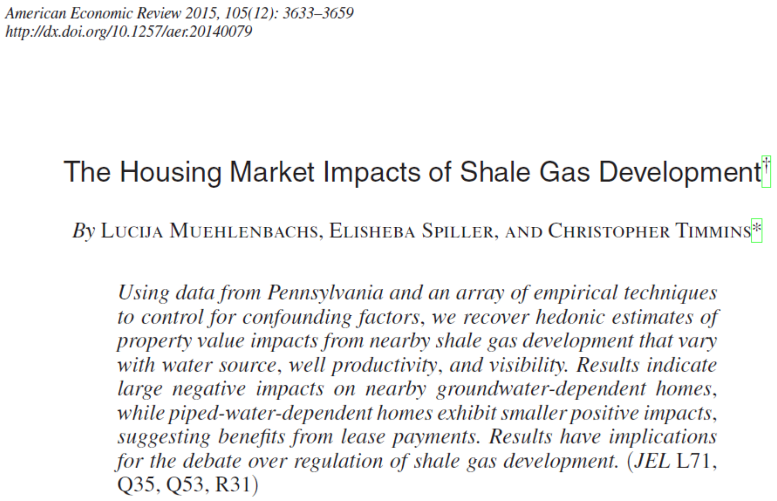
The mean of y is a cubic in x, with x taking values $1, 2, \ldots, 100$, with turning points at $x = 20$ and $x = 80$. To this is added a normally distributed error term with standard deviation 25.

k-NN performance



Lowess


Lowess Nonparametric Regression
Actual Data
Lowess (k=25)
OLS Cubic Regression
Dependent variable y
Regressor x
150
200
250
300
350
0
20
40
60
80
100



American Economic Review 2015, 105(12): 3633–3659
http://dx.doi.org/10.1257/aer.20140079

The Housing Market Impacts of Shale Gas Development[†]

By Lucija Muehlenbachs, Elisheba Spiller, and Christopher Timmins[*]

Using data from Pennsylvania and an array of empirical techniques to control for confounding factors, we recover hedonic estimates of property value impacts from nearby shale gas development that vary with water source, well productivity, and visibility. Results indicate large negative impacts on nearby groundwater-dependent homes, while piped-water-dependent homes exhibit smaller positive impacts, suggesting benefits from lease payments. Results have implications for the debate over regulation of shale gas development. (*JEL* L71, Q35, Q53, R31)

Use Lowess Plot to Motivate DD

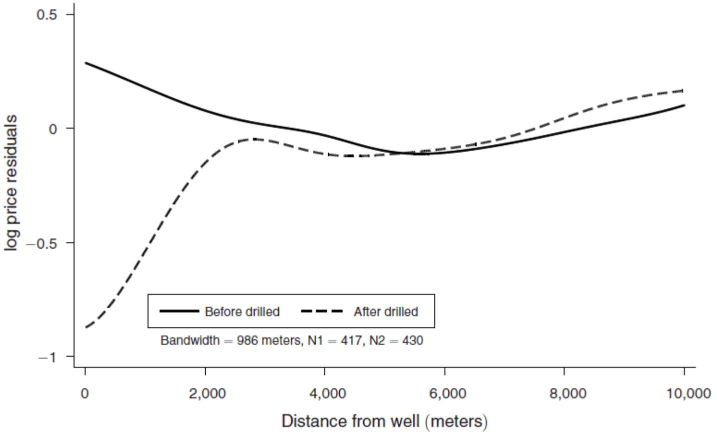


FIGURE 5. PRICE GRADIENT OF DISTANCE FROM FUTURE/CURRENT WELL, USING GROUNDWATER-DEPENDENT AREAS

Nonparametric Regression, summary, 1

Nadaraya–Watson for $E(y|x) = m(x)$

$$\hat{m}(x) = \frac{\sum_i y_i K_h(x - x_i)}{\sum_i K_h(x - x_i)}$$

- bias in $O(h^2)$, variance in $1/(nh^{p_x})$
- optimal h in $n^{-1/(p+4)}$: then bias, standard error and RMSE all converge at rate $n^{-2/(p+4)}$
- to select h, no rule of thumb: cross-validate on a subsample and scale up.

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Better: **local linear regression**

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Better: **local linear regression**

1. regress y_i on 1 and $(x_i - x)$ with weight $K_h(x - x_i)$

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Better: **local linear regression**

1. regress y_i on 1 and $(x_i - x)$ with weight $K_h(x - x_i)$
2. take the estimated coeffs as your $\hat{m}(x)$ and $\hat{m}'(x)$.

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Better: **local linear regression**

1. regress y_i on 1 and $(x_i - x)$ with weight $K_h(x - x_i)$
2. take the estimated coeffs as your $\hat{m}(x)$ and $\hat{m}'(x)$.

Nonparametric Regression, summary, 2

Nadaraya–Watson=**local constant regression**: to get $\hat{m}(x)$,

1. regress y_i on 1 with weight $K_h(x - x_i)$
2. take the estimated coeff as your $\hat{m}(x)$.

Better: **local linear regression**

1. regress y_i on 1 and $(x_i - x)$ with weight $K_h(x - x_i)$
2. take the estimated coeffs as your $\hat{m}(x)$ and $\hat{m}'(x)$.

To estimate the standard errors: bootstrap on an *undersmoothed* estimate (so that bias is negligible.)

Seminonparametric (=Flexible) Regression

- Linear model won't fit well when true relationship is nonlinear
- But fully flexible models may require a lot of data
- Alternative is to replace X with M transformations of $h_m(X)$
- Then model $f(X) = \sum_m \beta_m h_m(X)$

Piecewise polynomials

- First consider a single dimensional X
- Divide f into contiguous intervals, and pick a function to represent y on each interval
- A natural starting point would be a piecewise constant function, broken up by boundary points ξ_1 and xi_2

$$h_1 = I(X < \xi_1); h_2 = I(\xi_1 < X < \xi_2); h_3 = I(\xi_2 < X)$$

- This predicts the interval average for all X
- We can improve this by adding three additional basis functions: $h_{m+3} = h_m(X)X$.
- We can then make it continuous by

$$\begin{aligned} h_1 &= 1 \\ h_2 &= X \\ h_3 &= (X - \xi_1)_+ \\ h_4 &= (X - \xi_2)_+ \end{aligned}$$

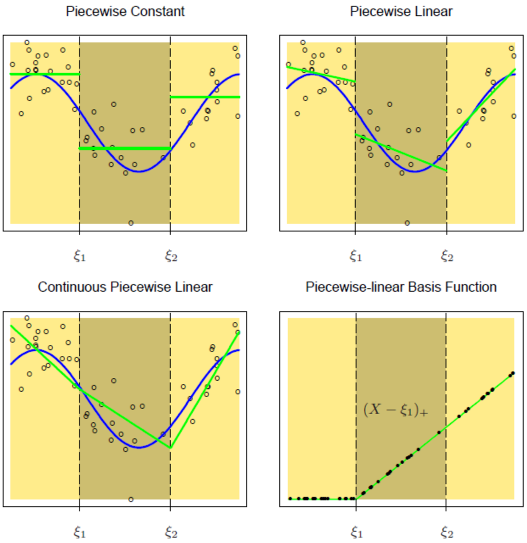


FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise linear functions fit to the same data—the top right unrestricted, and the lower left restricted to be continuous at the knots. The lower right panel shows a piecewise–linear basis function, $h_3(X) = (X - \xi_1)_+$, continuous at ξ_1. The black points indicate the sample evaluations $h_3(x_i)$, $i = 1, \ldots, N$.*

Smooth splines

- We can improve this further by increasing the order of the local polynomial
- A common option is cubic splines:

$$h_1 = 1; h_2 = X; h_3 = X^2$$
$$h_4 = x^3; h_5(X - \xi_1)_+^3; h_6 = (X - \xi_2)_+^3$$

- This is a six dimensional space: (3 regions) X (4 parameters per region) - (2 knots) X (3 constraints per knot)
- In practice people never use higher than cubic
- Picking the knots is more subjective (use CV?)
- Also typically use B-splines (see HTF)

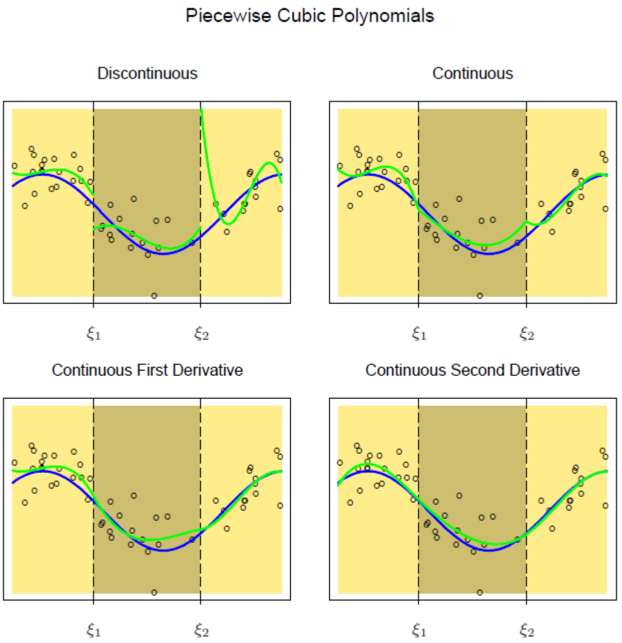


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

Cubic Splines

Note that simply polynomials often perform poorly in some parts of the data!

Another common option is

$$\min_{m(.)} \sum_{i}^{N} (y_i - m(x_i))^2 + \lambda J \int (m''(x))^2 dx$$

where λ is a **smoothing parameter** and m is a cubic polynomial

Then we "obtain" the natural cubic spline with knots=$(x_1, \ldots, x_n)$:

- m is a cubic polynomial between consecutive x_i's

Cubic Splines

Note that simply polynomials often perform poorly in some parts of the data!

Another common option is

$$\min_{m(.)} \sum_{i}^{N} (y_i - m(x_i))^2 + \lambda J \int (m''(x))^2 dx$$

where λ is a **smoothing parameter** and m is a cubic polynomial

Then we "obtain" the natural cubic spline with knots=$(x_1, \ldots, x_n)$:

- m is a cubic polynomial between consecutive x_i's
- it is linear out-of-sample

Cubic Splines

Note that simply polynomials often perform poorly in some parts of the data!

Another common option is

$$\min_{m(.)} \sum_{i}^{N} (y_i - m(x_i))^2 + \lambda J \int (m''(x))^2 dx$$

where λ is a **smoothing parameter** and m is a cubic polynomial

Then we “obtain” the natural cubic spline with knots=$(x_1, \dots, x_n)$:

- m is a cubic polynomial between consecutive x_i's
- it is linear out-of-sample
- it is C^2 everywhere.

Cubic Splines

Note that simply polynomials often perform poorly in some parts of the data!

Another common option is

$$\min_{m(.)} \sum_{i}^{N} (y_i - m(x_i))^2 + \lambda J \int (m''(x))^2 dx$$

where λ is a **smoothing parameter** and m is a cubic polynomial

Then we "obtain" the natural cubic spline with knots=$(x_1, \ldots, x_n)$:

- m is a cubic polynomial between consecutive x_i's
- it is linear out-of-sample
- it is C^2 everywhere.

Cubic Splines

Note that simply polynomials often perform poorly in some parts of the data!

Another common option is

$$\min_{m(.)} \sum_{i}^{N} (y_i - m(x_i))^2 + \lambda J \int (m''(x))^2 dx$$

where λ is a **smoothing parameter** and m is a cubic polynomial

Then we “obtain” the natural cubic spline with knots=$(x_1, \ldots, x_n)$:

- m is a cubic polynomial between consecutive x_i's
- it is linear out-of-sample
- it is C^2 everywhere.

“Consecutive” implies one-dimensional. . . harder to generalize to $p_x > 1$.

Cubic Splines

Note that simply polynomials often perform poorly in some parts of the data!

Another common option is

$$\min_{m(.)} \sum_{i}^{N} (y_i - m(x_i))^2 + \lambda J \int (m''(x))^2 dx$$

where λ is a **smoothing parameter** and m is a cubic polynomial

Then we "obtain" the natural cubic spline with knots=$(x_1, \ldots, x_n)$:

- m is a cubic polynomial between consecutive x_i's
- it is linear out-of-sample
- it is C^2 everywhere.

"Consecutive" implies one-dimensional…harder to generalize to $p_x > 1$.

Semiparametric Regression

- Previous methods placed no structure on the model
- Sometime we're willing to impose *some* structure, typically to isolate a parameter or ratio that we're particularly interested in
- Fully non-parametric methods controlling for other factors not possible due to the curse of dimensionality.

Examples

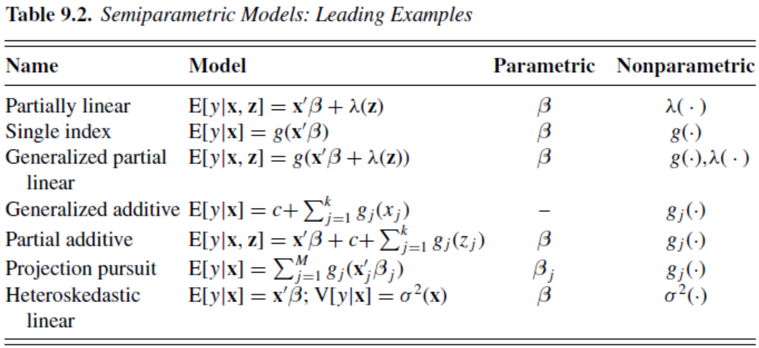


Table 9.2. *Semiparametric Models: Leading Examples*

Name	Model	Parametric	Nonparametric
Partially linear	$E[y\|\mathbf{x},\mathbf{z}] = \mathbf{x}'\beta + \lambda(\mathbf{z})$	β	$\lambda(\cdot)$
Single index	$E[y\|\mathbf{x}] = g(\mathbf{x}'\beta)$	β	$g(\cdot)$
Generalized partial linear	$E[y\|\mathbf{x},\mathbf{z}] = g(\mathbf{x}'\beta + \lambda(\mathbf{z}))$	β	$g(\cdot),\lambda(\cdot)$
Generalized additive	$E[y\|\mathbf{x}] = c + \sum_{j=1}^{k} g_j(x_j)$	–	$g_j(\cdot)$
Partial additive	$E[y\|\mathbf{x},\mathbf{z}] = \mathbf{x}'\beta + c + \sum_{j=1}^{k} g_j(z_j)$	β	$g_j(\cdot)$
Projection pursuit	$E[y\|\mathbf{x}] = \sum_{j=1}^{M} g_j(\mathbf{x}_j'\beta_j)$	β_j	$g_j(\cdot)$
Heteroskedastic linear	$E[y\|\mathbf{x}] = \mathbf{x}'\beta;\ V[y\|\mathbf{x}] = \sigma^2(\mathbf{x})$	β	$\sigma^2(\cdot)$

For a formal treatment of these and discussion of convergence, see ?.

Partial linear model

We'll focus on the most commonly used model: the **partial linear model**

$$y_i = X_i'\beta + g(Z_i) + e_i$$

We are interested in β, but $g()$ is an unrestricted (unknown) function of z.

Identification requires $Cov[X, g(Z)] = 0$

? showed that we can concentrate this $g()$ out of the model.

Robinson 1988 I

$$y_i = X_i'\beta + g(Z_i) + e_i$$

where $e_i = y_i - E[y_i|X_i, Z_i]$.

Take the conditional expectation

$$E[y_i|Z_i] = E[X_i|Z_i]'\beta + g(Z_i) + E[e_i|Z_i]$$

then note that $E[e_i|Z_i] = E[E[e_i|X_i, Z_i]] = 0$

Subtractcing yields

$$y_i - E[y_i|Z_i] = (X_i - E[X_i|Z_i])'\beta + e_i$$

Robinson 1988 II

So this suggests a natural procedure:

1. Estimate the conditional expectations $m_y(z) = E[y_i|Z_i]$ and $m_x(z) = E[X_i|Z_i]$ separately (and flexibly) **for each** x using the methods discussed in the previous section.

2. Estimate β using OLS on the residuals

$$y_i - \hat{m}_y(z_i) = (X_i - \hat{m}_x(z_i))'\beta + e_i$$

Recent example: Hastings and Shapiro (2018)

Single-Index Models I

- Assume we are willing to specify that the conditional mean is a scalar function of the linear combination of $X's$

$$E\left[y|\mathbf{x}\right] = g(\mathbf{x}'\beta)$$

- Often we'll assume $g()$ as in logit and probit.
- Instead, note that the conditional mean of a change in x

$$\delta \equiv E\left[g'(\mathbf{x}'\beta)\beta_j\right]$$

- Suggests the average derivative estimator which doesn't require normality assumptions of probit

Single-Index Models II

- Let $m(x_i) = g(x_i'\beta)$.

$$\hat{\delta}_{AD} = -\frac{1}{N}\sum_{i=1}^{N} y_i \frac{f'(x_i)}{f(x_i)}$$

where $f(x)$ is the density of x

- More robust, but potentially less efficient if true distribution of $\varepsilon \sim N(0,1)$.
- Can use Marginal Rate of Substitution (MRS) to identify β_k/β_l.

$$\frac{\partial\left[y|\mathbf{x}\right]/\partial x_j}{\partial\left[y|\mathbf{x}\right]/\partial x_k} = \frac{\beta_j}{\beta_k}$$

Bootstrap and Delta Method

- We know how to construct confidence intervals for parameter estimates: $\hat{\theta}_k \pm 1.96SE(\hat{\theta}_k)$
- Often we are asked to construct standard errors or confidence intervals around model outputs that are not just parameter estimates: ie: $g(x_i, \hat{\theta})$.
- Sometimes we can't even write $g(x_i, \theta)$ as an explicit function of θ ie: $\Psi(g(x_i, \theta), \theta) = 0$.
- Two options:
    1. Delta Method
    2. Bootstrap

Delta Method

Delta method works by considering a Taylor Expansion of $g(x_i, \theta)$.

$$g(z) \approx g(z_0) + g'(z_0)(z - z_0) + o(||z - z_0||)$$

Assume that θ_n is asymptotically normally distributed so that:

$$\sqrt{n}(\theta_n - \theta_0) \sim N(0, \Sigma)$$

Then we have that

$$\sqrt{n}(g(\theta_n) - g(\theta_0)) \sim N(0, D(\theta)'\Sigma D(\theta))$$

Where $D(\theta) = \frac{\partial g(x_i,\theta)}{\partial \theta}$ is the Jacobian of g with respect to theta evaluated at θ.
We need g to be continuously differentiable around the center of our expansion θ.

Delta Method: Examples

Start with something simple: $Y = \overline{X}_1 \cdot \overline{X}_2$ with $(X_{1i}, X_{2i}) \sim IID$. We know the CLT applies so that:

$$\sqrt{n}\begin{pmatrix}\overline{X}_1 - \mu_1 \\ \overline{X}_2 - \mu_2\end{pmatrix} \sim N\left[\begin{pmatrix}0\\0\end{pmatrix}, \Sigma\right]$$

The Jacobian is just $D(\theta) = \begin{pmatrix}\frac{\partial g(\theta)}{\partial \theta_1} \\ \frac{\partial g(\theta)}{\partial \theta_2}\end{pmatrix} = \begin{pmatrix}\mu_2 \\ \mu_1\end{pmatrix}$

So,

$$V(Y) = D(\theta)'\Sigma D(\theta) = \begin{pmatrix}\mu_2 & \mu_1\end{pmatrix}\begin{pmatrix}\sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22}\end{pmatrix}\begin{pmatrix}\mu_2 \\ \mu_1\end{pmatrix}$$

$$\sqrt{n}(\overline{X}_1\overline{X}_2 - \mu_1\mu_2) \sim N(0, \mu_2^2\sigma_{11}^2 + 2\mu_1\mu_2\sigma_{12} + \mu_1^2\sigma_{22}^2)$$

Delta Method: Examples

Think about a simple logit:

$$P(Y_i = 1|X_i) = \frac{\exp^{\beta_0+\beta_1 X_i}}{1+\exp^{\beta_0+\beta_1 X_i}} \quad P(Y_i = 0|X_i) = \frac{1}{1+\exp^{\beta_0+\beta_1 X}}$$

Remember the "trick" to use GLM (log-odds):

$$\log P(Y_i = 1|X_i) - \log P(Y_i = 0|X_i) = \beta_0 + \beta_1 X_i$$

- Suppose that we have estimated $\hat{\beta}_0, \hat{\beta}_1$ via GLM/MLE but we want to know the confidence interval for the probability: $P(Y_i = 1|X_i, \hat{\theta})$
- The derivatives are a little bit tricky, but the idea is the same.
- This is what STATA should be doing when you type: `mfx, compute`

Delta Method: Learning example

Often we have a nonlinear model that we transform and estimate linearly.

Here is an example on learning curves from Eric Zivot.

Example 2 *Estimation of Generalized Learning Curve*

Consider the generalized learning curve (see Berndt, 1992, chapter 3)

$$C_t = C_1 N_t^{\alpha_c/R} Y_t^{(1-R)/R} \exp(u_t)$$

where

$$\begin{aligned} C_t &= \text{real unit cost at time } t \\ N_t &= \text{cumulative production up to time } t \\ Y_t &= \text{production in time } t \\ & u_t \sim iid\ (0, \sigma^2) \\ \alpha_c &= \text{learning curve parameter} \\ R &= \text{returns to scale parameter} \end{aligned}$$

The generalized learning curve may be converted to a linear regression model by taking logs:

$$\begin{aligned}
\ln C_t &= \ln C_1 + \left(\frac{\alpha_c}{R}\right) \ln N_t + \left(\frac{1-R}{R}\right) \ln Y_t + u_t \\
&= \beta_0 + \beta_1 \ln N_t + \beta_2 \ln Y_t + u_t \\
&= \mathbf{x}_t' \boldsymbol{\beta} + u_t
\end{aligned}$$

where

$$\begin{aligned}
\beta_0 &= \ln C_1 \\
\beta_1 &= \alpha_c / R \\
\beta_2 &= (1-R)/R \\
\mathbf{x}_t &= (1, \ln N_t, \ln Y_t)'.
\end{aligned}$$

The learning curve parameters may be recovered using

$$\begin{aligned} \alpha_c &= \frac{\beta_1}{1+\beta_2} = g_1(\boldsymbol{\beta}) \\ R &= \frac{1}{1+\beta_2} = g_2(\boldsymbol{\beta}) \end{aligned}$$

Least squares gives consistent and asymptotically normal estimates

$$\begin{aligned}
\hat{\boldsymbol\beta} &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \xrightarrow{p} \boldsymbol\beta \\
\hat\sigma^2 &= n^{-1}\sum_{t=1}^{n}(y_t - \mathbf{x}_t'\hat{\boldsymbol\beta})^2 \xrightarrow{p} \sigma^2 \\
&\hat{\boldsymbol\beta} \overset{A}{\sim} N(\boldsymbol\beta, \hat\sigma^2(\mathbf{X}'\mathbf{X})^{-1})
\end{aligned}$$

Then from Slutsky's Theorem

$$\begin{aligned}
\hat\alpha_c &= \frac{\hat\beta_1}{1+\hat\beta_2} \xrightarrow{p} \frac{\beta_1}{1+\beta_2} = \alpha_c \\
\hat R &= \frac{1}{1+\hat\beta_2} \xrightarrow{p} \frac{1}{1+\beta_2} = R
\end{aligned}$$

provided $\beta_2 \neq -1$.

We can use the delta method to get the asymptotic distribution of $\hat{\eta} = (\hat{\alpha}_c, \hat{R})'$:

$$\begin{aligned}\begin{pmatrix} \hat{\alpha}_c \\ \hat{R} \end{pmatrix} &= \begin{pmatrix} g_1(\hat{\boldsymbol{\beta}}) \\ g_2(\hat{\boldsymbol{\beta}}) \end{pmatrix} \\ &\overset{A}{\sim} N\left(\mathbf{g}(\boldsymbol{\beta}), \left(\frac{\partial \mathbf{g}(\hat{\boldsymbol{\beta}})}{\partial \boldsymbol{\beta}'}\right) \hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1} \left(\frac{\partial \mathbf{g}(\hat{\boldsymbol{\beta}})}{\partial \boldsymbol{\beta}'}\right)'\right)\end{aligned}$$

where

$$\begin{aligned}\frac{\partial \mathbf{g}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}'} &= \begin{pmatrix} \frac{\partial g_1(\boldsymbol{\beta})}{\partial \beta_1} & \frac{\partial g_1(\boldsymbol{\beta})}{\partial \beta_2} & \frac{\partial g_1(\boldsymbol{\beta})}{\partial \beta_3} \\ \frac{\partial g_2(\boldsymbol{\beta})}{\partial \beta_1} & \frac{\partial g_2(\boldsymbol{\beta})}{\partial \beta_2} & \frac{\partial g_2(\boldsymbol{\beta})}{\partial \beta_2} \end{pmatrix} \\ &= \begin{pmatrix} 0 & \frac{1}{1+\beta_2} & \frac{-\beta_1}{(1+\beta_2)^2} \\ 0 & 0 & \frac{-1}{(1+\beta_2)^2} \end{pmatrix}\end{aligned}$$

Delta Method: Some Failures

But we need to be careful. Suppose that $\theta \approx 0$ and

- $g(x) = |X|$
- $g(x) = 1/X$
- $g(x) = \sqrt{X}$

These situations can arise in practice when we have weak instruments or other problems.

Bootstrap I

- Bootstrap takes a different approach.
  - Instead of estimating $\hat{\theta}$ and then using a first-order Taylor Approximation...
  - What if we directly tried to construct the sampling distribution of $\hat{\theta}$?

Bootstrap II

- Data $(X_1, \ldots, X_n) \sim P$ are drawn from some measure P
  - We can form a nonparametric estimate $\hat{P}$ by just assuming that each X_i has weight $\frac{1}{n}$.
- We can then simulate a new sample $X^* = (X_1^*, \ldots X_n^*) \sim \hat{P}$.
  - Easy: we take our data and construct n observations by sampling with replacement
- Compute whatever statistic of X^*, $S(X^*)$ we would like.
  - Could be the OLS coefficients $\beta_1^*, \ldots, \beta_k^*$.
  - Or some function β_1^*/β_2^*.
  - Or something really complicated: estimate parameters of a game $\hat{\theta}^*$ and now find Nash Equilibrium of the game $S(X^*, \hat{\theta}^*)$ changes.
- Do this B times and calculate at $Var(S_b)$ or $CI(S_1, \ldots, S_b)$.

Bootstrap: OLS

- Linear predictor $\beta = E[X_i X_i']^{-1} E[X_i Y_i]$
- Sample $\{(Y_i, X_i\}_{i=1}^N$
- OLS: $\hat{\beta}_{ols} = [\sum_{j=1}^N X_j X_j']^{-1} [\sum_{j=1}^N X_j Y_j]$
- To get the variance, resample with replacement B times and calculate $\hat{\beta}_{ols,b} = [\sum_{j=1}^N X_j X_j']^{-1} [\sum_{j=1}^N X_j Y_j]$
- Now an estimate of the variance is just

$$V(\beta_{ols}) = \frac{1}{B} \sum_{b}^{B} (\hat{B}_{ols,b} - \bar{\beta}_{ols,b}) \cdot (\hat{B}_{ols,b} - \bar{\beta}_{ols,b})'$$

Bootstrap: Variants I

The bootstrap I have presented is sometimes known as the nonparametric bootstrap and is the most common one.

Parametric Bootstrap instead of bootstrapping pairs (Y_i, X_i), we can bootstrap the residuals.

- First estimated β by OLS
- Get residuals $\hat{\epsilon}_i$
- For $b = 1, ..., B$, resample N residuals $\hat{\epsilon}_i$
- Predict a new $Y_{b,j} = X_j'\hat{\beta}_{ols} + \hat{\epsilon}_{b,j}$
- Estimate with OLS and proceed as before.

Bootstrap: Variants II

Wild Bootstrap Similar to parametric bootstrap but we rescale ϵ_i to allow for heteroskedasticity

Block Bootstrap For correlated data (e.g.: time series). Blocks can be overlapping or not.

Bootstrap: Bias Correction

The main idea is that $\hat{\theta}^{1*}, \dots, \hat{\theta}^{B*}$ approximates the sampling distribution of $\hat{\theta}$. There are lots of things we can do now:

- We already saw how to calculate $Var(\hat{\theta}^{1*}, \dots, \hat{\theta}^{B*})$.

$$\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^*_{(b)} - \overline{\theta^*})^2$$

- Calculate $E(\hat{\theta}^*_{(1)}, \dots, \hat{\theta}^*_{(B)}) = \overline{\theta^*} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^*_{(b)}$.
  - We can use the estimated bias to bias correct our estimates

$$\begin{aligned} Bias(\hat{\theta}) &= E[\hat{\theta}] - \theta \\ Bias_{bs}(\hat{\theta}) &= \overline{\theta^*} - \hat{\theta} \end{aligned}$$

Recall $\theta = E[\hat{\theta}] - Bias[\hat{\theta}]$:

$$\hat{\theta} - Bias_{bs}(\hat{\theta}) = \hat{\theta} - (\overline{\theta^*} - \hat{\theta}) = 2\hat{\theta} - \overline{\theta^*}$$

- Correcting bias isn't for free - variance tradeoff!
- Linear models are (hopefully) unbiased, but most nonlinear models are consistent but biased.

Bootstrap: Confidence Intervals

There are actually three ways to construct bootstrap CI's:

1. Obvious way: sort $\hat{\theta}^*$ then take $CI : [\hat{\theta}^*_{\alpha/2}, \hat{\theta}^*_{1-\alpha/2}]$.
2. Asymptotic Normal: $CI : \hat{\theta} \pm 1.96\sqrt{V(\hat{\theta}^*)}$. (CLT).
3. Better Way: let $W = \hat{\theta} - \theta$. If we knew the distribution of W then: $Pr(w_{1-\alpha/2} \leq W \leq w_{\alpha/2})$:

$$CI : [\hat{\theta} - w_{1-\alpha/2}, \hat{\theta} - w_{\alpha/2}]$$

We can estimate with $W^* = \hat{\theta}^* - \hat{\theta}$.

$$CI : [\hat{\theta} - w^*_{1-\alpha/2}, \hat{\theta} - w^*_{\alpha/2}] = [2\hat{\theta} - \theta^*_{1-\alpha/2}, 2\hat{\theta} - \theta^*_{\alpha/2}]$$

Why is this preferred? Bias Correction!

Bootstrap: Why do people like it?

- Econometricians like the bootstrap because under certain conditions it is higher order efficient for the confidence interval construction (but not the standard errors).
    - Intuition: because it is non-parametric it is able to deal with more than just the first term in the Taylor Expansion (actually an Edgeworth Expansion).
    - Higher-order asymptotic theory is best left for real econometricians!
- Practitioner's like the bootstrap because it is easy.
    - If you can estimate your model once in a reasonable amount of time, then you can construct confidence intervals for most parameters and model predictions.

Bootstrap: When Does It Fail?

- Bootstrap isn't magic. If you are constructing standard errors for something that isn't asymptotically normal, don't expect it to work!
- The Bootstrap exploits the notion that your sample is IID (by sampling with replacement). If IID does not hold, the bootstrap may fail (but we can sometimes fix it!).
- Bootstrap depends on asymptotic theory. In small samples weird things can happen. We need $\hat{P}$ to be a good approximation to the true P (nothing missing).

Bootstrap vs Delta Method

- Delta Method works best when working out Jacobian $D(\theta)$ is easy and statistic is well approximated with a linear function (not too curvy).
- I would almost always advise Bootstrap unless:
  - Delta method is trivial e.g.: β_1/β_2 in linear regression.
  - Computing model takes many days so that 10,000 repetitions would be impossible.
- Worst case scenario: rent time on Amazon EC2!
  - I “bought” over $1,000 of standard errors recently.
- But neither is magic and both can fail!

Review: What was the point?

- OLS is lowest variance among linear unbiased estimators.
- But there are nonlinear estimators and potentially biased estimators.
    - Everything faces a bias-variance tradeoff.
    - Nearly anything can be written as Kernel.