Nonparametrics and Local Methods

Richard L. Sweeney

based on slides by Chris Conlon

Empirical Methods
Spring 2019

1. Nonparametric Density Estimation

2. Cross-Validation

3. Example: Auctions

4. Non-parametric Regression
    - Multivariate Kernels
    - Local linear

5. Semi-parametrics

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.

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

Some Common Kernels

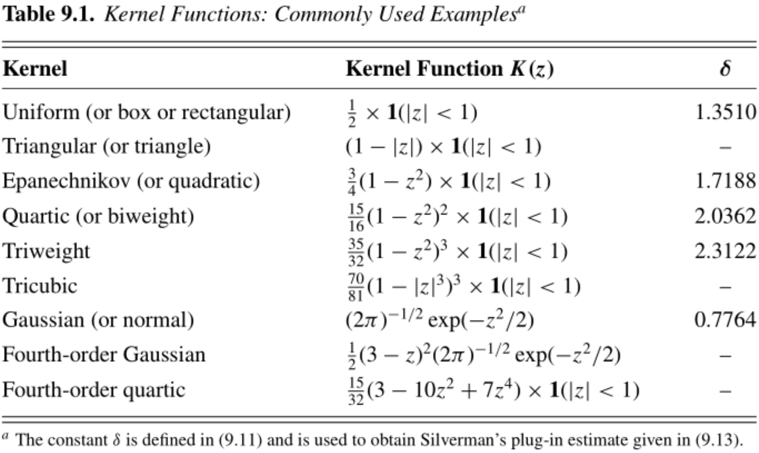


Table 9.1. *Kernel Functions: Commonly Used Examples*[a]

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

[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

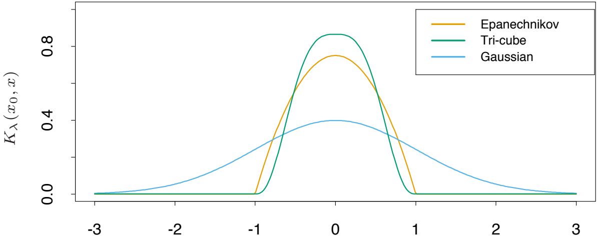


FIGURE 6.2. *A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to* 1*. The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov kernel has none. The Gaussian kernel is continuously differentiable,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

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

The **variance** of the estimator is

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

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

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

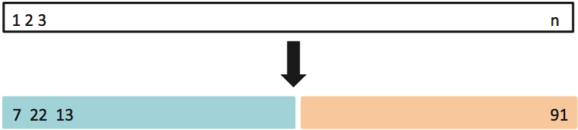


FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning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

Challenge with Sample Splitting

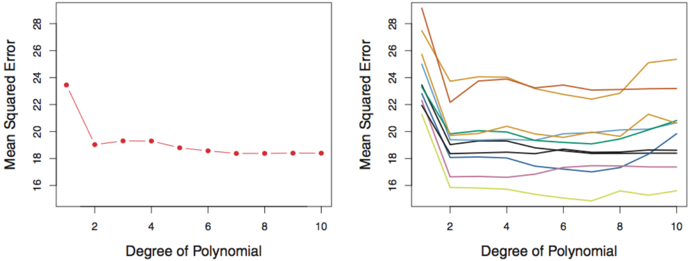


FIGURE 5.2. *The validation set approach was used on the* Auto *data set in order to estimate the test error that results from predicting* mpg *using polynomial functions of* horsepower. Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation 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

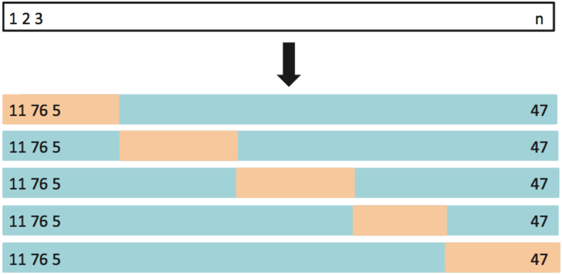


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE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

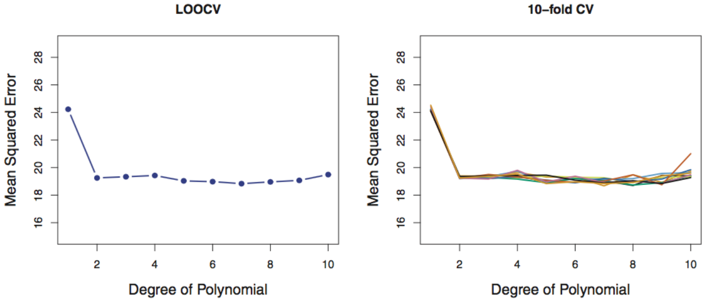


FIGURE 5.4. *Cross-validation was used on the* `Auto` *data set in order to estimate the test error that results from predicting* `mpg` *using polynomial functions of* `horsepower`. Left: *The LOOCV error curve.* Right: *10-fold CV was run nine separate times, each with a different random split of the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

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 → leave them to theorists.

Back to the CDF

Since now we have estimated the density with

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

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

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

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

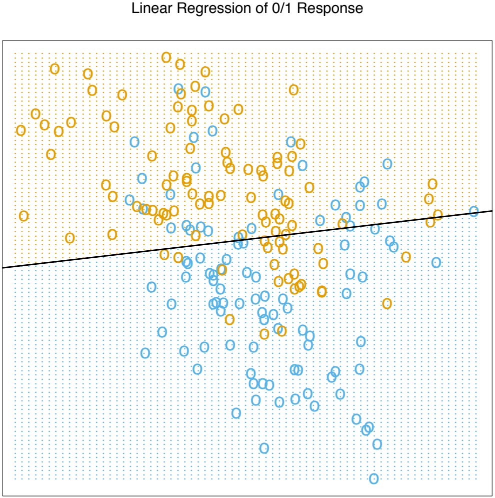


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE *$= 0$,* ORANGE *$= 1$), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified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

15-Nearest Neighbor Classifier

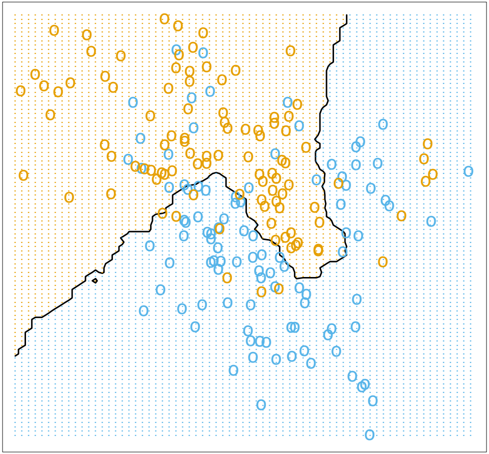


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Extreme: 1 Nearest Neighbor

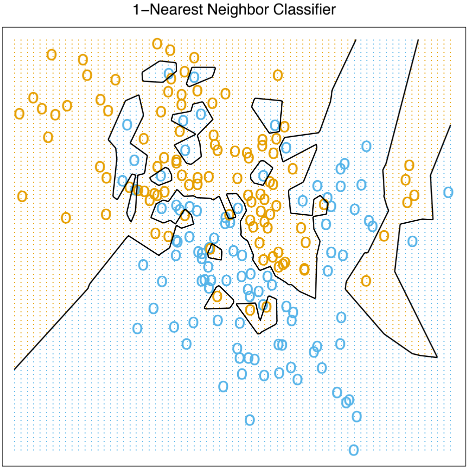


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by 1-nearest-neighbor classification.*

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

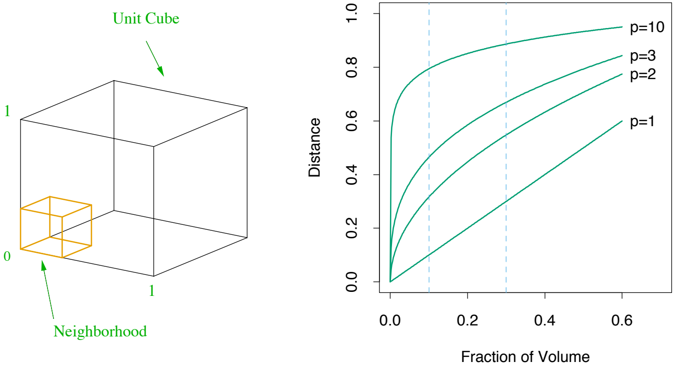


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 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 $\sum_i w_{i0,h} = 1$.

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 (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

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

Local Linear Regression II

Advantages:

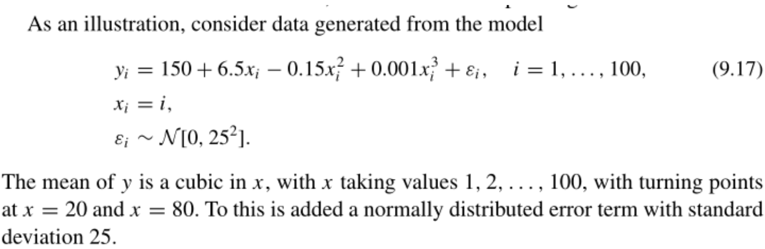
- the bias becomes 0 if the true $m(x)$ is linear.
- the coefficient of $(x - x_i)$ estimates $m'(x)$.
- behaves better in "almost empty" regions.

Disadvantages: hardly any, just do it! How?

Locally Weighted Scatterplot Smoothing (**lowess**)

- uses a variable bandwidth $h_{0,k}$
- tricubic kernel
- downweights observations with large residuals (through N iterations)
- Output: robust to outliers, smooth, and better at boundaries
- In most packages, but can be

Example from Cameron and Trivedi

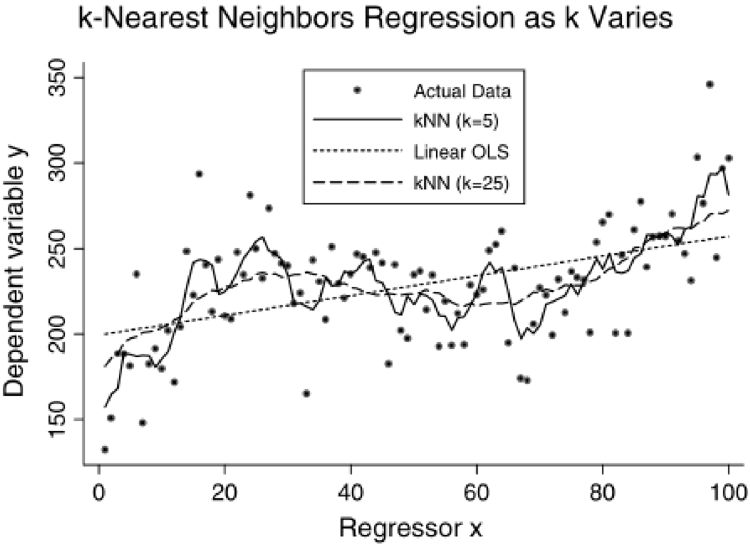


As an illustration, consider data generated from the model

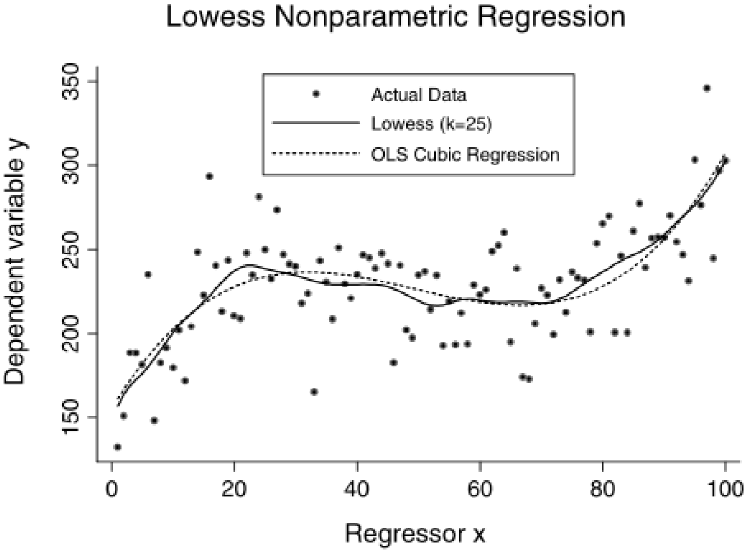
$$y_i = 150 + 6.5x_i - 0.15x_i^2 + 0.001x_i^3 + \varepsilon_i, \quad i = 1, \dots, 100, \qquad (9.17)$$
$$x_i = i,$$
$$\varepsilon_i \sim \mathcal{N}[0, 25^2].$$

The mean of y is a cubic in x, with x taking values $1, 2, \dots, 100$, 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
350
300
250
200
150
0
20
40
60
80
100
Regressor x

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 $K_h(x - x_i)$
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

- This is a *parametric* approximation that becomes more accurate as the sample size increases (series of sieves)
- **Idea:** we add regressors when we have more data, eventually providing an arbitrarily close approximation to the true regression function
- Polynomial Example: $m(x;\beta_K) = \sum_{j=1}^{K} \hat{\beta}_j x^j$
- Given K, **just estimate model using OLS**
- In practice, you'll want to choose K using leave-one-out cross validation.

Splines: trading off fit and smoothness

- Another option is to use splines
- **Idea:** Approximate regression function *locally* using (low order) polynomials, then connect these polynomials at knots.
- The regression function will be continuous at the knots, but not as many times differentiable as elsewhere
- For example, can estimate a linear spline with an intercept and separate slopes depending on whether $x > x_0$

Cubic Splines

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 $p_x > 1$.

Orthogonal polynomials: check out Chebyshev, $1, x, 2x^2 - 1, 4x^3 - 3x \ldots$ (on $[-1, 1]$ here.)

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.

Acknowledgements/ References