Nonparametric Methods: Part I

Abi Adams

Hilary 2019

Overview

- Introduction
  - Why NP methods useful?
  - Practicalities of Kernel Density & Regression Estimation
  - Impact of Bandwidth
  - Local Polynomial Regression

Overview

- Lecture 2: Nitty Gritty
  - Optimising Bandwidth
  - The Curse of Dimensionality
  - Semiparametric Methods: Partially Linear Model

Overview

- Lecture 3: Kernel Regression in the Context of RDD
  - RDD Recap
  - Local Polynomial Regression & RDD
  - Practical Considerations

Introduction

- Imagine you are interested in the relationship between y and $\mathbf{X}$.
- Theory may include/exclude variables, imply monotonicity or concavity, or optimising behaviour.
- It almost NEVER implies a functional form.
- Nonparametric methods offer flexible approaches to modelling in which the functional form of the object of interest is not pre-specified.

Introduction

- These techniques are particularly useful for data description and when checking the suitability of a particular parametric specification
- As we will see next week, we will typically need to combine nonparametric and parametric models for practically useful empirical work
- Many of these techniques can be easily applied within Stata
- (Although you should probably explore the use of Matlab or R if you want to take this seriously)

Introduction

- There are two ways of ‘doing’ nonparametric estimation
  - **Smoothing**: Kernel estimation
  - **Summing**: Splines and Sieves
- In this course we will focus on *smoothing* methods

Nonparametric Density Estimation

Density Estimation

- The parametric approach to density estimation assumes that the data are drawn from one of a known family of distributions, e.g. the normal distribution with mean μ and variance σ^2.
- Nonparametric estimation methods are less prescriptive
- They assume that the data *has* a density but beyond that the data are "allowed to speak for themselves" in determining the estimate to a much greater extent than the parametric method allows.
- They are also familiar.
- In fact you have been using them ever since you started to study statistics

The Histogram

- The most widely used density estimator is the *histogram*.
- You specify an origin (x_0) and a bin-width (h) - the latter mainly controls the number of bins/the smoothness of the estimator.
- Bins are intervals $[x_0 + mh, x_0 + (m+1)h)$ $m = 0, 1, 2, ...$
- Given an i.i.d. sample $X_1, ..., X_n$ the histogram is

$$\widehat{f}_n(x) = \frac{1}{nh} [\text{No. of observations in the same bin as } x]$$

The Histogram

Open Stata!

The Histogram

- In general....
  - under-smoothing produces a noisy, wiggly picture with many artificial and confusing modes,
  - over-smoothing hides modes and obscures the fine structure.
- The patterns in these figures reflect the most important issue for any nonparametric estimator, namely, how much to smooth the data.

The Naive Estimator

- From the definition of a probability density, if X has density $f(x)$, then

$$f(x) = \lim_{h \to 0} \frac{1}{2h} P(x - h < X < x + h)$$

The Naive Estimator

- The so called 'naive density estimator' mimics this.

$$\widehat{f}_n(x) = \frac{1}{2nh}\,[\text{No. of observations falling in } (x-h, x+h)]$$

or

$$\widehat{f}_n(x) = \frac{1}{2nh}\sum_{i=1}^{n} 1\left(|X_i - x| \leq h\right)$$

The Naive Estimator

- This can be written more succinctly (and more naturally as we shall see) as

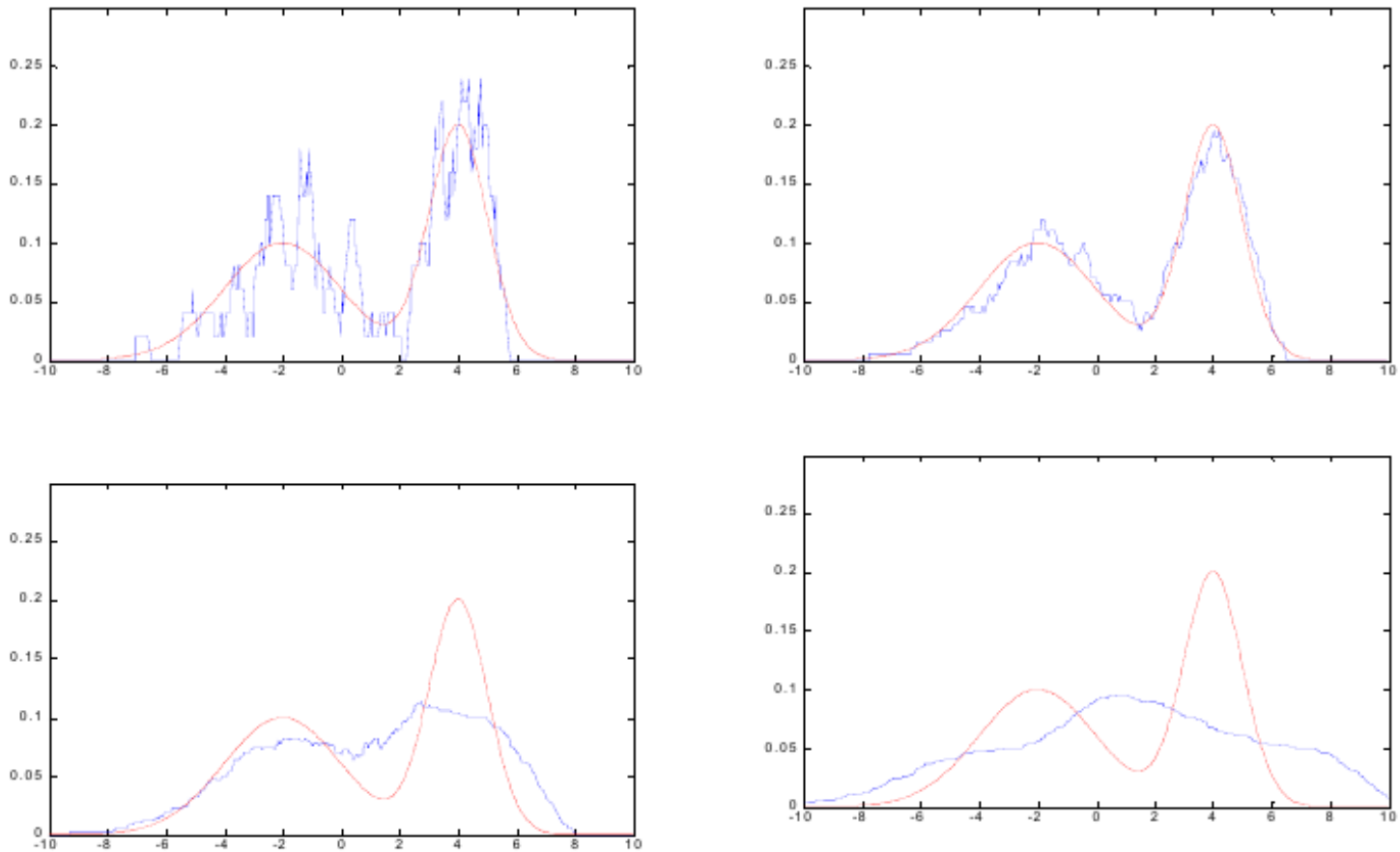
$$\widehat{f}_n(x) = \frac{1}{nh}\sum_{i=1}^{n} w\left(\frac{(X_i - x)}{h}\right)$$

where $w(.)$ is a weight function

$$w(x) = \frac{1}{2}1(|x| \leq 1)$$

The Naive Estimator

The Naive Estimator with $h = \{0.25, 1, 2.5, 5\}$



The Naive Estimator

- The fundamental smoothness versus retention trade-off is still evident
- 'Jumpy' because of the estimator is step-wise constant with jumps at points $X_i + h$
- The main differences from the histogram are:
  1. There is no origin
  2. The center of a bin is an observation

The Kernel Estimator

The Kernel Estimator

- The naive estimator is a particular example of a large class of estimators called *kernel* estimators.
- Kernel smoothing can be used for any statistical model (we're interested in density estimation at the moment).
- The naive estimator is an example but it has the drawback of being "steppy".
- But it's easy to generalise this estimator to avoid this problem.
- Namely, replace the rectangular weight function with a smooth weight function.....

The Kernel Estimator

- Kernel theory refers to the weight function as a "*kernel function*" and it is denoted

$$K(x)$$

- By assumption the kernel function is integrated to one

$$\int_{-\infty}^{\infty} K(x)\, dx = 1$$

- In fact any function which is smooth and integrates to one is a usable kernel, but a natural set of functions with just such properties are p.d.f.'s.

The Kernel Estimator

- Given some choice of kernel function, the kernel density estimator is defined by

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

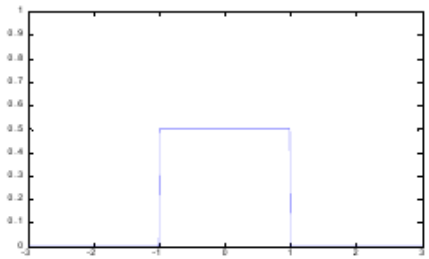
where h is referred to as the *bandwidth* or *smoothing parameter*.

Some Kernel Functions

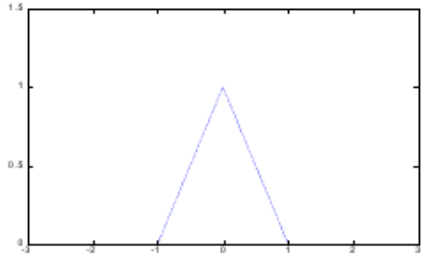
Kernel	$K(v)$	
Rectangular	$\frac{1}{2}$	for $\lvert v\rvert < 1$, 0 otherwise
Triangular	$1 - \lvert v\rvert$	for $\lvert v\rvert < 1$, 0 otherwise
Biwieght	$\frac{15}{16}\left(1 - v^2\right)^2$	for $\lvert v\rvert < 1$, 0 otherwise
Triweight	$\frac{35}{32}\left(1 - v^2\right)^3$	for $\lvert v\rvert < 1$, 0 otherwise
Epanechnikov	$\frac{3}{4}\left(1 - \frac{1}{5}v^2\right)5^{-0.5}$	for $\lvert v\rvert < 5^{0.5}$, 0 otherwise
Gaussian	$(2\pi)^{-1/2}\exp\left(-0.5v^2\right)$	

Some Kernel Functions

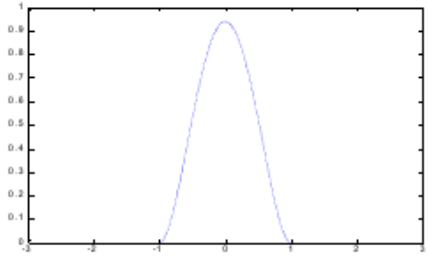
Rectangular



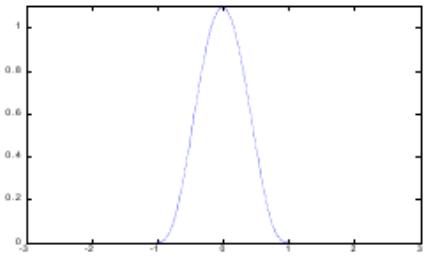
Triangular



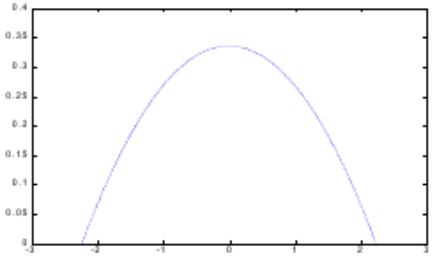
Biweight



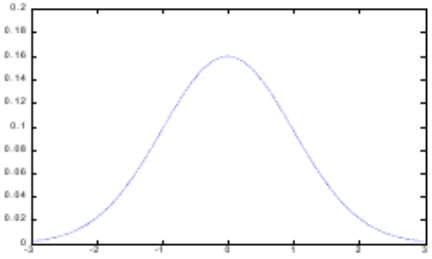
Biweight



Epanechnikov

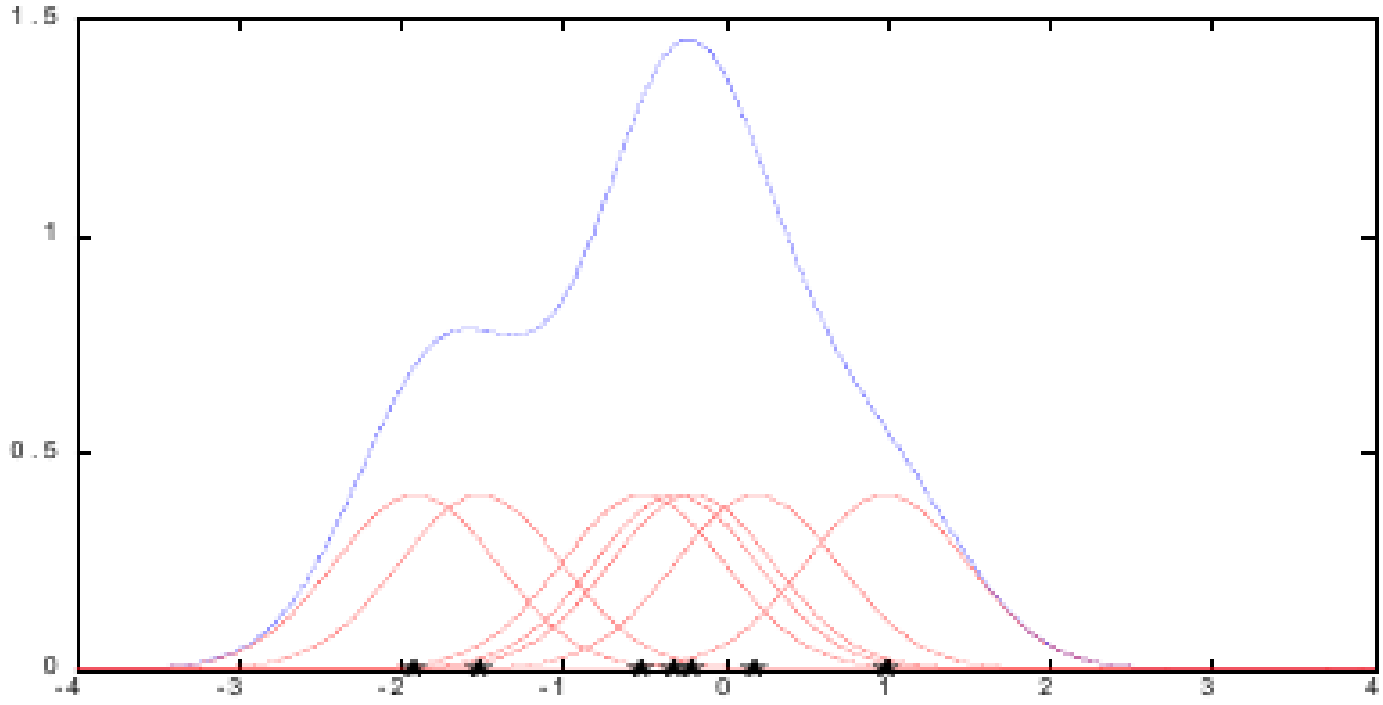


Gaussian



The Kernel Estimator

- Kernel estimators can be considered the ‘sum of bumps’



- Each kernel is centered at X_i and the estimator is constructed as the average of the kernel ordinates at that point

Open Stata!

Kernel Regression

Recap

- You normally think of estimating regression functions of the form:

$$Y_i = X_i\beta + e_i$$

- Nonparametric methods do not specify a functional form a priori

$$Y_i = m(X_i) + e_i$$

- This allows the data to 'speak for itself' but does not lend itself to giving one answer ('How does X affect y??)

Kernel Regression

- We form the regression function at the point x as a weighted sum of the y_i data, where the weights depend on the point of interest.

Kernel Regression

- Using

$$\widehat{m}(x) = \sum_{i=1}^{n} w_i(x)\, Y_i$$

and

$$w_i(x) = \frac{\frac{1}{nh} K\left(\frac{X_i - x}{h}\right)}{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

where

1. the numerator is the familar kernel weight function evaluated at x
2. the denominator is the kernel density estimate.

gives the Kernel Regression function estimator (Nadaraya (1964), Watson (1964)):

$$\widehat{m}(x) = \frac{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) Y_i}{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

Kernel Regression

- If the denominator of $w_i(x)$ is zero so is the numerator and the estimator is not defined - this happens (quite rightly) when the data is sparse/absent i.e. $\widehat{f}(x) = 0$.
- Bandwidth determines the degree of smoothness
- Choosing the bandwidth to trade off over- and under-smoothness is the key problem.

Kernel Regression

- If the Kernel estimator is evaulated at $\{X_i\}_{i=1,\ldots,n}$ then as $h \to 0$

$$\widehat{m}(X_i) \to Y_i$$

and as $h \to \infty$

$$\widehat{m}(X_i) \to \frac{1}{n}\sum Y_i$$

- So small bandwidth gives interpolation, and wide bandwidths gives you an oversmoothed curve.

Open Stata!

Picking the Bandwidth

Mean Squared Error

- Most widely used measure of performance is the MSE

$$\begin{aligned} MSE\left[\widehat{m}(X_i)\right] &= E\left[\left(\widehat{m}(X_i) - m(X_i)\right)^2\right] \\ &= E\left[\left(\widehat{m}(X_i) - m(X_i)\right)^2\right] + / - E(\widehat{m}(X_i))^2 \\ &= \left[E\left(\widehat{m}(X_i)\right) - m(X_i)\right]^2 + var\left(\widehat{m}(X_i)\right) \end{aligned}$$

- Incorporating values of Y_i for which $X_i \neq x$ into the weighted average introduces bias
- However, these points also reduce the variance of the estimator as we are averaging over more data

Mean Squared Error

- The optimal bandwidth minimises the mean integrated square error (global measure of accuracy)
- Plug-in approaches exist — given a kernel function and 'guesses' about the nature of the underlying function, gives you $h^\star$
- Another, more principled approach, is to use **cross-validation** which bases the choice of $h^\star$ directly on the data

Cross Validation

- (Almost) the empirical counterpart to the MISE
- In theory would want to pick $h^\star$ to minimise:

$$\sum_{i=1}^{N} \left(Y_i - \widehat{m}\left(X_i\right)\right)^2 \quad = \quad \sum_{i=1}^{N} e_i^2$$

- Why is doing this a not particularly sensible thing to do?

Cross Validation

- So pick h to minimise the square of 'leave-one-out' residuals

$$h^{\star}_{CV} = \arg\min_{h} \sum_{i=1}^{N} \left(Y_i - \widehat{m}_{\neg i}\left(X_i\right) \right)^2$$

subject to $h \geq 0$

- Luckily this is not quite as computationally intensive as it might first appear because you can show that:

$$Y_i - \widehat{m}_{\neg i}\left(X_i\right) = \frac{Y_i - \widehat{m}\left(X_i\right)}{1 - \frac{K_h(X_i - X_i)}{\sum_j K_h(X_j - X_i)}}$$

Local Polynomial Regression

Kernel Methods and Local Regression

The Nadaraya-Watson estimator can be seen as a special case of a larger class of kernel regression estimators.

Take the true regression function and do a Taylor series expansion for t in the neighbourhood of x

$$m(t) \approx m(x) + m'(x)(t-x) + ...m^{(p)}(x)(t-x)^p \frac{1}{p!}$$

This suggests a local polynomial in the neghbourhood of x where we include kernel weights into the minimisation problem

$$\min_{\beta_0,...,\beta_p} \sum \left[Y_i - \beta_0 - \beta_1 (X_i - x) - ... - \beta_p (X_i - x)^p\right]^2 \frac{1}{h} K\left(\frac{X_i - x}{h}\right)$$

The result is a locally weighted least squares estimator with weights given by the kernel.

$$
\begin{aligned}
\mathbf{X} &= \begin{bmatrix} 1 & X_1 - x & \cdots & (X_1 - x)^p \\ 1 & X_2 - x & \cdots & (X_2 - x)^p \\ \vdots & \vdots & & \vdots \\ 1 & X_n - x & & (X_n - x)^p \end{bmatrix} \\
W &= \begin{bmatrix} \frac{1}{h} K\left(\frac{X_1 - x}{h}\right) & 0 & \cdots & 0 \\ 0 & \frac{1}{h} K\left(\frac{X_2 - x}{h}\right) & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & & & \frac{1}{h} K\left(\frac{X_n - x}{h}\right) \end{bmatrix} \\
\mathbf{Y} &= \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}
\end{aligned}
$$

Then the $\widehat{\boldsymbol{\beta}}(x)$ which minimises this is (of course)

$$\widehat{\boldsymbol{\beta}}(x) = \left(\mathbf{X}'\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{W}\mathbf{Y}$$

Note that in contrast to OLS the parameters depend on x - they are *local*.

The local polynomial estimation of the regression function is then

$$\widehat{m}_p(x) = \beta_0$$

Obviously (perhaps) if $\boldsymbol{\beta} = \beta_0$ this reduces to a local constant which is the Nadaraya-Watson estimator

$$\widehat{m}_0(x) = \frac{\frac{1}{n}\sum \frac{1}{h}K\left(\frac{X_i - x}{h}\right)Y_i}{\frac{1}{n}\sum \frac{1}{h}K\left(\frac{X_i - x}{h}\right)}$$

Bottom line - this makes the ends of the regression more sensitive to boundaries - why?

Local Polynomials

For estimating local polynomials the order p is usually taken to be one (local linear) or or three (local cubic regression).

The local linear fit performs (asymptotically) better than the Nadaraya-Watson estimator (local constant). This holds generally.

As the Nadaraya-Watson estimator, the local polynomial estimator is a weighted (local) average of the response variables.

As for all other kernel methods the bandwidth determines the degree of smoothness.

An infinitely large h makes all weights equal, thus we obtain a parametric pth order polynomial fit in that case.