Kernel and Local Regression

MAST90083 Computational Statistics and Data Mining

Karim Seghouane

School of Mathematics & Statistics
The University of Melbourne

Outline

Introduction

Fitting a good linear model often involves considerable time to adequately model:

- Nonlinear dependencies
- Significant and insignificant variables
- Interactions between variables

Various methods have been proposed to overcome these limitations, among them slpine repression. Here we look at an alternative to linear and spline regression that overcomes the issue of nonlinearities.

Introduction

- We discuss an alternative regression technique for estimating a regression function $f(\mathbf{x})$ over a domain in $\mathbb{R}^p$

- The approximation is realized by fitting a simple model at each point $\mathbf{x}_i$, $i = 1, \ldots, n$
- At each point $\mathbf{x}_i$, the model makes use of the those training samples close to $\mathbf{x}_i$ producing a smooth estimation $\hat{f}(\mathbf{x})$ in $\mathbb{R}^p$

- The selection of the training samples is realized using a weighting function known as kernel $K_h(\mathbf{x}_i, \mathbf{x}_j)$

Introduction

- $K_h(\mathbf{x}_i, \mathbf{x}_j)$ assigns a weight to $\mathbf{x}_j$ based on its scaled distance to $\mathbf{x}_i$ where the scale is controlled by a parameter h

- The scale h controls the size of the effective neighborhood to use for estimation
- These methods differ by the shape of the kernel function and do not require training

- The only parameter that needs to be tuned using training samples is the width of the kernel h

Introduction

Kernel regression has been around since the 1960s, and is one of the most popular methods for "nonparametrically" fitting a model to data. We work here in regression context, but there exist extensions to classification models via logistic regression.

We will focus on the most popular kernel regression and local polynomial regression.

One Dimensional Kernel

- Consider the regression model

$$y_i = f(x_i) + \epsilon_i, \quad E(\epsilon_i) = 0$$

- and we are interested in estimating the regression function

$$f(x) = E(y|x)$$

- using a training set (x_i, y_i), $i = 1, ..., n$.
- The relationship between x and y is more likely to be nonlinear

One Dimensional Kernel

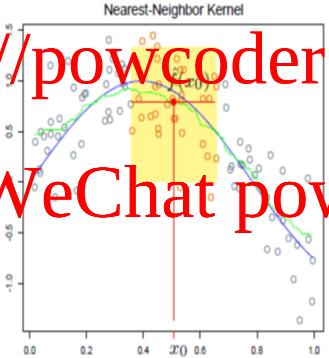
- A direct method: k-nearest-neighbor average. Use the average of those observations in the defined neighborhood of x, $N_k(x)$ to build the estimator of $f(x)$

$$\hat{f}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i = \mathrm{Ave}(y_i | x_i \in N_k(x))$$

- $N_k(x)$ defines the k closest points x_i to x in the training sample to use or select for the estimation

One Dimensional Kernel

- The average changes in a discrete way, leading to a discontinuous $\hat{f}(x)$



One Dimensional Kernel

- Problem: The k-nearest-neighbor estimator gives the same weight to all the points in the neighbor used for the estimation of $\hat{f}(x)$

- Alternative: Make the weights attributed to the points used in the estimation inversely proportional (smoothly) to the distance from the point of estimation interest

Nadaraya-Watson Kernel

The Nadaraya-Watson kernel leads to a weighted average estimation

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_h(x_0, x_i)\, y_i}{\sum_{i=1}^{N} K_h(x_0, x_i)}$$

$$\text{if} \quad \sum_{i=1}^{N} K_h(x_0, x_i) \neq 0$$

$$\text{and} \quad \hat{f}(x_0) = 0 \quad \text{if} \quad \sum_{i=1}^{N} K_h(x_0, x_i) = 0$$

Kernel Function

- The Kernel function plays a central role in the fitting and it is defined by

$$K_h(x_0, x) = K\left(\frac{x - x_0}{h}\right)$$

- $K(x)$ needs to be smooth, maximal at 0, symmetrical around 0 and decreasing with respect to $|x|$

- Having

$$\int K(u)du = 1 \quad \int uK(u)du = 0$$

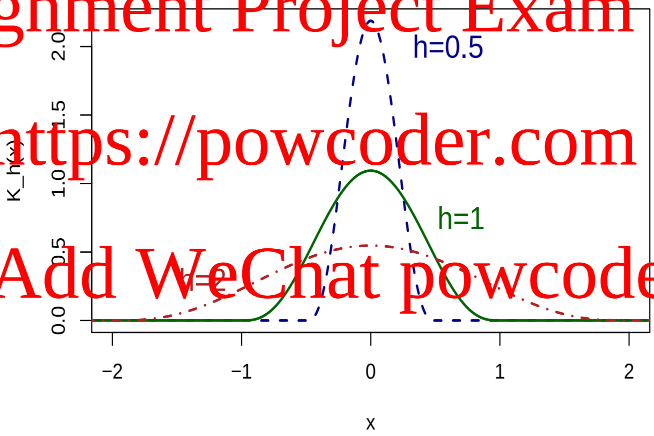
- is also common

Kernel Functions

Common kernel functions are

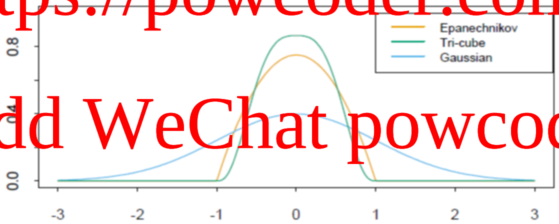
Name	$K(x)$	Support
Epanechnikov	$\frac{3}{4}(1-x^2)\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1,1]$
Gaussian	$(2\pi)^{-1/2}\exp\left\{\frac{-x^2}{2}\right\}$	$[-\infty,\infty]$
Biweight	$\frac{15}{16}(1-x^2)^2\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1,1]$
Triweight	$\frac{35}{32}(1-x^2)^3\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1,1]$
Uniform	$\frac{1}{2}\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1,1]$
Tricube	$\frac{70}{81}(1-\lvert x\rvert^3)^3\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1,1]$

Triweight kernel $K_h(x)$ for various choices of h



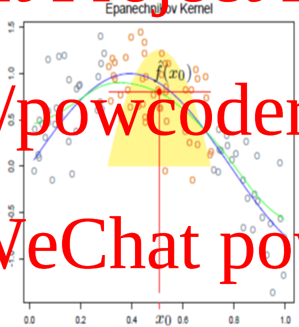
Kernel Functions

- The Gaussian function is non-compact kernel where σ^2 plays the role of the window size



Nadaraya-Watson Kernel

Epanechnikov quadratic kernel application example



The contribution of the points (their weights in the estimation) slowly increases as the approximation evolves. The contribution is initially with weight zero.

Example

- The nearest-neighbor corresponds to

$$K(x) = \frac{1}{2}\mathrm{I}\{|x| < 1\}$$

- In this case $\hat{f}(x)$ =average of y_i's such that $x_i \in [x-h, x+h]$ or $|x_i - x| \leq h$

Example

There are two extreme cases

- $h \to \infty$, $\hat{f}$ is independent of x (high bias case)

$$\hat{f}(x) \to \frac{1}{N}\sum_{i=1}^{N} y_i = \text{const}$$

- $h \to 0$, $h < \min_{i,j} |x_i - x_j|$, (high variance case)

$$\hat{f}(x_i) = y_i \quad \text{and} \quad \hat{f}(x) = 0 \quad \text{for} \quad x \neq x_i$$

- The estimator reproduces the data y_i at x_i and zero in other points.
- The optimal h is between these two extremes and provides the appropriate compromise between the bias and variance

Linear Estimator

- The Nadaraya-Watson can be written as a weighted sum

$$\hat{f}(x) = \sum_{i=1}^{N} y_i W_i(x)$$

- where the weights

$$W_i(x) = \frac{K_h(x, x_i)}{\sum_{i=1}^{N} K_h(x, x_i)} \mathrm{I}\left(\sum_{i=1}^{N} K_h(x, x_i) \neq 0\right)$$

- are independent of the responses y_i

Justification or Interpretation

Let (x, y) be a pair of random variables in $\mathbb{R}^2$ with density $p(x, y)$ and marginal density $p(x) = \int p(x, y)dy > 0$, then

$$f(x) = E[y/x] = \int yp(y/x)dy = \int y\frac{p(y,x)}{p(x)}dy$$

$$= \frac{1}{p(x)}\int yp(y,x)dy = \frac{\int yp(x,y)dy}{\int p(x,y)dy}$$

If we replace $p(x, y)$ by $\hat{p}(x, y)$ (its estimator) and $p(x)$ by $\hat{p}(x)$ we recover $\hat{f}(x)$

Note 1

Justification or Interpretation

If the density p is assumed uniform then

$$\hat{f}(x) = \sum_{i=1}^{N} y_i K\left(\frac{x_i - x}{h}\right)$$

Properties

- The width of the used local neighborhood h plays the role of the smoothing parameter
- Large values of h implies lower variance (use more samples for estimation) but higher bias (assume the function is constant within the window)

- For k-nearest neighborhoods, the neighborhood size k plays the role of the window size h and $h_k(x_i) = |x_i - x_k|$ where x_k is the k^{th} closest x_i to x_i

- Adaptive width $h(x)$ can also be considered instead of constant width $h(x) = h$ and the kernel is

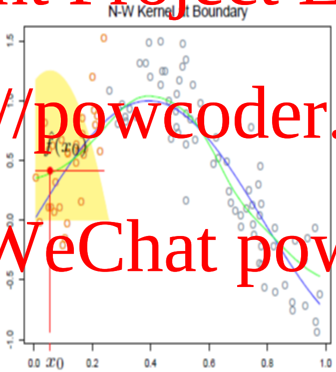
$$K_h(x_i, x) = K\left(\frac{|x - x_i|}{h(x_i)}\right)$$

Local Polynomial Regression

- Kernel fit can still have problems due to the asymmetry at the boundaries

- or in the interior if the x values are not equally spaced
- Locally weighted linear regression provide an alternative local approximation

Local Polynomial Regression



Local Linear Regression

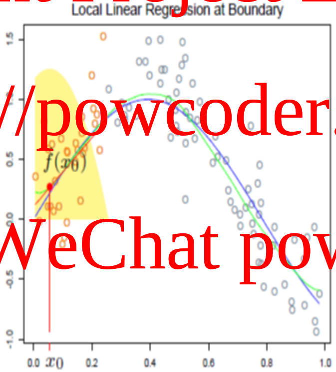
- It is obtained by solving a weighted least squares criterion at each target points x_0

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_h(x_0, x_i)\left[y_i - \alpha(x_0) - \beta(x_0)x_i\right]^2$$

- and the estimate at x_0 is given by

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)\, x_0$$

Local Polynomial Regression



Local Linear Regression

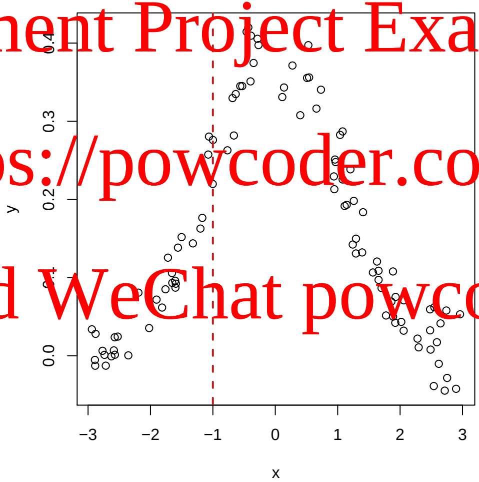
Let B be the $N \times 2$ regression matrix with row $b(x_i)^\top = (1, x_i)$ and $W(x_0)$ the $N \times N$ diagonal matrix with i^{th} diagonal element $K_h(x_0, x_i)$ then

$$\hat{f}(x_0) = b(x_0)^\top \left(B^\top W(x_0) B\right)^{-1} B^\top W(x_0)\mathbf{y} = \sum_{i=1}^{N} \ell_i(x_0)\, y_i$$

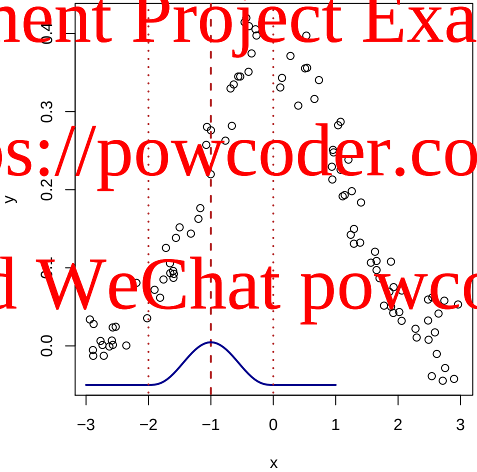
where $\ell_i(x_0)'$s do not involve $\mathbf{y}$

Local linear regression tends to be biased in curved regions of the true function

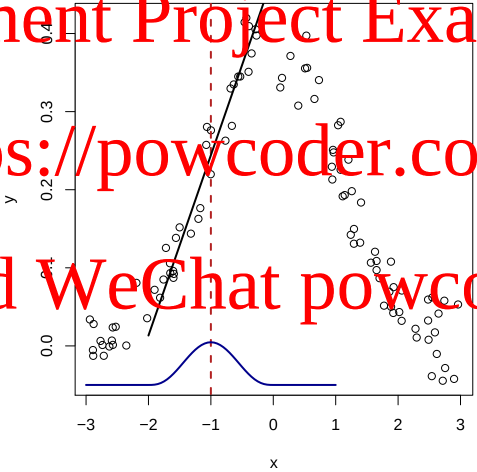
An example of local linear regression



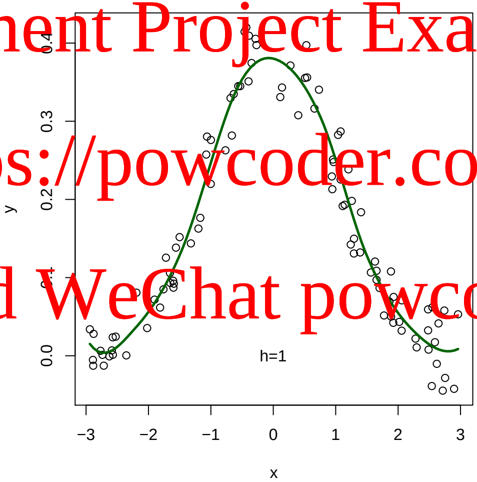
An example of local linear regression



An example of local linear regression



An example of local linear regression

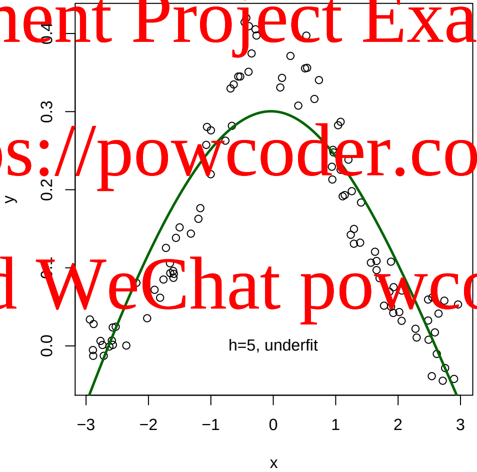


Overfitting and underfitting

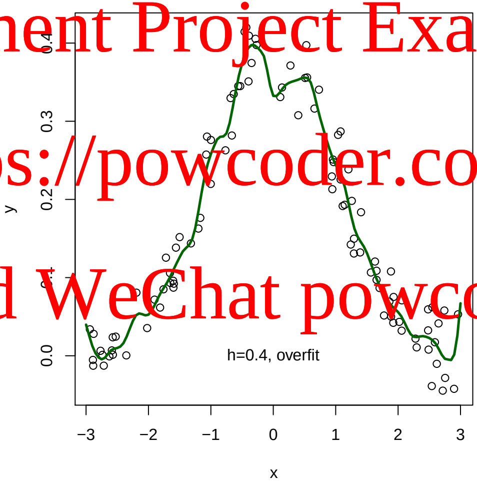
The choice of bandwidth h very directly controls the bias-variance tradeoff. Choosing h too small will tend to give overfitted models (high variance, low bias), while h too large will give underfit models (high bias, low variance).

In practice we can employ methods like cross-validation, or even plug-in estimates to decide on an appropriate value of h.

Underfitted local linear regression



Overfitted local linear regression



Adaptive choices of h

A common alternative to using a fixed h is to vary it with respect to x. The most common example of this is the nearest neighbour bandwidth, where h_x is chosen so that the window always contains a fixed proportion of the data t,

$$t = \frac{\sum_i \mathbb{I}\{|X_i - x| \le h_x\}}{n}.$$

Local Polynomial Regression

- Local polynomial regression are generally able to correct this bias
- In this case we fit a local polynomial

$$\min_{\alpha(x_0),\beta_j(x_0)} \sum_{i=1}^{N} K_h(x_0, x_i) \left[y_i - \alpha(x_0) - \sum_{j=1}^{d} \beta_j(x_0) x_i^j \right]^2$$

- with fit

$$\hat{f}(x_0) = \alpha(x_0) + \sum_{j=1}^{d} \beta_j(x_0) x_0^j$$

Local Polynomial Regression

- The reduction in bias generates an increase in variance
- The bias-variance tradeoff is controlled by the polynomial degree d

Local Constant Regression

A special kernel regression smoother — the local constant regression smoother, which minimises

$$\sum_{i=1}^{n} (Y_i - \alpha_x)^2 K_h(x - X_i),$$

can be found to be

$$\hat{\alpha}_x = \left[\sum_{i=1}^{n} K_h(x - X_i) \right]^{-1} \sum_{i=1}^{n} Y_i K_h(x - X_i)$$

The aim

- We are interested in a flexible model to predict y using multiple predictors, say $x_1, \cdots, x_p$.
- LS: $f(x_1, \cdots, x_p) = \alpha + \beta_1 x_1 + \cdots + \beta_p x_p$
- GAM: $f(x_1, \cdots, x_p) = \alpha + f_1(x_1) + \cdots + f_p(x_p)$, where each $f_j(x_j)$ is a smoothing spline function of x_j.
- GAM is an additive model of many functions that depend on a single predictor. (Although, you can create a 'new' predictor $x_k x_l$ and add a function $f_{p+1}(x_k x_l)$, but this quickly leads to an over-fit model)

GAMs

$f_i(x_i)$ is a building block and can take many forms. For example

- Smoothing spline (the most popular)
- Natural spline
- Local regression
- Polynomial regression

GAMs

In the regression context, the model is fit by minimising

$$\sum_{i=1}^{n}\left\{y_i-\alpha-\sum_{j=1}^{p}f_j(x_{ij})\right\}^2+\sum_{j=1}^{p}\lambda_j\int f_j''(t_j)^2\,dt_j,$$

which involves parameter estimation.

In the fitting procedure, each λ_j is a constant that control the degree of smoothing (determined by the degrees of freedom specified for f_j).

Each f_j is then estimated by estimating the associated regression coefficient parameters as for the smoothing spline fit.

GAMs

Estimating all the f_j's simultaneously is difficult. The **backfitting algorithm** is an iterative solution to this, which fits each f_j in turn and iteratively:

1. Initialize $\hat{\alpha} = \bar{y}$, all $\hat{f}_j = 0$.
2. For $j = 1, \dots, p$,

$$\hat{f}_j \leftarrow \text{Smooth fit using } \{x_{ij}\}_{i=1}^n \text{ for } \left\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \right\}_{i=1}^n$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{n} \sum_{i=1}^n \hat{f}_j(x_{ij}) \quad (\text{so } \textstyle\sum_{i=1}^n \hat{f}(x_{ij}) = 0 \text{ is assured})$$

3. Repeat step 2 until convergence (each $\hat{f}_j$ changes less than some threshold)