Kernel smoothing methods

Chmiela Bartosz

University of Wrocław

28.03.2022

Contents

Introdution

- Class of regression techniques,
- flexible in estimating the regression function $f(X)$,
- fit simple model separately at each point,
- use only only observations close to the point,
- estimated function $\hat{f}(X)$ is smooth,
- weighting function (*kernel*) $K_\lambda(x_0, x_i)$,
- little or no training needed.

One-dimensional kernel smoothers

Let $x_i \in \mathbb{R}^p$ be the training sample and $y_i \in \mathbb{R}$ response associated with it.

k-nearest-neighbor average:

$$\hat{f}(x) = \operatorname{Ave}\left(y_i | x_i \in N_k(x)\right),$$

as and estimate of the regression function $E(Y|X = x)$, where $N_k(x)$ is the set of k points nearest to x in squared distance.

One-dimensional kernel smoothers

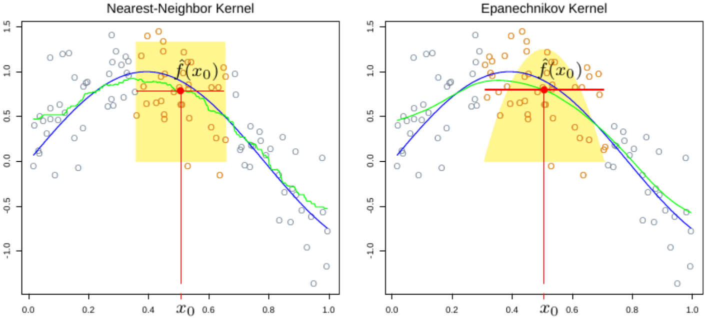

Figure: pairs x_i, y_i are generated at random from the blue curve with Gaussian errors: $Y = \sin(4X) + \varepsilon$, $X \sim U[0,1], \varepsilon \sim N(0, 1/3)$. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.1*.

One-dimensional kernel smoothers

Nadaraya-Watson kernel-weighted average

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

with the Epanechnikov quadratic kernel

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right), \quad D(t) = \begin{cases} \frac{3}{4}(1 - t^2), & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

One-dimensional kernel smoothers

In general, we can use a width function $h_\lambda(x_0)$:

- k-nearest-neighbors: $h_\lambda(x_0) = |x_0 - x_{[k]}|$ where $x_{[k]}$ is the kth closest x_i to x_0,
- Nadaraya-Watson: $h_\lambda(x_0) = \lambda$,

then we have

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right).$$

One-dimensional kernel smoothers

In practice one has to attend to:

- The smoothing parameter λ, which determines the width of the local neighborhood,
- metric window widths,
- issues with ties in x_i,
- boundary issues.

Local linear regression

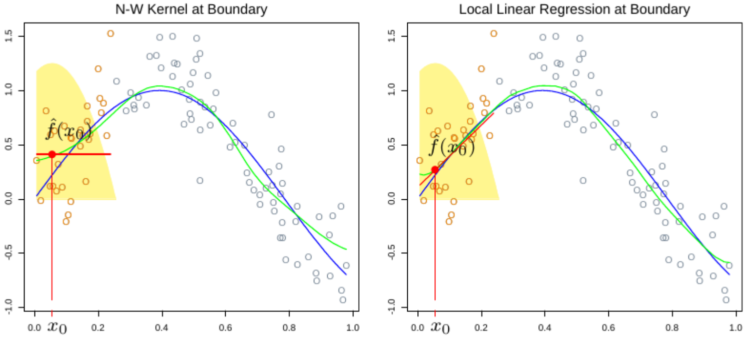

Figure: The locally weighted average has bias problems at or near the boundaries of the domain. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.3*.

Local linear regression

Locally weighted regression solves a separate weighted least squares problem at each target point x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2.$$

The estimate is then:

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

Local linear regression

Let $b(x)^T = (1, x)$, $\mathbf{B} \in \mathbb{R}^{N \times 2}$ with ith row $b(x_i)^T$ and $\mathbf{W}(x_0) \in \mathbb{R}^{N \times N}$ diagonal matrix with ith diagonal element $K_\lambda(x_0, x_i)$, then

$$\begin{aligned}\hat{f}(x_0) &= b(x_0)^T \left(\mathbf{B}^T \mathbf{W}(x_0) \mathbf{B}\right)^{-1} \mathbf{B}^T \mathbf{W}(x_0) \mathbf{y} \\ &= \sum_{i=1}^{N} l_i(x_0) y_i.\end{aligned}$$

These weights $l_i(x_0)$ combine the weighting kernel $K_\lambda(x_0, x_i)$ and the least squares operations, and are sometimes referred to as the *equivalent kernel*.

Local linear regression

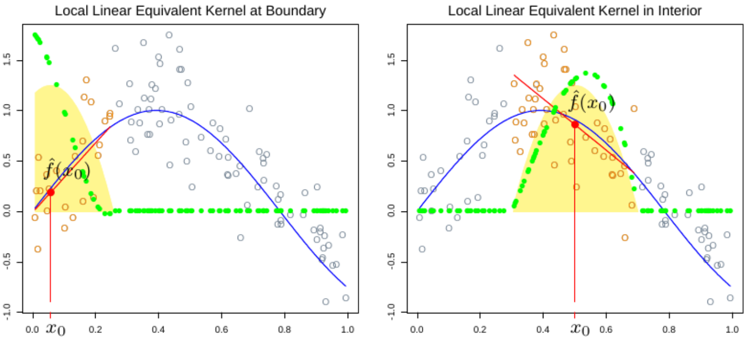

Figure: The green points show the equivalent kernel $l_i(x_0)$ for local regression. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.4*.

Local polynomial regression

We can fit local polynomial fits of any degree d,

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

with solution:

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

Local polynomial regression

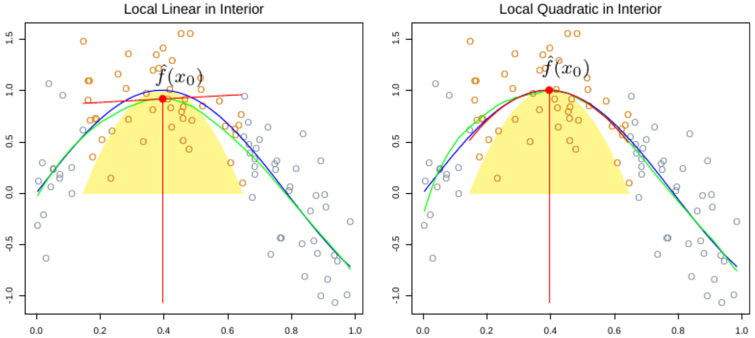

Figure: Local linear fits exhibit bias in regions of curvature of the true function. Local quadratic fits tend to eliminate this bias. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.5*.

Local polynomial regression

Assuming the model:

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

with ε_i i.i.d with mean 0 and variance σ^2, then

$$\mathrm{Var}(\hat{f}(x_0)) = \sigma^2 \|l(x_0)\|^2,$$

where $l(x_0)$ is the vector of equivalent kernel weights at x_0. It can be shown that $\|l(x_0)\|$ increases with d and so there is a bias–variance tradeoff in selecting the polynomial degree.

Local polynomial regression

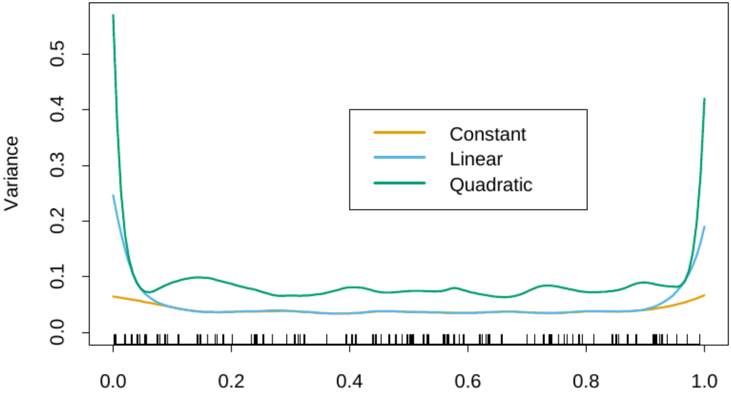

Figure: The variances functions $\|l(x_0)\|^2$ for local constant, linear and quadratic regression, for a metric bandwidth ($\lambda = 0.2$) tri-cube kernel. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.6*.

Local polynomial regression

- Local linear fits can help bias dramatically at the boundaries at a modest cost in variance,
- Local quadratic fits do little at the boundaries for bias, but increase the variance a lot,
- Local quadratic fits tend to be most helpful in reducing bias due to curvature in the interior of the domain.

Local regression in $\mathbb{R}^p$

Let $b(X)$ be a vector of polynomial terms in X of maximum degree d. For example:

- with $d = 0$ we get $b(X) = 1$,
- with $d = 1$ and $p = 2$ we get $b(X) = (1, X_1, X_2)$,
- with $d = 2$ we get $b(X) = (1, X_1, X_2, X_1^2, X_2^2)$.

Local regression in $\mathbb{R}^p$

At each $x_0 \in \mathbb{R}^p$ solve

$$\min_{\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \left(y_i - b(x_i)^T \beta(x_0)\right)^2$$

to produce fit

$$\hat{f}(x_0) = b(x_i)^T \hat{\beta}(x_0).$$

Local regression in $\mathbb{R}^p$

Typically the kernel will be a radial function, such as the radial Epanechnikov or tri-cube kernel with Euclidean norm.

$$K_\lambda(x_0, x) = D\left(\frac{\|x - x_0\|}{\lambda}\right).$$

Since the Euclidean norm depends on the units in each coordinate, it makes most sense to standardize each predictor.

Local regression in $\mathbb{R}^p$

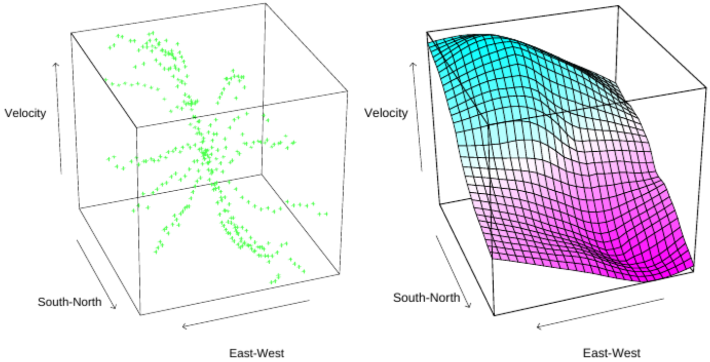

Figure: The left panel shows three-dimensional data, where the response is the velocity measurements on a galaxy, and the two predictors record positions on the celestial sphere. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.8*.

Structured local regression models in $\mathbb{R}^p$

A more general approach is to use a positive semidefinite matrix $\mathbf{A}$ to weigh the different coordinates:

$$K_{\lambda,\mathbf{A}} = D\left(\frac{(x-x_0)^T\mathbf{A}(x-x_0)}{\lambda}\right).$$

Structured local regression models in $\mathbb{R}^p$

We are trying to fit a regression function $E(Y|X) = f(X_1, X_2, \ldots, X_p)$ in $\mathbb{R}^p$, in which every level of interaction is potentially present. It is natural to consider (ANOVA) decompositions of the form

$$f(X_1, X_2, \ldots, X_p) = \alpha + \sum_j g_j(X_j) + \sum_{k<l} g_{kl}(X_k, X_l) + \cdots$$

and then introduce structure by eliminating some of the higher-order terms.

Structured local regression models in $\mathbb{R}^p$

We divide the p predictors in X into a set $X_1, \ldots, X_q$ with $q < p$, and the remainder of the variables we collect in the vector Z. Then assume the conditionally linear model

$$f(X) = \alpha(Z) + \beta_1(Z)X_1 + \cdots + \beta_q(Z)X_q.$$

For given Z, this is a linear model, but each of the coefficients can vary with Z.

Structured local regression models in $\mathbb{R}^p$

Fit such a model by locally weighted least squares:

$$\min_{\alpha(z_0),\beta(z_0)} \sum_{i=1}^{N} K_\lambda(z_0, z_i)[y_i - \alpha(z_0) - x_{1i}\beta_1(z_0) - \cdots + x_{qi}\beta_q(z_0)]^2.$$

Structured local regression models in $\mathbb{R}^p$

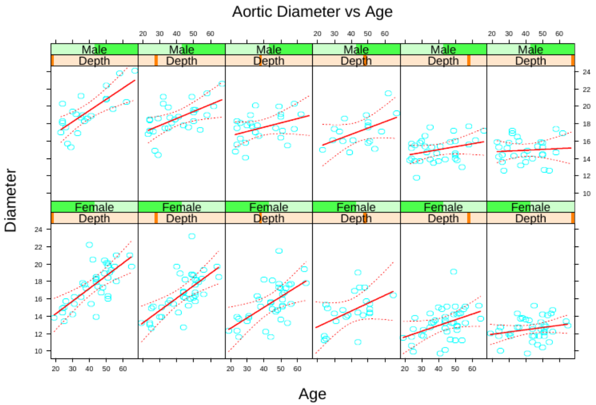

Figure: In each panel the aorta diameter is modeled as a linear function of age. The coefficients of this model vary with gender and depth down the aorta. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.10*.

Selecting the width of the kernel

In each of the kernels K_λ, λ is a parameter that controls its width:

- For the Epanechnikov or tri-cube kernel with metric width, λ is the radius of the support region,
- for the Gaussian kernel, λ is the standard deviation.
- λ is the number k of nearest neighbors in k-nearest neighborhoods, often expressed as a fraction or span k/N of the total training sample.

Selecting the width of the kernel

There is a natural bias-variance tradeoff as we change the width of the averaging window, which is most explicit for local averages:

- If the window is narrow, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0 , and its variance will be relatively large close to that of an individual y_i,
- if the window is wide, the variance of $\hat{f}(x_0)$ will be small relative to the variance of any y_i, because of the effects of averaging.

Kernel density estimation

Suppose we have a random sample $x_1, \dots, x_N$ drawn from probability density $f_X(x)$ and we wish to estimate $f_X(x_0)$.. A natural local estimate has the form:

$$\hat{f}(x_0) = \frac{\# x_i \in \mathcal{N}(x_0)}{N\lambda},$$

where $\mathcal{N}(x_0)$ is a small metric neighborhood around x_0 of width λ. This estimate is "bumpy" so the smooth *Parzen* estimate is preferred

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

Kernel density estimation

With Gaussian kernel

$$K_\lambda(x_0, x) = \phi\left(\frac{|x - x_0|}{\lambda}\right) = \phi_\lambda(|x - x_0|),$$

the *Parzen* estimate has form

$$\hat{f}(x_0) = \frac{1}{N}\sum_{i=1}^{N} \phi_\lambda(|x - x_0|) = \left(\hat{F} * \phi_\lambda\right)(x).$$

This is the convolution of the sample empirical distribution $\hat{F}$ with ϕ_λ.

Kernel density estimation

In $\mathbb{R}^p$ the natural generalization of the Gaussian density estimate amounts to using the Gaussian product kernel in

$$\hat{f}(x_0) = \frac{1}{N(2\lambda^2\pi)^{\frac{p}{2}}} \sum_{i=1}^{N} \exp\left(-\frac{1}{2}(\|x_i - x_0\|/\lambda)\right)^2 .$$

Kernel density estimation

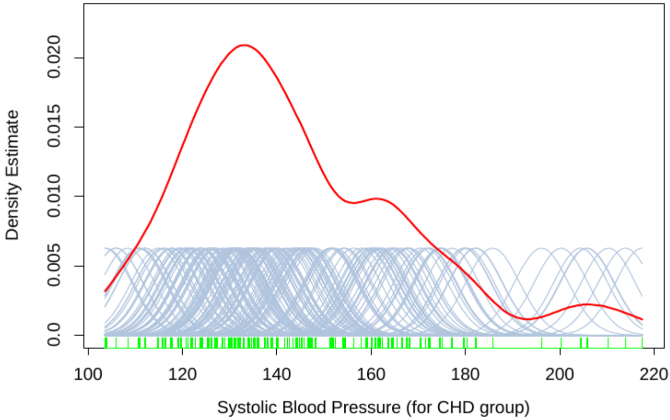

Figure: A kernel density estimate for systolic blood pressure.The density estimate at each point is the average contribution from each of the kernels at that point. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.13*.

Kernel density classification

Suppose for a J class problem we fit nonparametric density estimates $\hat{f}_j(X), j = 1, \dots, J$, separately in each of the classes, and we also have estimates of the class priors $\hat{\pi}_j$ (usually the sample proportions). Then

$$\hat{P}\left(G = j | X = x_0\right) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_{k=1}^{J} \hat{\pi}_k \hat{f}_k(x_0)}.$$

Kernel density classification

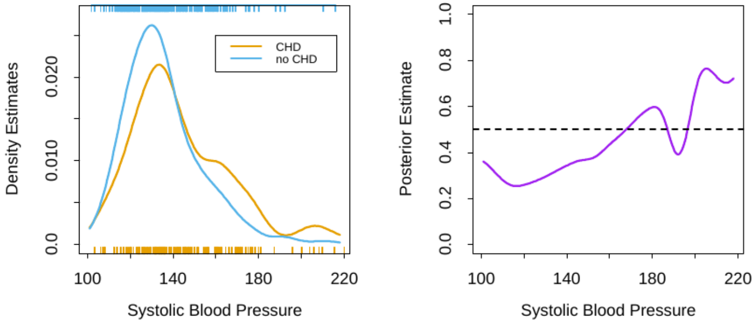

Figure: The left panel shows the two separate density estimates for systolic blood pressure in the CHD versus no-CHD groups, using a Gaussian kernel density estimate in each. The right panel shows the estimated posterior probabilities for CHD. Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.14*.

Kernel density classification

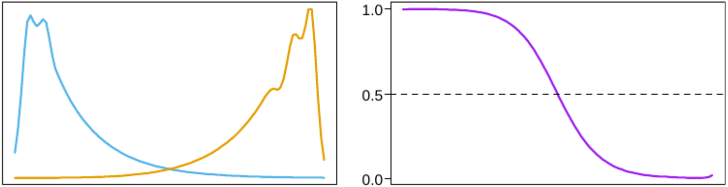

Figure: The population class densities may have interesting structure (left) that disappears when the posterior probabilities are formed (right). Source: *The elements of statistical learning, T Hastie, R Tibshirani, JH Friedman, fig. 6.15*.

Kernel density classification

We need only to estimate the posterior well near the decision boundary, for two classes, this is the set

$$\left\{ x | P(G = 1 | X = x) = \frac{1}{2} \right\}.$$

Naive Bayes classifier

The naive Bayes model assumes that given a class $G = j$, the features X_k are independent:

$$f_j(X) = \prod_{k=1}^{p} f_{jk}(X_k).$$

While this assumption is generally not true, it does simplify the estimation dramatically.

Naive Bayes classifier

- The individual class-conditional marginal densities f_{jk} can each be estimated separately using one-dimensional kernel density estimates,
- if a component X_j of X is discrete, then an appropriate histogram estimate can be used.

Computational considerations

- *memory-based* methods,
- fitting is done at evaluation or prediction time,
- for many real-time applications, this can make this class of methods infeasible.

Computational considerations

- cost to fit single observation x_0 is $O(N)$,
- the smoothing parameter λ for kernel methods are typically determined using cross-validation, at a cost of $O(N^2)$,
- implementations of local regression, such as the loess function in R use triangulation schemes to reduce the computations,
- it computes the fit exactly at M carefully chosen locations at a cost of $O(NM)$,
- then use blending techniques to interpolate the fit elsewhere ($O(M)$ per evaluation).

Thank you for your attention.