

STAT 542: Statistical Learning

Kernel Density and Regression Estimators

Ruoqing Zhu, Ph.D. <rqzhu@illinois.edu>

Course Website: <https://teazrq.github.io/stat542/>

February 25, 2022

Department of Statistics
University of Illinois at Urbana-Champaign

- We will introduce a new technique for nonparametric estimations
- An important concept is the **kernel function**
- We will use this in two different ways
 - Kernel density estimation
 - Kernel regression
- The central idea is to estimate things **locally**, but the bias-variance trade-off also applies to this problem

Kernel Density Estimation

Kernel Density Estimation

- Given some observations from an unknown distribution, we want to estimate the probability density function (pdf)
- This is an unsupervised problem, however, the technique can be used for regression later
- Suppose we have

$$X_1, \dots, X_n \stackrel{\text{i.i.d}}{\sim} f(\cdot)$$

- Some popular methods
 - Assume a family of distributions (e.g. Gaussian) $f_\theta(\cdot)$ and estimate the parameters
 - Kernel density estimator

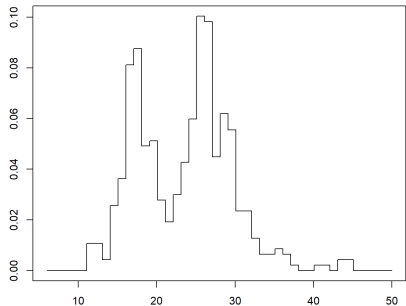
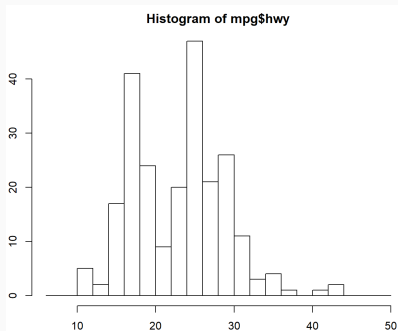
Histogram Estimator of Density Functions

- If a continuous variable $X \sim f(\cdot)$ the following are some facts:
 - $f(u) > 0$ and $\int f(u)du = 1$
 - $P(x - \frac{\lambda}{2} \leq X \leq x + \frac{\lambda}{2}) = \int_{x-\lambda/2}^{x+\lambda/2} f(u)du$
 - $f(x) = \lim_{\lambda \rightarrow 0} \frac{1}{\lambda} P(x - \lambda/2 \leq X \leq x + \lambda/2)$
- Histogram is a commonly used technique to visualize the data
- A similar and intuitive approach estimator is

$$\hat{f}(x) = \sum_{i=1}^n \frac{\mathbb{1}\{x_i \in [x - \lambda/2, x + \lambda/2]\}}{\lambda n}$$

- However, this estimation is bumpy and non-smooth

Kernel Density Estimation



Histogram Estimator of Density Functions

- Let's the previous estimator in a different way
 - Suppose we put $\frac{1}{n}$ point mass for each observation x_i
 - Further spread that probability onto a region with width λ , e.g., uniformly on $[x_i - \lambda/2, x_i + \lambda/2]$ for all i
 - Add up all such density functions.
- This is exactly the previous estimator (switch x and x_i). For any target point x , the estimator will be affected only by observations within $\lambda/2$

$$\hat{f}(x) = \sum_{i=1}^n \frac{\mathbb{1}\{x \in \overbrace{[x_i - \lambda/2, x_i + \lambda/2]}^{\text{uniform density range}}\}}{\underbrace{\lambda n}_{\text{normalizing constant}}}$$

- What if we use a different distribution, other than uniform?

Kernel Functions

- Denote K a kernel function, centered at 0
- Usually, we use a density function $K(\cdot)$ with
 - $\int K(u)du = 1$
 - $K(-u) = K(u)$
 - $\int K(u)u^2 du \leq \infty$
- Furthermore, we introduce a bandwidth λ that controls how “local” this estimator is $K_\lambda(u) = K(u/\lambda)/\lambda$
- In our previous uniform example, $K(u) = \mathbb{1}\{u \in [-\frac{1}{2}, \frac{1}{2}]\}$ and

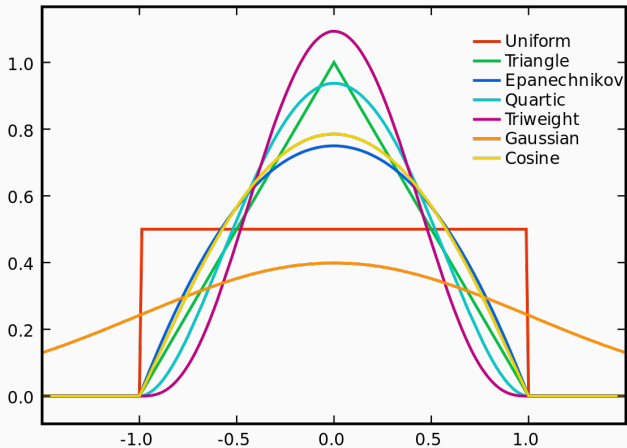
$$K_\lambda(u) = \frac{1}{\lambda} \mathbb{1}\{u \in [-\lambda/2, \lambda/2]\}$$

- Symmetric Beta family kernel

$$K(u, d) = \frac{(1 - u^2)^d}{2^{2d+1} B(d + 1, d + 1)} \mathbf{1}\{|u| \leq 1\}$$

- Uniform kernel $d = 0$
- Epanechnikov kernel $d = 1$
- Bi/Tri weight $d = 2, 3$
- Tri-cube kernel: $K(u) = (1 - u^3)^3 \mathbf{1}\{|u| \leq 1\}$
- Gaussian kernel: $K(u) = \phi(u) = 1/\sqrt{2\pi} \exp(-u^2/2)$

Kernels



- Parzen estimator

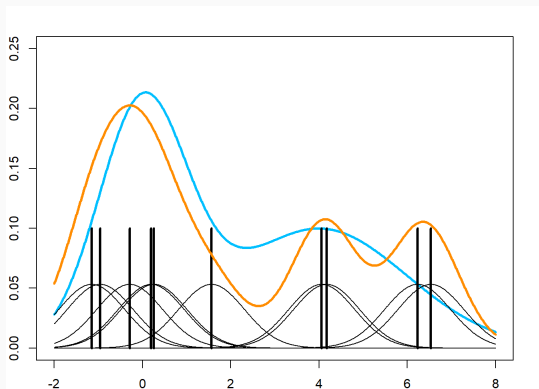
$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n K_{\lambda}(x, x_i)$$

where $K_{\lambda}(x, x_i) = K\left(\frac{|x-x_i|}{\lambda}\right)/\lambda$

- Popular choice: $K(u)$ is the **Gaussian density** function with mean zero and standard deviation λ

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{\lambda\sqrt{2\pi}} \exp\left\{-\frac{(x-x_i)^2}{2\lambda^2}\right\}$$

Kernel Density Estimation



Properties of the Kernel Density Estimator

- Under regularity assumptions, the kernel density estimator is **asymptotically unbiased** for a target point x

$$\begin{aligned} E[\hat{f}(x)] &= E\left[K\left(\frac{x-x_1}{\lambda}\right) / \lambda\right] \\ &= \int_{-\infty}^{\infty} \frac{1}{\lambda} K\left(\frac{x-x_1}{\lambda}\right) f(x_1) dx_1 \\ &= \int_{-\infty}^{\infty} \frac{1}{\lambda} K(t) f(x-t\lambda) d(x-t\lambda) \\ \text{(Taylor expansion)} \quad &= f(x) + \frac{\lambda^2}{2} f''(x) \int_{-\infty}^{\infty} K(t) t^2 dt + o(\lambda^2) \\ &\rightarrow f(x) \quad \text{as } \lambda \rightarrow 0 \end{aligned}$$

Properties of the Kernel Density Estimator

- We can further verify that the **integrated** Bias² (integrating the Bias² previously over the domain of x) is

$$\begin{aligned}\text{Bias}^2 &= \int \left(E[\hat{f}(x)] - f(x) \right)^2 dx \\ &\approx \frac{\lambda^4 \sigma_K^4}{4} \int [f''(x)]^2 dx\end{aligned}$$

where $\sigma_K^2 = \int_{-\infty}^{\infty} K(t)t^2 dt$.

Properties of the Kernel Density Estimator

- ... and the **integrated** variance is

$$\text{Var} \approx \frac{1}{n\lambda} \int K^2(t) dt$$

- Hence, the optimal bias-variance trade-off happens at where the asymptotic mean integrated squared error (AMISE)

$$\text{Bias}^2 + \text{Var}$$

is minimized

- This leads to $\lambda \approx \left(\frac{\int K^2(t) dt}{n\sigma_K^4 \int [f''(x)]^2 dx} \right)^{\frac{1}{5}}$
- This leads to an optimal convergence rate of $\mathcal{O}(n^{-4/5})$

- Epanechnikov kernel minimizes (the constant terms in) AMISE and is therefore optimal.
- Kernel efficiency is measured in comparison to Epanechnikov kernel:
 - Biweight 0.994; Triangular 0.986; Normal 0.951; Uniform 0.930
- However, choosing kernel is not as important as choosing the bandwidth!

- The rule of thumb (Silverman 1986) for the bandwidth λ in univariate case is

$$\hat{\lambda} = 1.06\hat{\sigma}n^{-1/5}$$

where $\hat{\sigma}$ is the standard deviation of the observed data

- This is based on assuming that both f and K are Gaussian
- For multidimensional case (p cannot be too large), the optimal λ is at the rate of $n^{-1/(p+4)}$.

- `hist` makes histograms
- `density` for kernel density estimator
- `bw.nrd` and a set of related functions for bandwidth selection
- Library `locfit`: function `locfit` can perform both local polynomial regressions and density estimation

Kernel regression

- k -Nearest Neighbor averaging

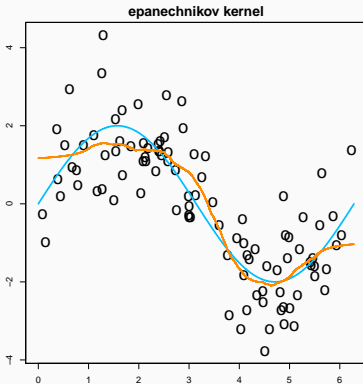
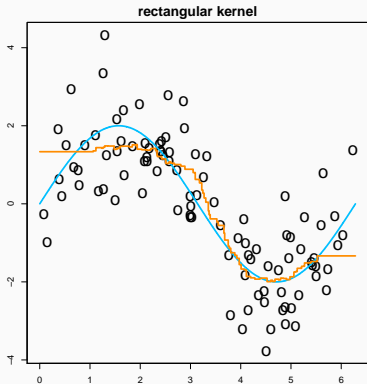
$$\hat{f}(x) = \sum_{i=1}^n w(x, x_i) y_i$$

where

$$w(x, x_i) = \begin{cases} \frac{1}{k} & \text{if } x_i \in N_k(x) \\ 0 & \text{o.w.} \end{cases}$$

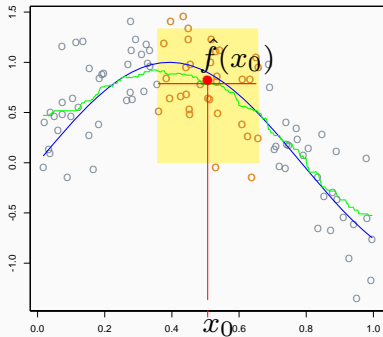
- The weight $w(x, x_i)$ drops off abruptly to zero outside the neighborhood of x . This accounts for jagged appearance of the fit.

Epanechnikov vs. Rectangular Kernels

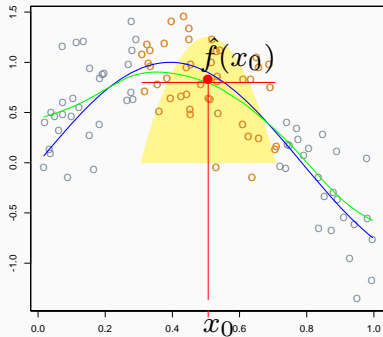


Epanechnikov vs. Rectangular Kernels

Nearest-Neighbor Kernel



Epanechnikov Kernel



Kernel Smoother (Univariate)

- We can still use the local averaging idea:
 - Fit a simple model locally at each point x using only those observations close to it
 - Localization via the weighting function $K(x, x_i)$, the weight of x_i is based on its distance from x
 - Weight decreases as we move further away from x
- For any point $x \in \mathcal{X}$,

$$\hat{f}(x) = \frac{\sum_i K_\lambda(x, x_i) y_i}{\sum_i K_\lambda(x, x_i)}$$

where $K_\lambda(x, x_i) = K(|x - x_i|/\lambda)/\lambda$ is a kernel function and λ is some tuning parameter for bandwidth.

- The estimator is called **Nadaraya-Watson** kernel estimator
- It shares the same intuition as the kernel density estimator, but with y_i as the outcome instead of $1/n$ as the point mass
- Requires little or no training time; **all the work gets done** during prediction (same as k NN).

- The bandwidth λ again controls how “local” the estimator is
- In many kernels (except Gaussian), only points within $[x - \lambda, x + \lambda]$ receive positive weights
- **Small λ** : rougher estimate, bias \downarrow , variance \uparrow
- **Large λ** : smoother estimate, bias \uparrow , variance \downarrow
- Although there are some theoretical values (e.g. Fan and Gijbels (1992, 1995)) for the optimal λ , most of may fail in practice (violation of assumptions)
- We can choosing λ using CV in practice

Drawbacks of Local Averaging

- The kernel averaging formulation can be badly **biased on the boundaries** of the domain due to the asymmetry of the kernel in that region (we already seen this)
- Locally weighted linear regression can make a first order correction (straight lines vs. constants)
- Minimizing the objective function

$$\underset{\beta_0(x), \beta_1(x)}{\text{minimize}} \sum_{i=1}^n K_\lambda(x, x_i) [y_i - \beta_0(x) - \beta_1(x)x_i]^2$$

- The estimation is extremely simple
- The solution $\hat{f}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0$ is evaluated only at x_0
- Correct the boundary bias of the kernel estimator

Drawbacks of Local Averaging

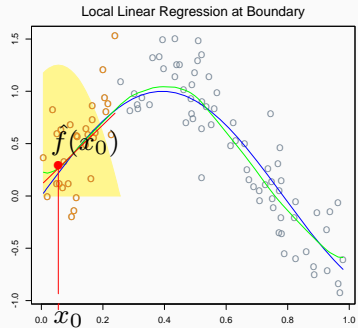
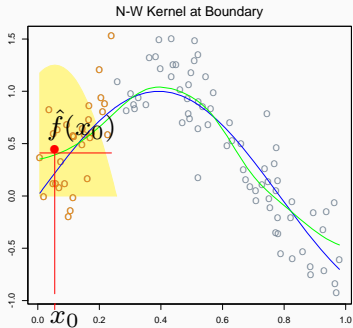
- The objective function can still be solved similarly as a linear regression
- If we let $\mathbf{W}(x_0) = \text{diag}(K(x_0, x_1), K(x_0, x_1), \dots, K(x_0, x_n))$
- Then the objective function (for target point x_0) is

$$\ell(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{W}(x_0) (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

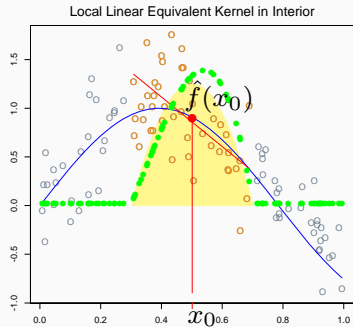
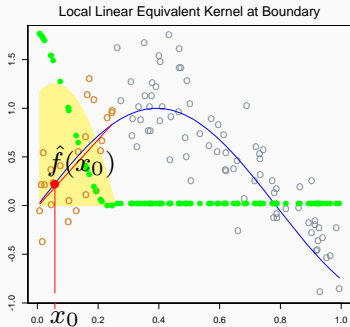
- The solution is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}$$

Kernel Boundary Bias



Kernel Boundary Bias



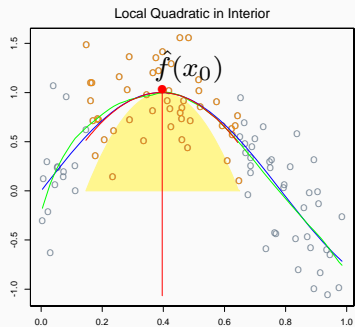
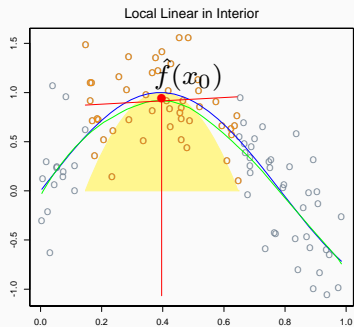
Local Polynomial Regression

- Locally weighted d polynomial regression
- Minimizing the objective function

$$\underset{\beta_0(x), \beta_r(x)}{\text{minimize}} \sum_{i=1}^n K_\lambda(x, x_i) \left[y_i - \beta_0(x) - \sum_{r=1}^d \beta_r(x) x_i^r \right]^2$$

- Still a weighed linear regression problem at each target point x_0
- $\hat{f}(x_0) = \hat{\beta}_0(x_0) + \sum_{r=1}^d \hat{\beta}_r(x_0) x_0^r$
- Correct the boundary bias of the kernel estimator
- Reduce bias in regions of curvature, however, at a price of higher variance

Kernel Boundary Bias



R implementation

- R function `loess` provides fitting of the local polynomial regressions
- The most important parameter `span` = α controls the degree of smoothing: only αn number of closest points are used based on the distance $|x - x_i|$, forming the neighborhood “ $N(x)$ ”
- A weighted least-square linear regression is fit within the neighborhood
- The weights uses tri-cube kernel: $w_{x,i} = (1 - u^3)^3$ with

$$u_i = \frac{|x_i - x|}{\max_{N(x)} |x_j - x|}$$

- `degree` specifies the degree of the polynomial
- Other implementations such as `locfit` and `locpoly` (use Gaussian kernel)