Statistical Machine learning

Kernel methods

April 3, 2019

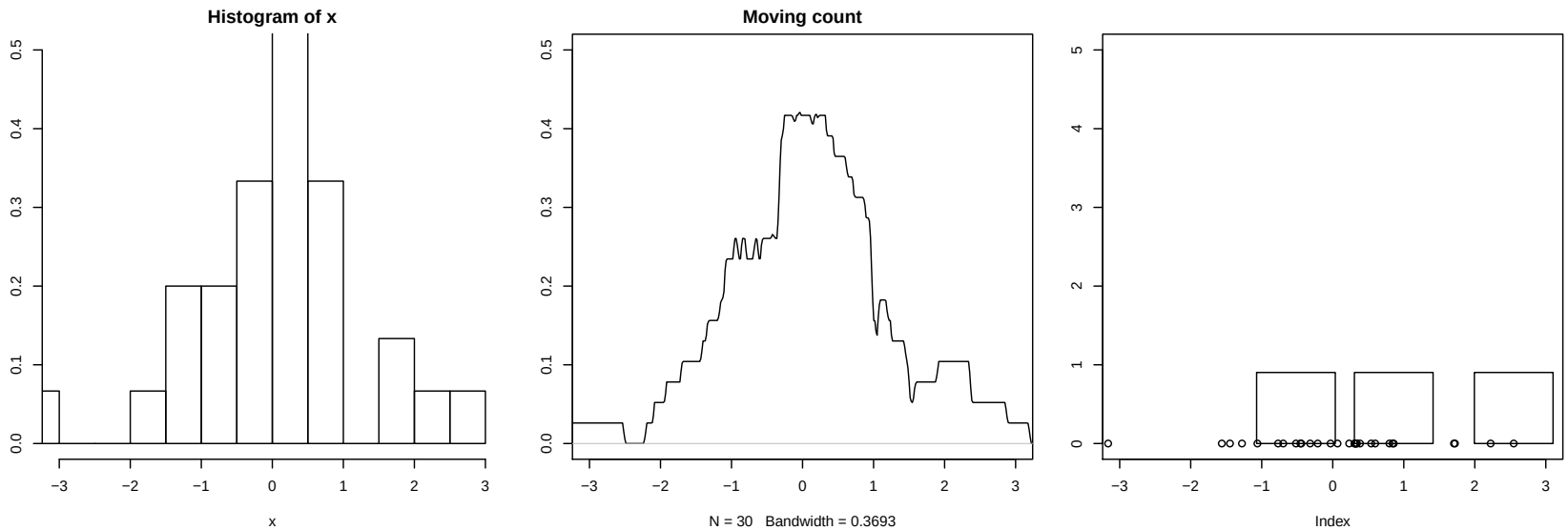
Histogram

- Suppose we have $x_1, \ldots, x_n$ univariate continuous observations
- The simplest form of non-parametric density estimation is the histogram
- The range of x is partitioned into intervals of length h: $C_1, \ldots, C_M$
- $$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} I(x_i \in C_j), \quad \text{for } x \in C_j$$
- Depending on the locations of the intervals C_j.

Better than histogram

- The rectangular estimator

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

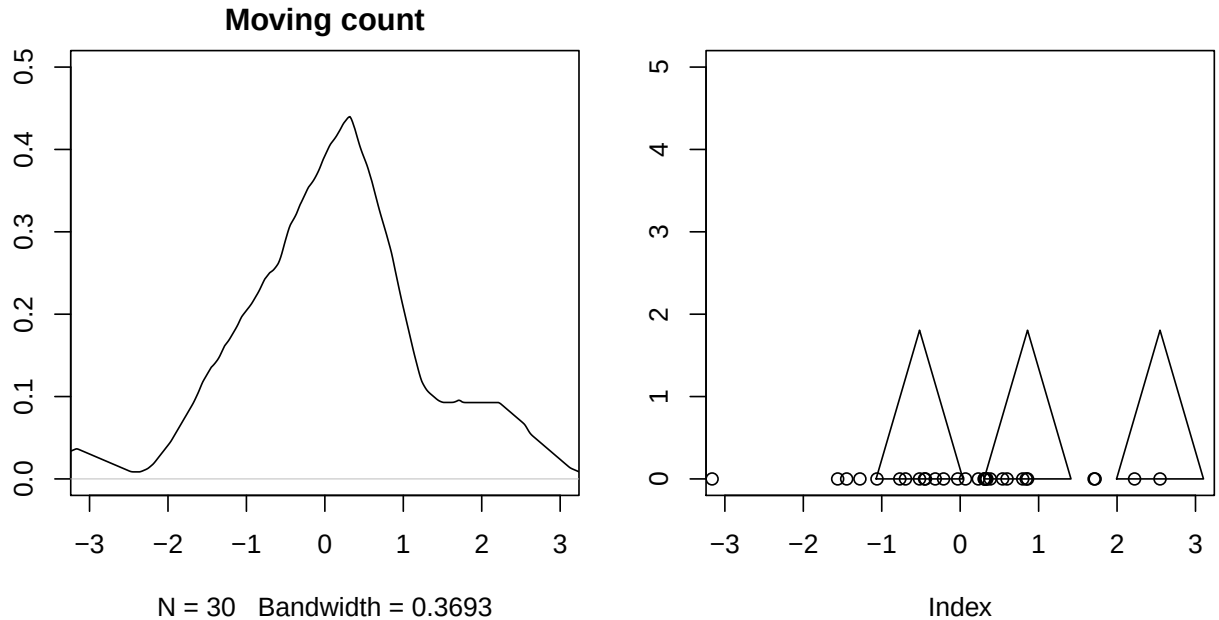
- 

  Left: histogram; Middle: rectangular estimator; Right: three specific rectangles

- Perhaps we can do better with continuous function

Triangular density estimator

- If the indicator functions are replaced by triangular functions, we have much smooth density estimate.



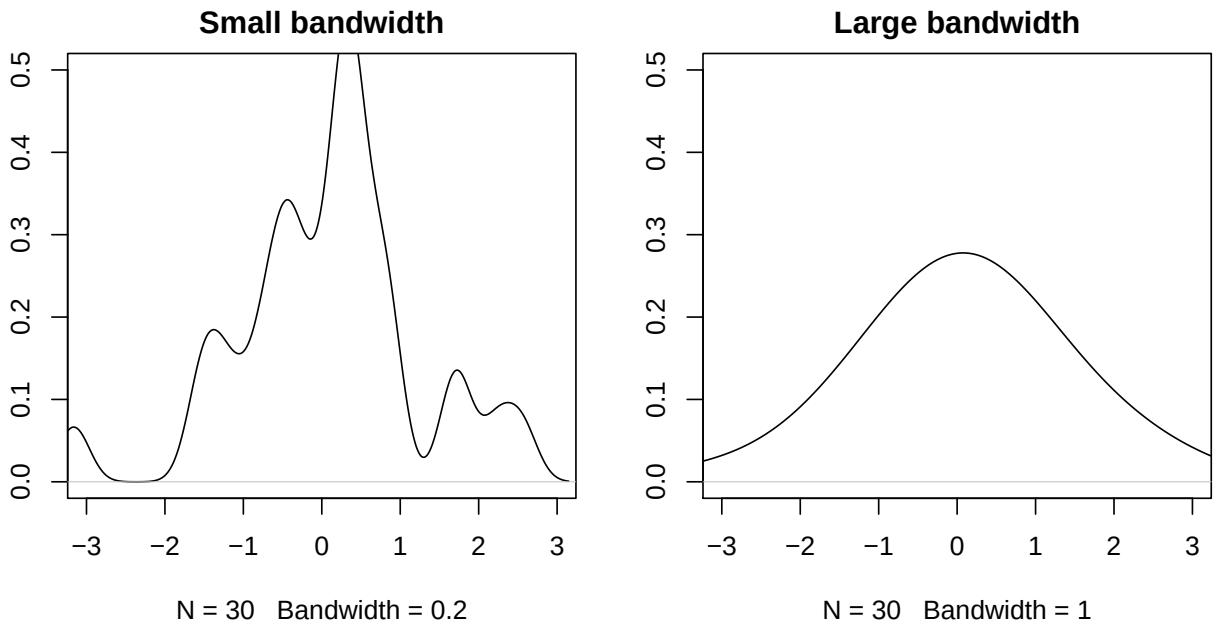
- We can generalize this idea to form a class of density estimators: Kernel density estimation. A kernel function K is a function which play similar roles as the rectangle and triangle here.

Kernel Density Estimator

- Given a kernel function $K(\cdot, \cdot)$, the corresponding kernel density estimator is
$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K(x, x_i)$$
- The kernel function has the following properties
  - It has integral one: $\int K_h(x, x_i) dx = 1$ for all x_i
  - usually positive
  - usually symmetric
- Common choices
  - Gaussian kernel: $K_h(x, x_i) = \phi(x, \mu = x_i, \sigma = h)$
  - Epanechnikov: $K_h(x, x_i) = 3/4h[1 - (t/h)^2]$ for $|t| \leq h$
  - Rectangular: $K_h(x, x_i) = I(x_i - h/2 \leq x \leq x_i + h/2)/h$
- h is called a bandwidth parameter which control the smoothness

Bandwidth

- The smoothing parameter h, which determines the width of the local neighborhood, has to be determined.
- Large h implies lower variance but higher bias.
- If h is small, we have rough density estimate which has smaller bias but large variance.



- In R, `density(x)` will do the job

Smoothing scatter plot

- If we have a scatter plot (x, y), we are interested in estimating the regression function $E(Y|X = x)$
- The simplest method may be the local neighborhood or averaging
  - k-nearest-neighbor average

$$\hat{f}(x) = \text{Avg}(y_i | x_i \in N_k(x)) = \frac{\sum_{x_i \in N_k(x)} y_i}{\#\{x_i \in N_k(x)\}}$$

  - local averaging

$$\hat{f}(x) = \text{Avg}(y_i | |x - x_i| < h) = \frac{\sum_{|x - x_i| < h} y_i}{\#\{|x - x_i| < h\}}$$

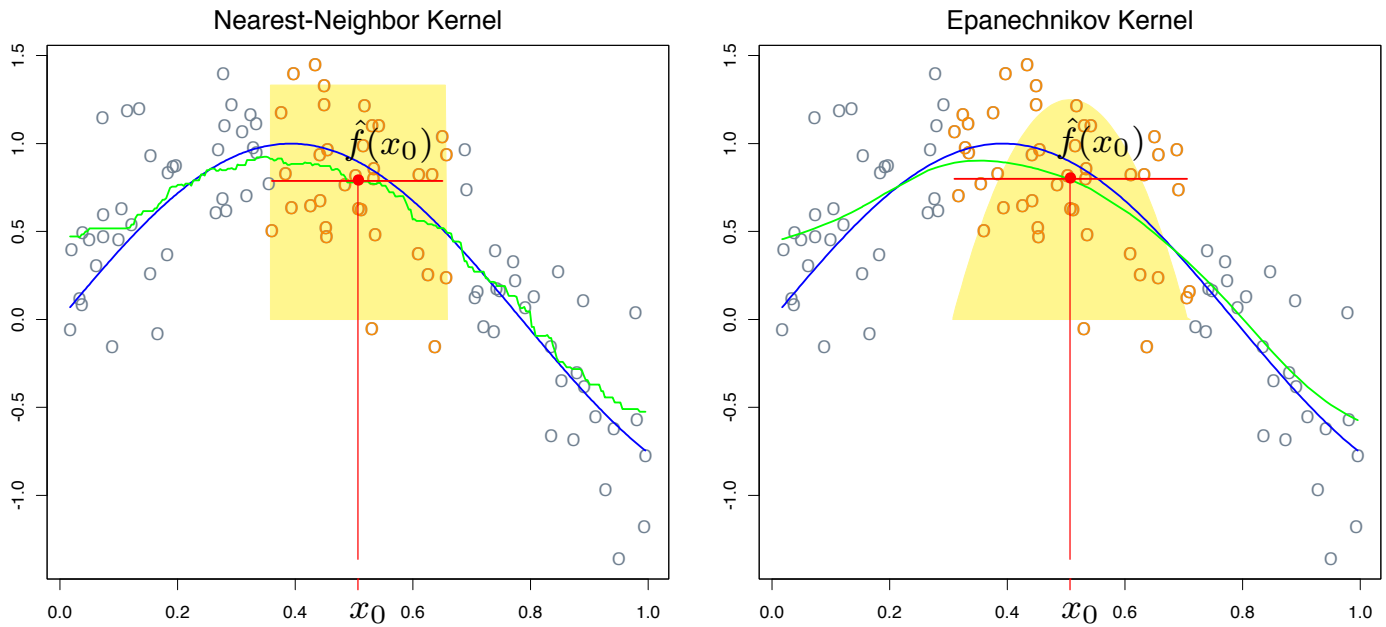
- The averages above change in discrete way, leading to discontinuous $\hat{f}(x)$
- Similar idea again, replace the indicator functions by kernel functions (Nadaraya–Watson),

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

Example

In each panel 100 pairs x_i, y_i are generated at random from the blue curve with Gaussian errors: $Y = \sin(4X) + \varepsilon,\ X \sim U[0,1],\ \varepsilon \sim N(0, 1/3)$.

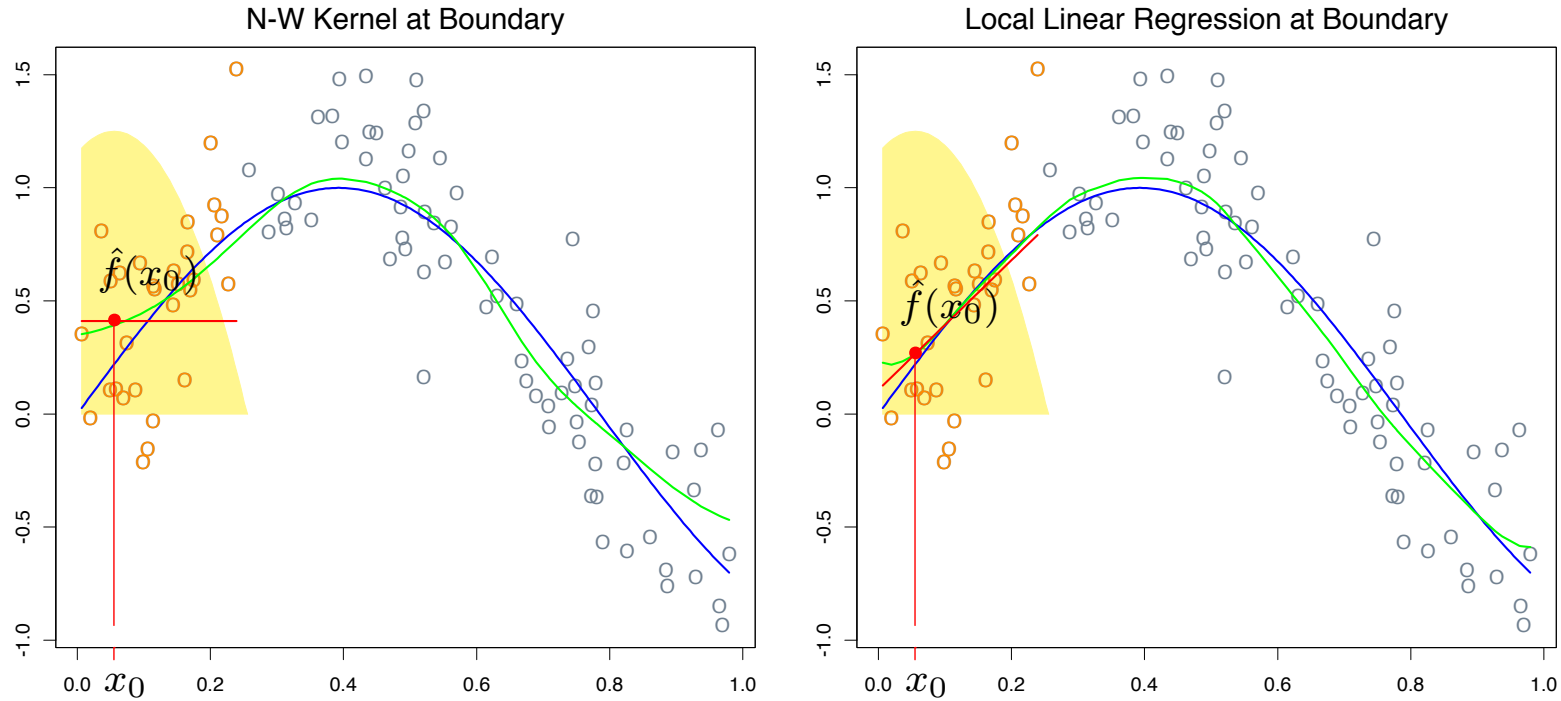
- green curve: 30-nearest-neighbor running-mean smoother.
- The red point is the fitted constant, the red circles indicate those observations contributing to the fit



- In R, `ksmooth` provides Nadaraya–Watson estimates with rectangular and gaussian kernels.

Drawback of raw moving average

- The smooth kernel fit still has problems.
- Locally-weighted averages can be badly biased on the boundaries of the domain because of the asymmetry of the kernel in that region.
- By fitting straight lines rather than constants locally, we can remove this bias.



Local linear regression

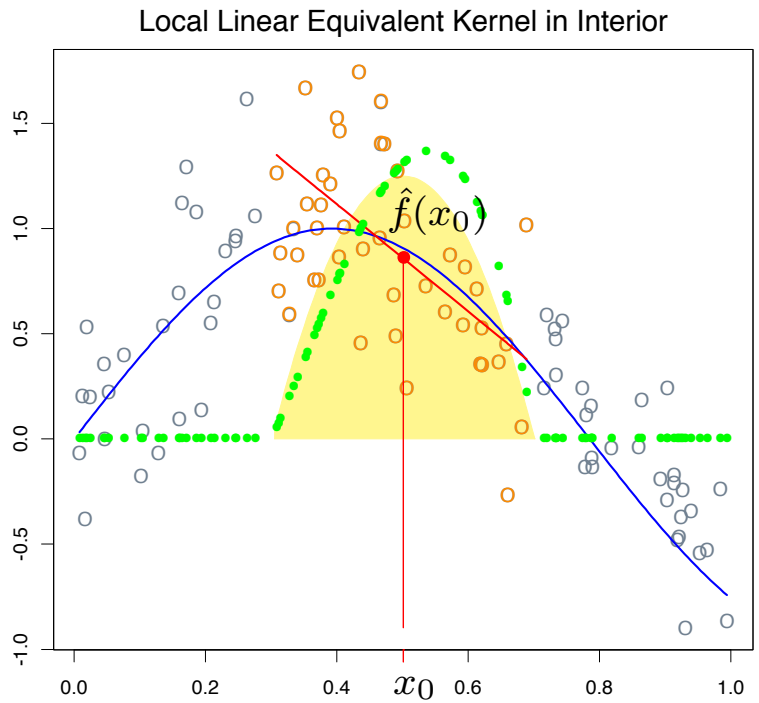
- Locally weighted regression solves a separate weighted least squares problem at each target point x:

$$\hat{f}(x) = \hat{a}(x) + \hat{b}(x)x$$

where

$$\hat{a}(x), \hat{b}(x) = \arg\min_{a,b} \sum_{k=1}^{n} K_h(x, x_i)(y_i - a - bx_i)^2$$

- Notice that although we fit an entire linear model to the data in the region, we only use it to evaluate the fit at the single point x
- In R, `loess` or `lowess`



High dimensional performance

- Local regression also generalizes very naturally when we want to fit models that are local in a pair of variables X_1 and X_2, rather than one (R package `locfit`)
- Theoretically the same approach can be implemented in higher dimensions, using linear regressions fit to p-dimensional neighborhoods.
- However, local regression can perform poorly if p is much larger than about 3 or 4 because there will generally be very few training observations close to x_0 (curse of dimensionality)
- Nearest-neighbors regression suffers from a similar problem in high dimensions.