STAT 432: Basics of Statistical Learning

Kernel Methods

Shiwei Lan, Ph.D. <shiwei@illinois.edu>

http://shiwei.stat.illinois.edu/stat432.html

March 28, 2019

University of Illinois at Urbana-Champaign

Outline

- K-Nearest Neighbor Revisited
- Kernel Smoother and Kernel Functions
- Local Polynomial Regression
- Kernel Density Estimation
- Multivariate Kernels

K-Nearest Neighbor Revisited

K-Nearest Neighbor

- K-nearest neighbor regression is weighted averaging
- Suppose we want to predict x. With training set $\{x_i, y_i\}_{i=1}^n$,

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

where the weights

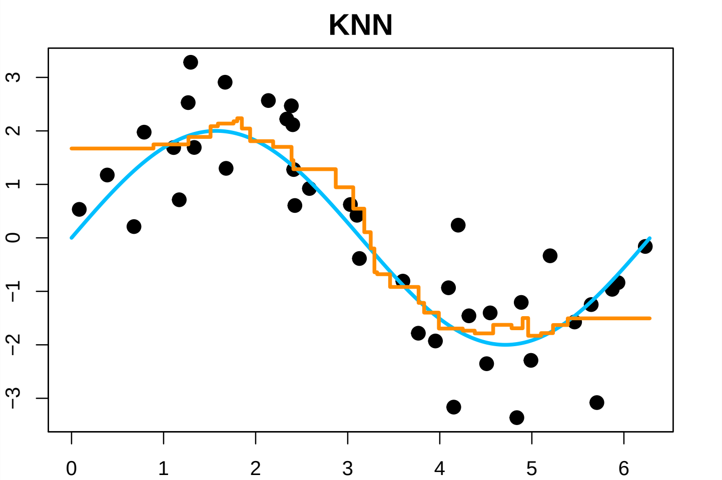
$$w(x, x_i) = \begin{cases} \frac{1}{k} & \text{if } x_i \in N_k(x) \\ 0 & \text{o.w.} \end{cases}$$

- $N_k(x)$ is a set of k observations in the neighborhood of x

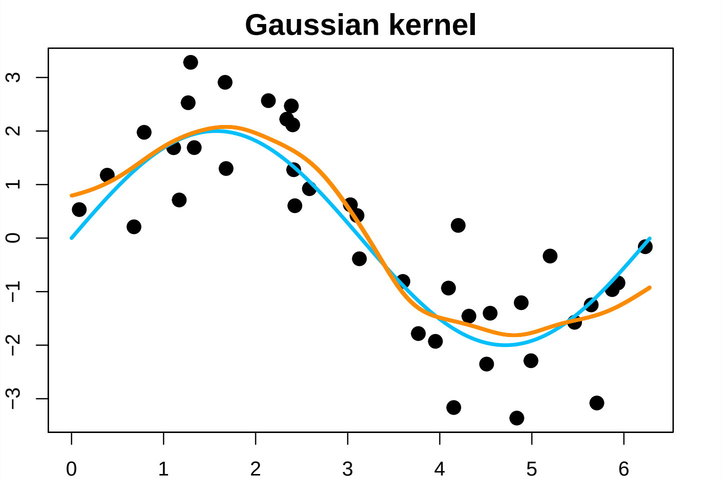
K-Nearest Neighbor

- Problems:
  - Requires sorting after the weights are calculated — $\mathcal{O}(n \log(n))$
  - The weights $w(x, x_i)$ drop off abruptly to zero outside the neighborhood of x. This accounts for jagged appearance of the fit.
- To improve:
  - Easier method to assign weights
  - Smoothed weight functions

KNN vs. Gaussian Kernel



KNN vs. Gaussian Kernel



Kernel Smoother and Kernel Functions

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
- For any point x, we calculate the weighted average

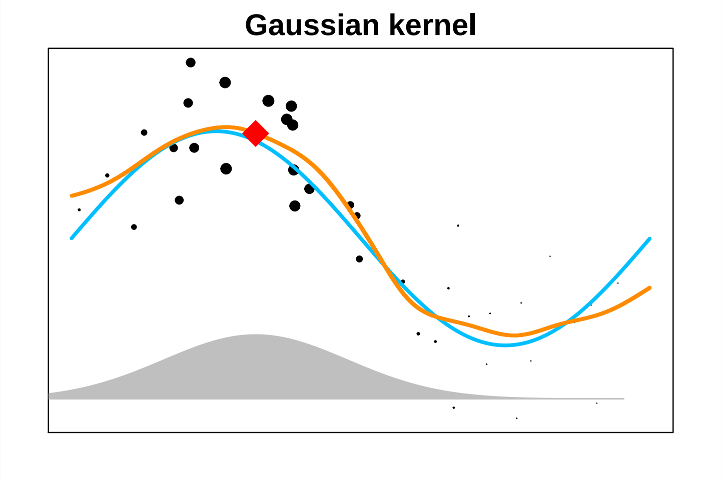
$$\widehat{f}(x) = \frac{\sum_i K_h(x, x_i) y_i}{\sum_i K_h(x, x_i)}$$

where h is a tuning parameter (called bandwidth) that controls the distance.

Kernel Smoother

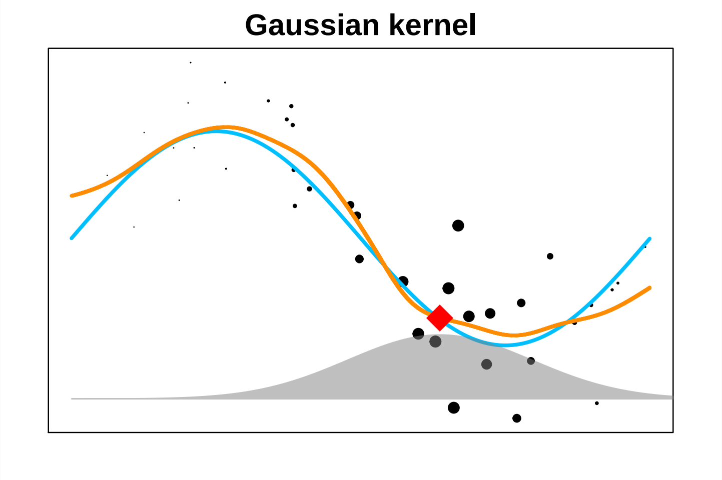
- The estimator is called Nadaraya-Watson kernel estimator
- $K_h(\cdot, \cdot)$ is a kernel function controlled by bandwidth h
  - Assigns larger value if the two inputs are closer to each other.
- Requires little or no training time. All the work gets done at evaluation time (same as kNN), however, no sorting is required.
- KNN is also a type of kernel method: the weight is $1/k$ or 0.

Gaussian Kernel



Predicting the target point $x = 2$

Gaussian Kernel



Predicting the target point $x = 4$

Gaussian Kernel

- We often write the kernel function $K(x_1, x_2)$ in a different way, with $u = x_1 - x_2$
$$K_h(u) = h^{-1}K(u/h)$$
and $K(\cdot)$ is a "standard" version of the kernel (with $h = 1$).
- For example, a popular choice is the Gaussian kernel:
$$\begin{aligned} K(u) &= \phi(u) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left\{-u^2/2\right\} \end{aligned}$$
- To incorporate the bandwidth (sd of Gaussian),
$$K_h(u) = \frac{1}{h\sqrt{2\pi}} \exp\left\{-(u/h)^2/2\right\}$$

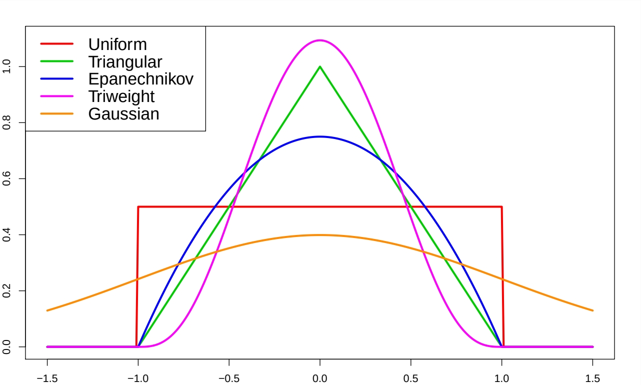
The Effect of Bandwidth

- The bandwidth h plays a crucial role:
- If h is large, we assign a relatively large kernel weight even when two points are far away from each other.
  - This works the same as a larger k in KNN
- If h is small, we assign a relatively small kernel weight as soon as two points are moving away from each other.
  - This works the same as a small k in KNN
- Large h: smoother estimate, bias $\uparrow$, variance $\downarrow$
- Small h: rougher estimate, bias $\downarrow$, variance $\uparrow$

Other Popular Kernels

- A kernel function usually satisfies the following properties:
  - K is properly normalized (e.g. pdf): $\int K(u)du = 1$;
  - K is symmetric around 0: $K(u) = K(-u)$;
  - $\int u^2 K(u)du \leq \infty$
  - $\int K^2(u)du \leq \infty$
- Many different kernel functions (besides Gaussian):
  - Uniform: $K(u) = \frac{1}{2} \cdot 1(|u| \leq 1)$
  - Triangular: $K(u) = (1 - |u|) \cdot 1(|u| \leq 1)$
  - Epanechnikov: $K(u) = \frac{3}{4}(1 - u^2) \cdot 1(|u| \leq 1)$
  - Triweight: $K(u) = \frac{35}{32}(1 - u^2)^3 \cdot 1(|u| \leq 1)$
  - ...
- They can all incorporate the bandwidth h

Different Kernel Functions



Different Kernel Functions

- So what's the difference between different kernels?
- Efficiency of a kernel function:
  - Efficiency is measured by $\left(\int u^2K(u)du\right)^{\frac{1}{2}}\int K^2(u)du$
  - A quantity that evaluates the mean integrated squared error (MISE) of a kernel estimator defined as

$$\text{MISE}(\widehat{f}) = \text{E}\int\left(\widehat{f}(x) - f(x)\right)^2 dx$$

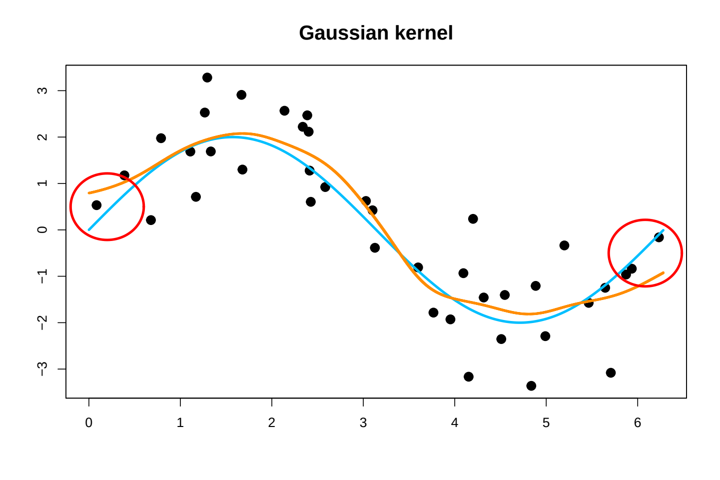
  - Epanechnikov kernel is the most efficient.
- However, the efficiency of other kernels are not too bad: Gaussian is 95%; Uniform is 93% (the worst, relative to Epanechnikov kernel)
- Choosing h is far more important than choosing the kernel.

Local Linear Regression

Boundary Effects

- The Nadaraya-Watson kernel is notorious for boundary effects.
- In the previous Gaussian kernel example, there is a substantial bias at the boundaries.
- Intuition: all neighboring points are smaller/larger than the boundary $f(x)$.
- Solution: Locally weighted linear regression can make a first order correction (constants vs. straight lines)

Kernels



Local Linear Regression

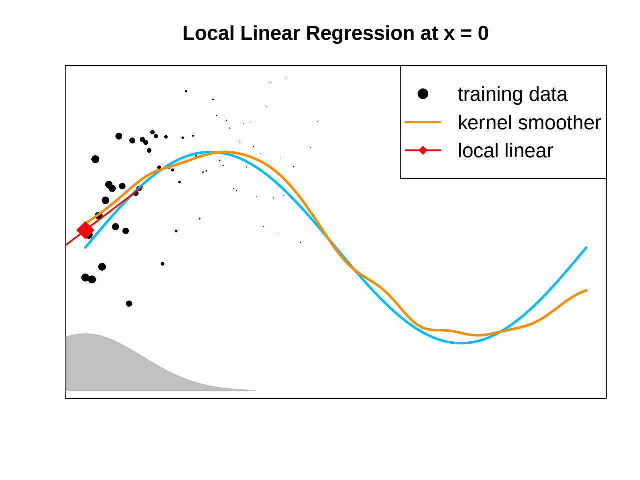
- Recall that a (global) simple linear regression minimize the following objective function

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i\right)^2$$

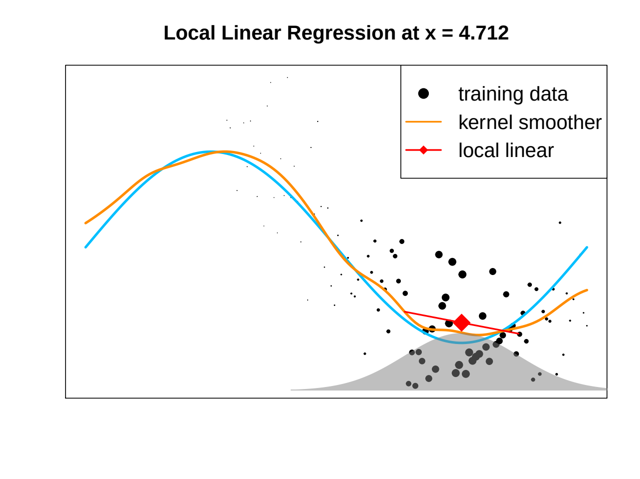
- Idea: how about we pay more attention to the points that are closer to a target point x?

$$\begin{aligned} &\text{RSS (locally weighted)} \\ &= \sum_{i=1}^{n} K_h(x, x_i)\left(y_i - \beta_0 - \beta_1 x_i\right)^2 \end{aligned}$$

Local Linear Regression



Local Linear Regression



Local Linear Regression

- Similar to other kernel methods, the estimated (local) linear function is only valid at the local point x.
- The parameters β_0 and β_1 are in fact $\beta_0(x)$ and $\beta_1(x)$
- Hence, if we are interested in a different target point, we need to refit the model entirely. This could be computationally intense.
- The catch: more parameters to be estimated — smaller bias but larger variance

Local Polynomial Regression

- In general, we may consider a locally weighted d polynomial regression, which minimizes the local RSS objective function

$$\sum_{i=1}^{n} K_h(x, x_i) \Big[y_i - \beta_0(x) - \sum_{r=1}^{d} \beta_j(x) x_i^r \Big]^2$$

- Further reduces bias, at a price of higher variance.
- Warning: very sensitive to the choice of bandwidth h.
- Note: although its possible, but we usually do no use $r > 2$

Solving the Local Polynomial Regression

- Since the local polynomial regression is a weighted linear model, we may rewrite things in a matrix form:
- Let $\mathbf{W}$ be a $n \times n$ diagonal matrix defined as

$$\mathbf{W} = \text{diag}\big(K_h(x, x_1), K_h(x, x_2), \ldots, K_h(x, x_n)\big)$$

- Then the weighted RSS can be written as

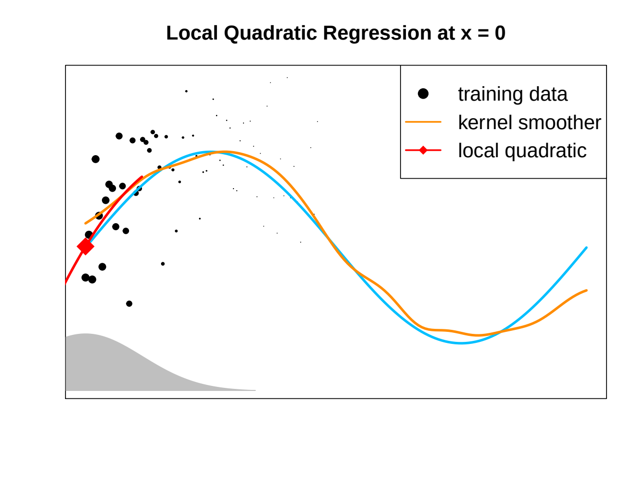
$$\sum_{i=1}^{n} K_h(x, x_i)\big(y_i - \beta_0 - \beta_1 x_i\big)^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\mathsf{T}\mathbf{W}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

- And the solution is (from normal equations)

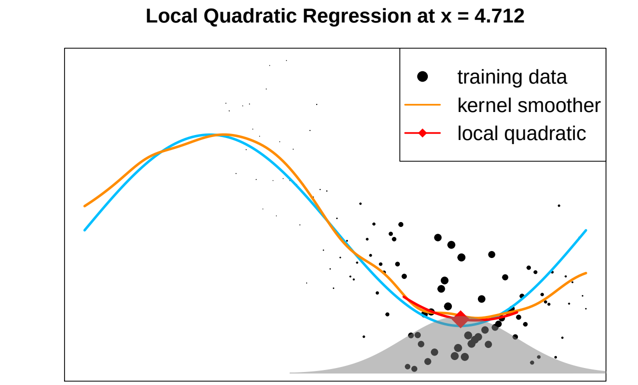
$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{y}$$

- Note that we need to recalculate this for each target points x.

Local Quadratic Regression



Local Quadratic Regression



R implementations

- 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)

Kernel Density Estimation

Kernel Density Estimation

- Another area where we often use the kernel methods is estimating the density
- Given some observations from an unknown distribution, we want to estimate the pdf of that distribution (unsupervised)

$$X_1, \ldots, X_n \quad \overset{\text{i.i.d}}{\sim} \quad f(\cdot)$$

- Some density estimation methods
  - Histograms
  - Assume a family of distributions and estimate parameters
  - Kernel density estimator

Histogram Estimator of Density Functions

- If $f(\cdot)$ is the pdf of X, then we have:

$$\int f(u)du = 1, \quad \text{and} \quad f(u) > 0 \quad \text{for all } u$$

$$\int_{x-\frac{h}{2}}^{x+\frac{h}{2}} f(u)du = \mathsf{P}\left(x - \frac{h}{2} \leq X \leq x + \frac{h}{2}\right)$$

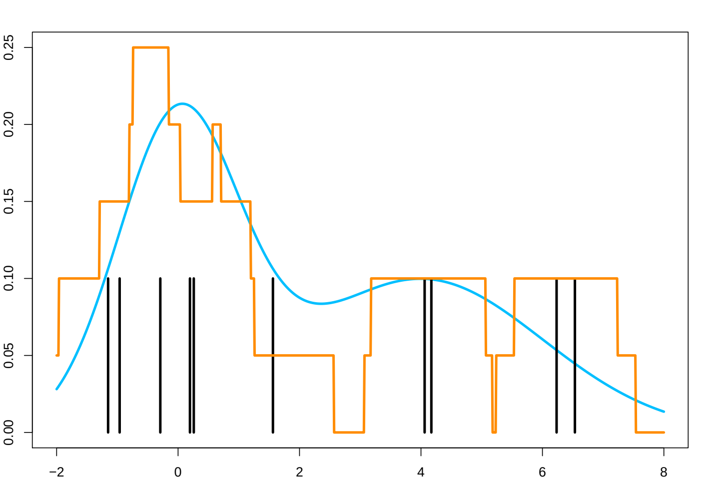
$$f(x) = \lim_{h \to 0} \frac{1}{h} \mathsf{P}\left(x - \frac{h}{2} \leq X \leq x + \frac{h}{2}\right)$$

- A natural estimator, with a set of observations $\{x_i\}_{i=1}^n$, is

$$\widehat{f}(x) = \frac{\#\left\{x_i : x_i \in [x - \frac{h}{2}, x + \frac{h}{2}]\right\}}{hn}$$

- This is very similar to the histogram estimator. But it is bumpy and non-smooth.

Uniform Kernel Density Estimation



Kernel Density Estimation

- Parzen estimate

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

- $K_h(\cdot)$ here is a kernel function, controlled by h.
- Again, we can use the popular Gaussian kernel function for this task

$$K_h(x, x_i) = \frac{1}{h\sqrt{2\pi}} \exp\left\{ - \frac{(x - x_i)^2}{2h^2} \right\}$$

R implementations

- hist makes histograms
- density for kernel density estimator
- bw.nrd and a set of related functions for bandwidth selection
- The rule of thumb (Silverman 1986) for h in univariate case is

$$\widehat{h} = 1.06\widehat{\sigma}n^{-1/5}$$

where $\widehat{\sigma}$ is the sample standard deviation.

Multivariate Kernel Estimations

Multivariate Density Functions

- We can extend the idea from univariate to multivariate case.
- The only thing that needs to be changed is the kernel function:

$$K_{\mathbf{H}}(\mathbf{u}, \mathbf{v})$$

  for any p dimensional vectors $\mathbf{u}$ and $\mathbf{v}$, and a kernel bandwidth matrix $\mathbf{H}$.
- If we are still using a Gaussian density function, then

$$K_{\mathbf{H}}(\mathbf{u}, \mathbf{v}) = \frac{1}{(2\pi)^{p/2}|\mathbf{H}|^{1/2}} \exp\Big\{ -\frac{1}{2}(\mathbf{u}-\mathbf{v})^{\mathsf{T}}\mathbf{H}^{-1}(\mathbf{u}-\mathbf{v})\Big\}$$

- A simplified version is to take $\mathbf{H}$ as a diagonal matrix.