Kernel Methods

Matthew S. Shotwell, Ph.D.

Department of Biostatistics
Vanderbilt University School of Medicine
Nashville, TN, USA

February 16, 2020

Kernel methods

Kernel methods are a broad class, including:

- k-nearest-neighbors
- support vector maching (SVM)
- local regression
- kernel density estimation

Kernel methods

All kernel methods used for supervised learning:

- fit a different but simple model $f(x_0)$ at each x_0
- only use data in a local neighborhood about x_0
- localize using weighting or 'kernel' function: $K_\lambda(x_0, x)$
- λ is smoothing parameter; determines size of neighborhood
- requires little or no training until the time of prediction; most computation occurs at time of prediction

Nadaraya-Watson estimator

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

We want to make a prediction at x_0. NW-estimator is a weighted average of training y_i's. Weights are bigger x_i's near x_0. Kernel function determines how near x_i is to x_0.

Kernel functions

- Epanechnikov quadratic kernel

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

- as λ increaes, neighborhood gets bigger
- as λ increaes, model of Y vs X less flexible
- as λ increaes, higher bias, lower variance
- λ is a tuning parameter

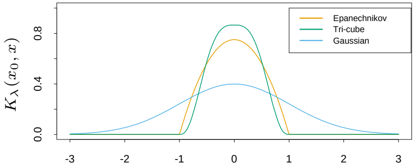

FIGURE 6.2. *A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to* 1*. The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov kernel has none. The Gaussian kernel is continuously differentiable, but has infinite support.*

Kernel functions

- Epanechnikov quadratic kernel

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

- note that $K_\lambda(x_0, x)$ equal to zero for $||x - x_0|| > \lambda$
- why prefer a kernel that becomes exactly zero?

Kernel functions

- K-nearest neighbors kernel

$$K_k(x_0, x) = I(||x - x_0|| \leqslant ||x_{[k]} - x_0||)$$

- $x_{[k]}$ is the k'th nearest neighbor to x_0
- as k increaes, neighborhood gets bigger
- as k increaes, model of Y vs X less flexible
- as k increaes, higher bias, lower variance
- k is a tuning parameter

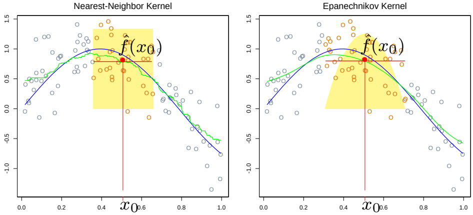

FIGURE 6.1. *In each panel* 100 *pairs* x_i, y_i *are generated at random from the blue curve with Gaussian errors:* $Y = \sin(4X) + \varepsilon$, $X \sim U[0, 1]$, $\varepsilon \sim N(0, 1/3)$. *In the left panel the green curve is the result of a* 30*-nearest-neighbor running-mean smoother. The red point is the fitted constant* $\hat{f}(x_0)$, *and the red circles indicate those observations contributing to the fit at* x_0. *The solid yellow region indicates the weights assigned to observations. In the right panel, the green curve is the kernel-weighted average, using an Epanechnikov kernel with (half) window width* $\lambda = 0.2$.

Code example

```
kernel-methods-examples-mcycle.R
```

Local linear regression

- N-W estimator is "local constant regression"
- local linear regression assumes local linearity
- different β for each x_0
- prediction at x_0 is $\hat{y}_0 = x_0\beta(x_0)$
- find $\beta(x_0)$ by minimizing weighted training error

$$\begin{aligned}\overline{\mathrm{err}}_\lambda(x_0) &= \sum_{i=1}^{N} K_\lambda(x_0, x_i) L(y_i, \hat{y}_i) \\ &= \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - x_i\beta(x_0)]^2\end{aligned}$$

NW vs. local linear

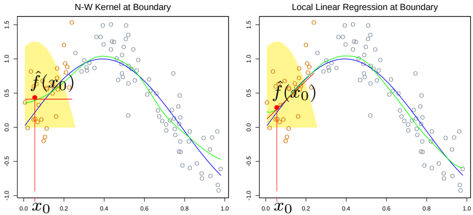

FIGURE 6.3. *The locally weighted average has bias problems at or near the boundaries of the domain. The true function is approximately linear here, but most of the observations in the neighborhood have a higher mean than the target point, so despite weighting, their mean will be biased upwards. By fitting a locally weighted linear regression (right panel), this bias is removed to first order*

Local linear regression

- can implement NW as local linear; intercept only
- can also do local polynomials or local splines

Code example

`kernel-methods-examples-mcycle.R`

Local regression with multiple predictors

- local linear regression is very flexible
- need sample size to grow exponentially in p to maintain bias and variance
- can make restrictions to regularize the problem

Structured kernels

Structured Epanechnikov kernel

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

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

In this example A forces the kernel function to consider only on first dimension of the input; all others are ignored.

Code example

`mixture-data-knn-local.R`

Varying coefficients models

- divide p predictors into x and z
- assume $f(X, Z) = X\beta(Z)$
- $f(X, Z)$ is linear in X by different for each Z
- special kind of interaction between X and Z

$$KSS_\lambda(z_0) = \sum_{i=1}^{N} K_\lambda(z_0, z_i)[y_i - x_i\beta(z_0)]^2$$

z - gender and age, x - diameter

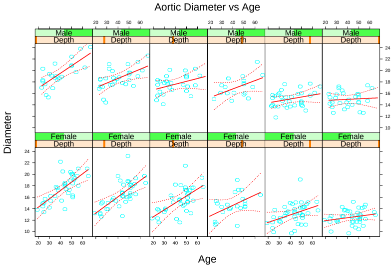

FIGURE 6.10. *In each panel the* `aorta diameter` *is modeled as a linear function of* `age`*. The coefficients of this model vary with* `gender` *and* `depth` *down the* `aorta` *(left is near the top, right is low down). There is a clear trend in the coefficients of the linear model.*

z - gender and age, x - diameter

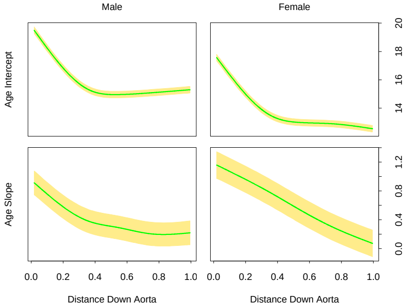

FIGURE 6.11. *The intercept and slope of* `age` *as a function of* `distance` *down the aorta, separately for males and females. The yellow bands indicate one standard error.*

Local likelihood

- likelihood function depends on x_0
- e.g., say $l_i(y_i, x_i, \theta) = \phi(y_i, \mu = x_i\theta, \sigma = 1)$
- $l(\theta(x_0)) = \sum_{i=1}^{N} K_\lambda(x_0, x_i) l_i(y_i, x_i, \theta)$

Local logistic regression in R

- use weighting
- R docs say that weights w_i affect binomial density as follows

$$
\begin{aligned}
f_w(y_i) &= p_i^{w_i y_i}(1-p_i)^{w_i(1-y_i)} \\
\log f_w(y_i) &= w_i y_i \log p_i + w_i(1-y_i)\log(1-p_i) \\
&= w_i(y_i \log p_i + (1-y_i)\log(1-p_i)) \\
&= w_i \log f(y_i)
\end{aligned}
$$

- let $w_i = K_\lambda(x_0, x_i)$

Kernel density estimation

- unsupervised learning method
- summarize distribution of some data
- Parzen estimator

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

- $K_\lambda(x_0, x)$ and N_λ must be chosen such that $\hat{f}_X(x_0)$ integrates to 1
- e.g., $K_\lambda(x_0, x) = \phi(x_0 - x, 0, B_\lambda)$ zero mean normal density with var-cov B_λ
- $\int_{-\infty}^{\infty} \hat{f}_X(t)dt = \frac{1}{N_\lambda} \sum_{i=1}^{N} \phi(t - x_i, 0, B_\lambda)dt = \frac{N}{N_\lambda}$
- thus $N_\lambda = N$

Kernel density classification

- suppose $x_1, \ldots, x_N$, classes $g_1, \ldots, g_N \in \mathcal{G}$, and targets $y_1, \ldots, y_N$
- let $\hat{f}_j(x_0)$ be the KDE for class $\mathcal{G}_j$
- let $\hat{\pi}_j = N_j/N$
- $\hat{Pr}(G = \mathcal{G}_j | X = x_0) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_l \hat{\pi}_l \hat{f}_l(x_0)}$