Kernels

36-290 – Statistical Research Methodology

Week 10 Thursday – Fall 2021

Kernel

A kernel $K(x)$ is a weighting function used in estimators. Full stop.

Why would we use a weighting function? Perhaps we wish to estimate a quantity (e.g., the value of a response variable) at a particular coordinate $\mathbf{x}$. One way to do this is to look at nearby data points (such as when we use K nearest neighbors). But maybe you don't want to have every nearby datum count equally; perhaps you wish to give data nearer to $\mathbf{x}$ more weight, and data farther from $\mathbf{x}$ less weight. You can accomplish this by incorporating a kernel into your estimator (creating, for instance, kernel K nearest neighbors or kernel-KNN).

A kernel technically has only one required property:

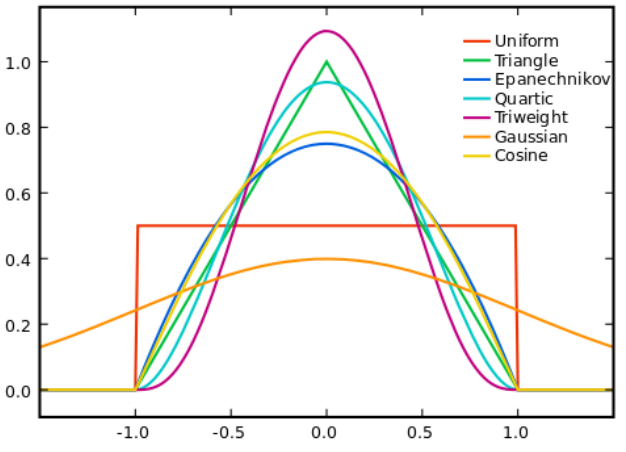
- $K(x) \geq 0$ for all x.

However, in the manner that kernels are used in statistics, there are two other properties that are usually satisfied:

- $\int_{-\infty}^{\infty} K(x)dx = 1$; and
- $K(-x) = K(x)$ for all x.

In short: a kernel is a symmetric probability density function!

Commonly Used Kernels



(See this web page.)

A general rule of thumb: the choice of kernel will have little effect on estimation, particularly if the sample size is large! The Gaussian kernel (i.e., a normal pdf) is by far the most common choice, and is the default for `R` functions that utilize kernels.

Kernel Density Estimation

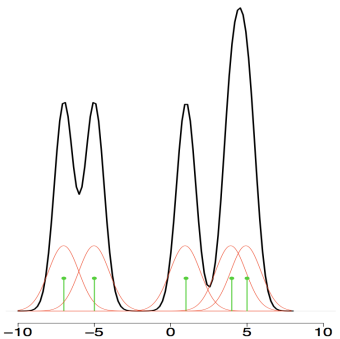
The setting: we have n data sampled from some unknown distribution P with pdf $f(x)$:

$$X_1, \ldots, X_n \sim P$$

Density estimation is a nonparametric technique for attempting to estimate $f(x)$ with as few assumptions about its form as possible. Note that there is no response variable here: we are simply trying to estimate the underlying pdf from which data are sampled.

In kernel density estimation (or KDE) in particular, the estimator is

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x - X_i}{h} \right) .$$



The key? To estimate the optimal value of h!

KDE: But Hold Up...

"Why exactly is $K(\cdot)$ written with that weird argument of $(x - X_i)/h$?"

Good question. The Gaussian kernel is conventionally defined (see the web page) as a *standard normal*:

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right).$$

When you plug in the argument $(x - X_i)/h$, you get

$$K\left(\frac{x - X_i}{h}\right) = \frac{1}{\sqrt{2\pi h^2}} \exp\left(-\frac{(x - X_i)^2}{2h^2}\right).$$

This is now a normal pdf with mean x and standard deviation h.

KDE: Estimating the Bandwidth h

The workhorse function for univariate KDE in `R` is `density()`:

```
density(x, ...)
## Default S3 method:
density(x, bw = "nrd0", adjust = 1,
        kernel = c("gaussian", "epanechnikov", "rectangular",
                   "triangular", "biweight",
                   "cosine", "optcosine"),
        weights = NULL, window = kernel, width,
        give.Rkern = FALSE,
        n = 512, from, to, cut = 3, na.rm = FALSE, ...)
```

`x` is your vector of data, and the `kernel` is `"gaussian"` by default. The key argument here is `bw`, standing for "bandwidth". There are two general approaches for estimating the optimal bandwidth.

The "plug-in" approach utilizes empirical formulae. Above, `nrd0` represents the plug-in formula of Silverman (1986):

$$h_{\text{opt}} = 0.9 \cdot \min\left(s, \frac{IQR}{1.34}\right) \cdot n^{-1/5}$$

where s is the sample standard deviation and IQR is the inter-quartile range. A variant, `nrd`, utilizes Scott's rule and simply changes 0.9 to 1.06.

KDE: Estimating the Bandwidth h

The other general approach is to minimize a loss function. (The MSE in regression is an estimation of the average value of a loss function...so you've seen loss functions before.)

A typical loss function to use is the *integrated mean-squared error*:

$$L = \int (\hat{f}_h(x) - f(x))^2 dx \propto \int \hat{f}_h^2(x) dx - 2 \int \hat{f}_h(x) f(x) dx \, .$$

This is coded as the "unbiased CV estimator"

$$\hat{J}(h) = \left(\int \hat{f}_h^2(x) dx \right) - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{h,-i}(X_i) \, ,$$

where in the second term, $\hat{f}_{h,-i}(X_i)$ is the predicted value of $\hat{f}_h(X_i)$ when X_i is left out of the estimation process. (This is "leave-one-out cross-validation.")

When you use `density()`, you would apply the argument `bw="ucv"`.

Note that the output from `density()` will list the optimum value of h but will not show values of $\hat{J}(h)$ for all values of h tried. If you need to plot $\hat{J}(h)$ vs. h, in the same manner that you plotted MSE vs. k during the validation step of KNN, you might use, e.g., the `kedd` package. But this is generally not necessary.

KDE in Two or More Dimensions

The base `stats` package function `density()` only works with univariate data.

In two dimensions, an oft-used density estimator is `kde2d()` from the `MASS` package. Note that

- while it allows h_x and h_y to be different, it doesn't allow for correlations between them; and
- there is no option for searching for optimal values.

So: you pick $\mathbf{h}$ but it might not be optimal.

For multivariate cases, you may need to strike out into newer territory, like the `ks` package.
Regardless of which package you choose, keep in mind that due to the curse of dimensionality, the fact that the amount of data you need to make precise statements in estimation goes up exponentially with dimension, useful KDE becomes impossible beyond $p \approx 5$.

Kernel Regression

As a final note, realize that one can apply kernels in the regression setting as well as in the density estimation setting.

The classic kernel regression estimator is the Nadaraya-Watson estimator:

$$\hat{y}_h(x) = \sum_{i=1}^{n} l_i(x) Y_i \,,$$

where

$$l_i(x) = \frac{K\left(\frac{x - X_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x - X_j}{h}\right)} \,.$$

Basically, the regression estimate is the average of all the *weighted* observed response values; the farther x is from a datum, the less weight that datum has in determining the regression estimate at x.

The workhorse function for kernel regression in `R` is `ksmooth()` from the base `stats` package. Tuning to find the optimal value of h is not necessarily simple; it is $O(n^{-1/5})$ and thus you can use $n^{-1/5}$ as an initial estimate.