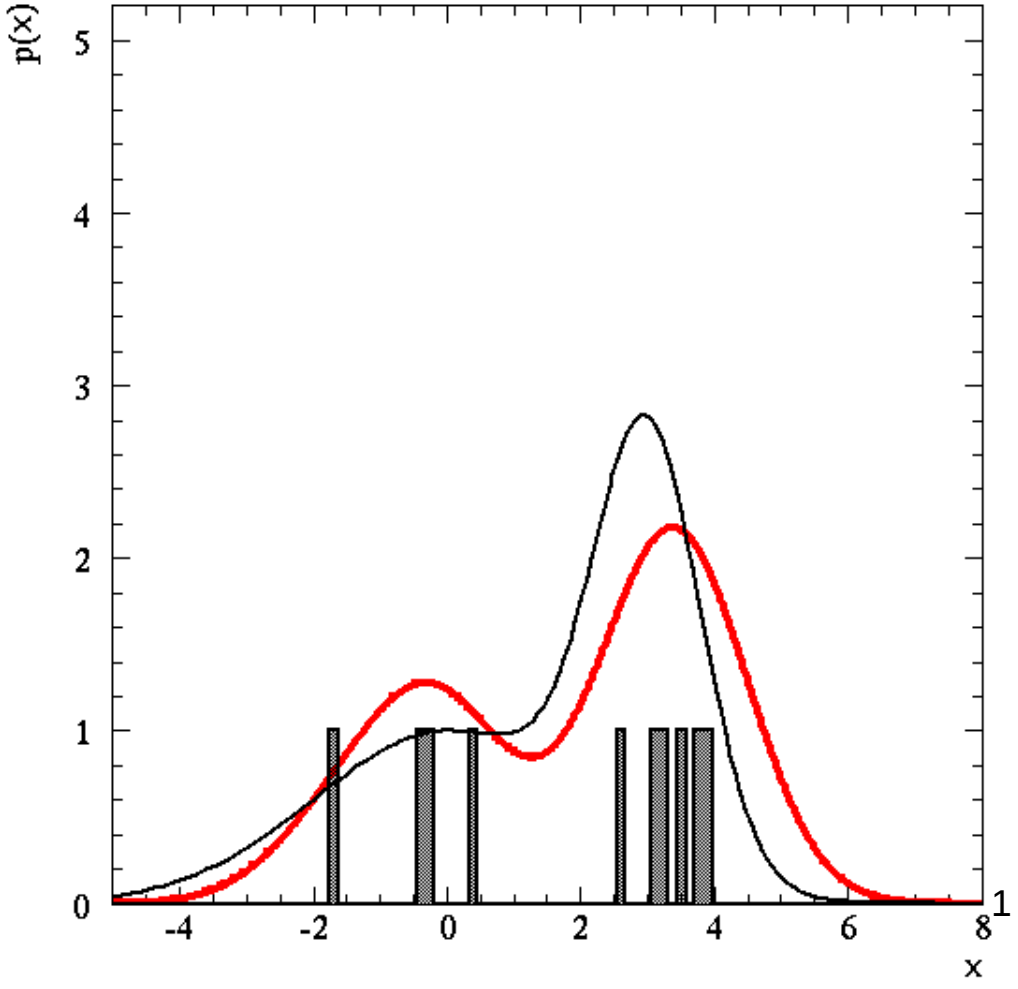
Physics 509: Kernel Density Estimation

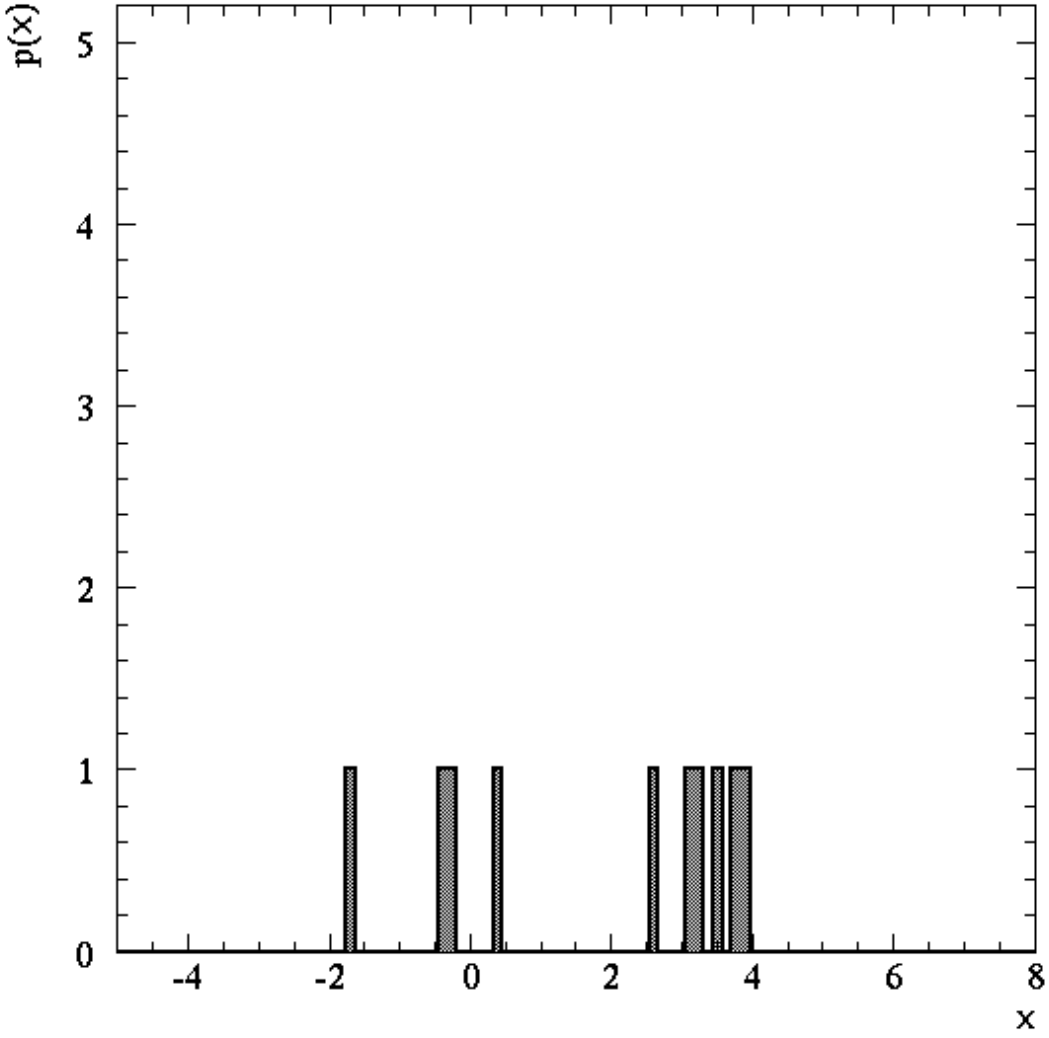
Scott Oser

Lecture #23



What distribution was this data drawn from?

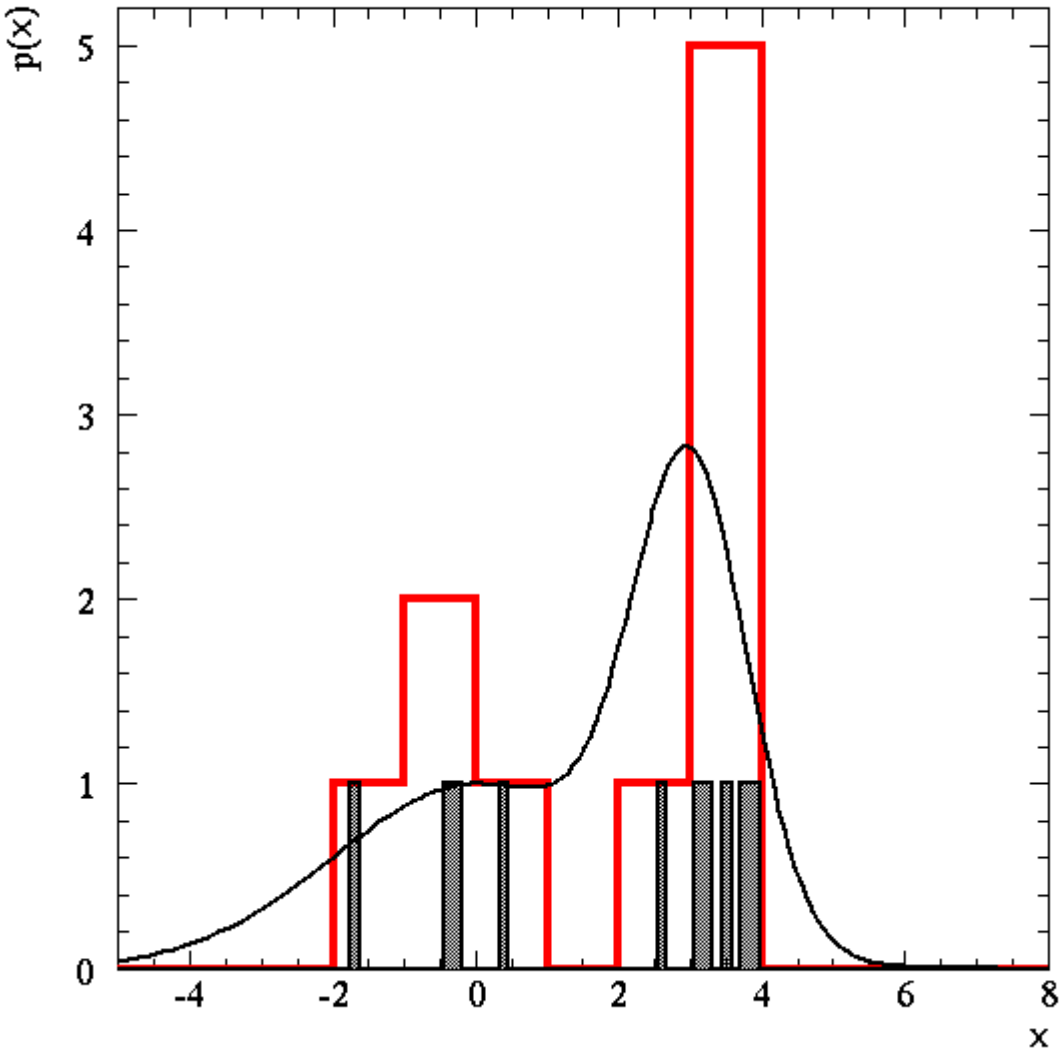
These ten points were drawn from some unknown distribution. How do we estimate approximate the underlying distribution?



Approximate it as a histogram?

The red histogram is not a great match to the true distribution (smooth black curve).

Also, what bin size should you use, anyway? Bigger bins poorly approximate shape, but narrow bins are limited by statistics.



Non-parametric estimation of distributions

We would like a non-parametric way of describing the shape of distributions. Histogramming is the simplest way to do this, but of course very few real distributions have “step” features in reality.

Histograms also are sensitive to where you start the bin boundaries.

Discontinuous functions also have mathematical difficulties ... cannot take derivatives of them. Most minimizers have trouble with them. Imagine the nightmare of trying to do a ML fit to a PDF with discontinuities ---- likelihood function is likely to be discontinuous if fit parameters can move data events across bin boundaries..

“Smart” parametrizations

Sometimes you might be able to empirically describe the distribution by a parametrized form. One useful version is Pearson’s hybrid density estimator, which is the family of solutions to

$$\frac{d \ln f(x)}{dx} = \frac{x-a}{bx^2+cx+d}$$

(This parametrization actually has just 3 degrees of freedom ... the fourth parameter is determined by normalization.)

This works for some distributions, but not all.

Kernal Density Estimation

A kernel density estimator (in 1D) is defined by

$$\hat{g}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h} \right)$$

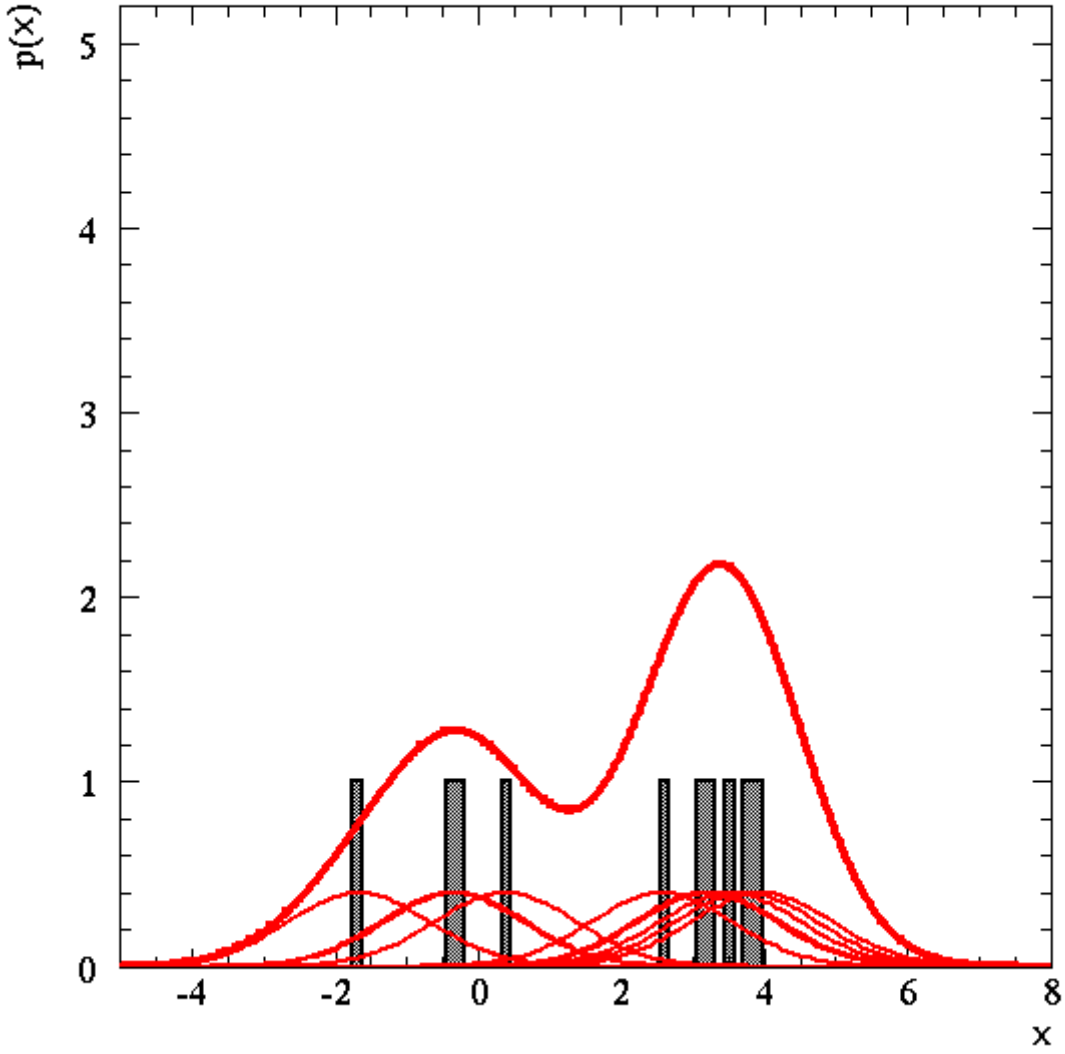
Here x_i is the i^{th} of N data points. K(x) is called the kernel function, and is normalized to one (and so $\hat{g}(x)$ is also normalized to 1). The parameter *h* is called the "bandwidth", and scales the width of the kernel.

Essentially this just means placing a smooth function at the location of each data point and then summing the result.

Kernal Density Estimation

It's intuitive (I hope) that in the limit that the number of data points goes to infinity and the bandwidth goes to zero, then the distribution should approach the true underlying distribution. Compare to:

$$f(x) = \int dy\, f(y)\, \delta(x - y)$$



Choice of kernel

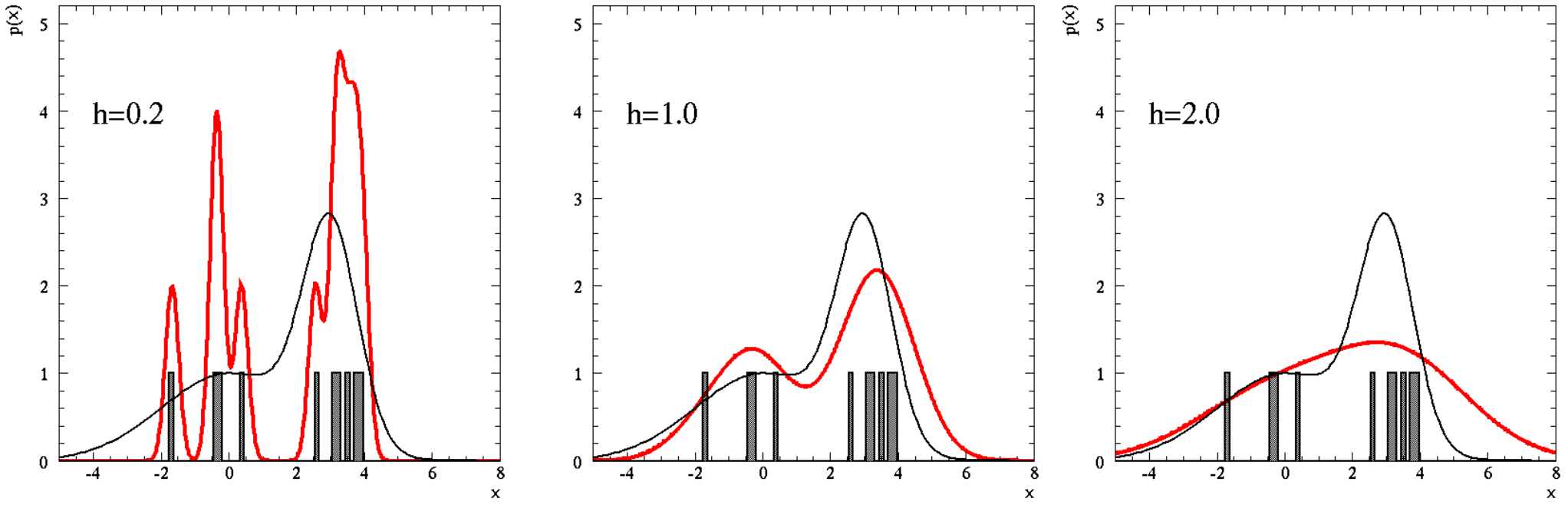
Some common choices of kernel function:

- the normal distribution: N(0,1)
- the uniform rectangle: =0.5 for |x|<1, else =0
- the Epanechnikov kernel: =¾(1-x^2) for |x|<1, else =0

It can be shown that the Epanechnikov kernel is "optimal" in 1D in a particular sense, but on the other hand it turns out that it makes very little difference which kernel you use. Many people like to use normal (Gaussian) distributions for simplicity.

Choice of bandwidth

Almost all of the art of KDE is in the choice of bandwidth.



Choosing an appropriate bandwidth is the key. Of course to do this perfectly you'd need to know what the underlying distribution is.

The figure on left is undersmoothed. On the right it's oversmoothed.

How to optimally choose bandwidth

The basic principles are similar to those behind choosing the best binning for a histogram.

Narrow bandwidth: allows you to sample narrow features of the distribution, but leaves you susceptible to random scatter. Generally the more points you have, the narrower you can make the bandwidth.

Wide bandwidth: smooths out statistical fluctuations, but may bias the result by smearing out narrower features.

How to optimally choose bandwidth: pointwise

Pointwise error (Here the brackets denote averaging over datasets):

$$\hat{g}_n(x) - g(x) = [\langle \hat{g}_n(x) \rangle - g(x)] + [\hat{g}_n(x) - \langle \hat{g}_n(x) \rangle]$$

Bias term: average difference between estimated and true function

Variance term: difference between estimate for this dataset and average estimate

$$Bias\, B_h(x) \approx \frac{h^2}{2} \sigma_K^2 \nabla^2 g(x)$$

$$Std\ dev\ E_n \approx \sqrt{\frac{\mu_K \cdot g(x)}{nh^d}} N(0,1)$$

$$\sigma_K^2 = \int \|x\|^2 K(x)\, dx$$

$$\mu_K = \int K^2(x)\, dx$$

To minimize typical pointwise error, minimize $B_h^{\ 2} + E_n^{\ 2}$ as function of h

How to optimally choose bandwidth: MISE

Rather than focusing on one value of x, minimize the mean integrated square error.

$$\int \langle (\hat{g}_n(x) - g(x))^2 \rangle dx = \int B_h^2(x)\,dx + \int Var(E_n(x))\,dx$$

$$\approx \frac{h^4}{4}\sigma_K^4 \int |\nabla^2 g(x)|^2 dx + \frac{\mu_K}{nh^d}$$

This is minimized by

$$h_{opt} = \left(\frac{4\mu_K}{\sigma_K^4 \int |\nabla^2 g(x)|^2 dx} \cdot \frac{1}{n} \right)^{\frac{1}{d+4}}$$

d=1 when estimating a 1D function.

The chicken and egg problem

Optimal bandwidth depends on true g(x), which you don't know. As a crude approximation, can use Silverman's rule of thumb, which works great for approximately Gaussian distributions:

$$\hat{h} = \left(\frac{4\hat{\sigma}^5}{3n} \right)^{1/5} \approx 1.06\,\hat{\sigma}\,n^{-1/5}$$

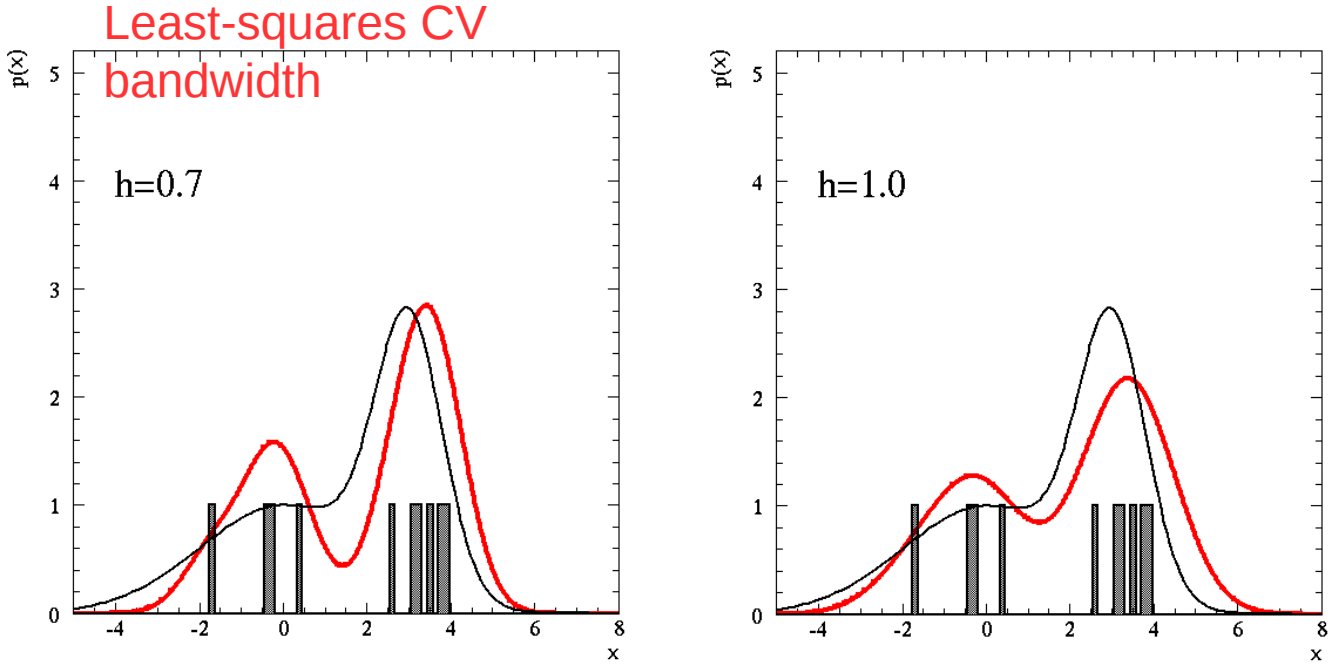
Here $\hat{\sigma}$ is the standard deviation estimated from the data.

This rule of thumb may be reasonable for distributions that look Gaussian, but will have issues when the distribution is double-peaked. For example, for the plots given earlier this formula gives h=2.09, which is definitely oversmoothed.

Least-squared cross-validated bandwidth

$$CV(h) = R(\hat{g}_h) - \frac{2}{n}\sum_{j=1}^{n} \hat{g}_{j,h}(X_j)$$

Pick the *h* that minimizes this function. Here X_j is the j^{th} data point and $\hat{g}_{j,h}$ is the kernel estimator derived from the data with X_j deleted. Also, $R(g) = \int g^2(x)\,dx$.



Other bandwidth approaches

There are a wide number of alternate approaches for data-driven bandwidth selection, which often work better than least-squares cross validation but which are harder to explain. See for example " Comparison of Data-Driven Bandwidth Selections" by Byeong U. Park and J.S. Marron (Journal of the American Statistical Association, Vol. 85, No. 409 (March 1990), pp. 66-72.

Common ones are "Park and Marron's plug-in selector" and the "Sheather and Jones plug-in", which attempt to do KDE estimates for the g″(x) which appears in the optimal bandwidth formula.

Pointwise error on KDE

At any given value of x, we've already seen an estimate for the statistical scatter in the estimate.

$$Std\ dev\ E_n \approx \sqrt{\frac{\mu_K \cdot g(x)}{nh^d}}\, N(0,1)$$

In the plug-in approach, just replace *g(x)* with $\hat{g}(x)$.

$$g(x) = \hat{g}(x) \pm \sqrt{\frac{\mu_K \cdot \hat{g}(x)}{nh^d}}$$

Alternatively use bootstrap method to get error.

Funtionwise error on KDE

The previous formulas are confidence intervals at a given value of x. If you want a confidence band that includes the entire function (over its region of domain) at some confidence interval, the best approach is probably to bootstrap it. See for example “A Tutorial on Kernel Density Estimation and Recent Advances”, by Yen-Chi Chen. (arXiv:1704.03924v2)

Adaptive bandwidth

A more sophisticated approach is to adjust the bandwidth for each data point. This is called “adaptive kernel density”. In the “pointwise estimator” implementation each point gets its own bandwidth:

$$\hat{g}(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h_i}K\left(\frac{x - x_i}{h_i}\right)$$

The idea is that in areas where the data samples have lower density you want to have a wider bandwidth, while you want narrower bandwidths in regions where the points are closer together.

Adaptive bandwidth: implementation

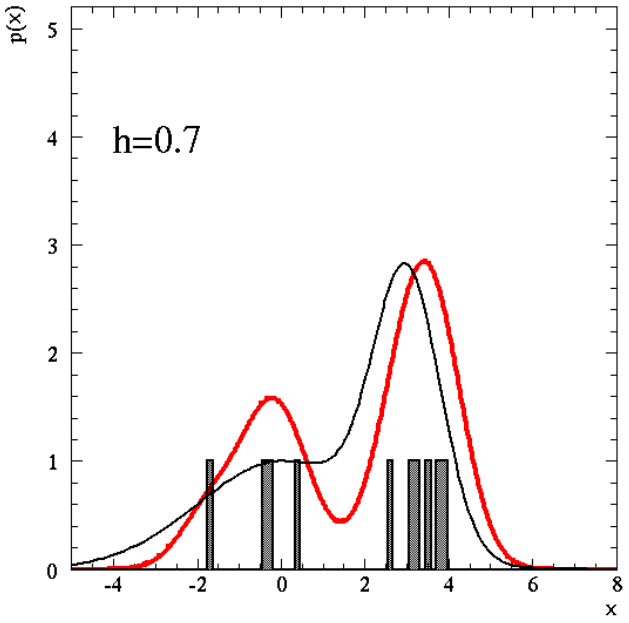
First do your favourite fixed bandwidth KDE, to get an initial guess for *g(x)*. Call this $g_h(x)$, where *h* is the fixed bandwidth. Then:

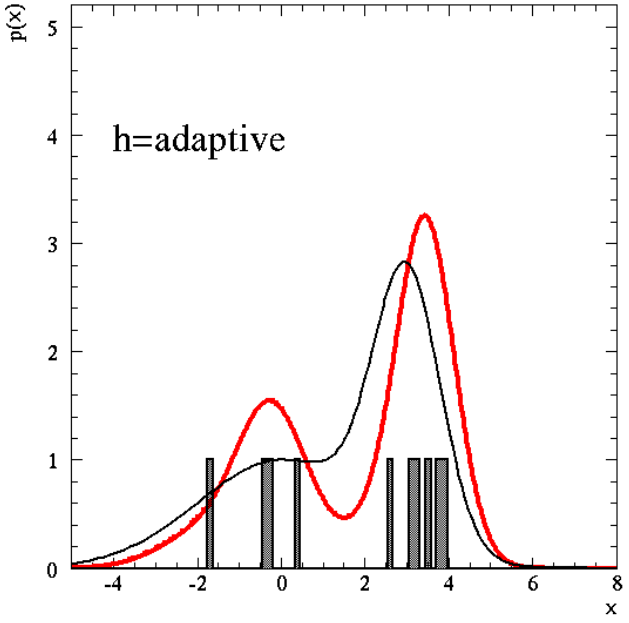
$$G \equiv \left(\prod_{i=1}^{N} g_h(x_i) \right)^{1/N}$$

$$h_i = h \sqrt{\frac{G}{g_h(x_i)}}$$

$$\hat{g}(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h_i} K\left(\frac{x - x_i}{h_i} \right)$$

Adaptive bandwidth: results

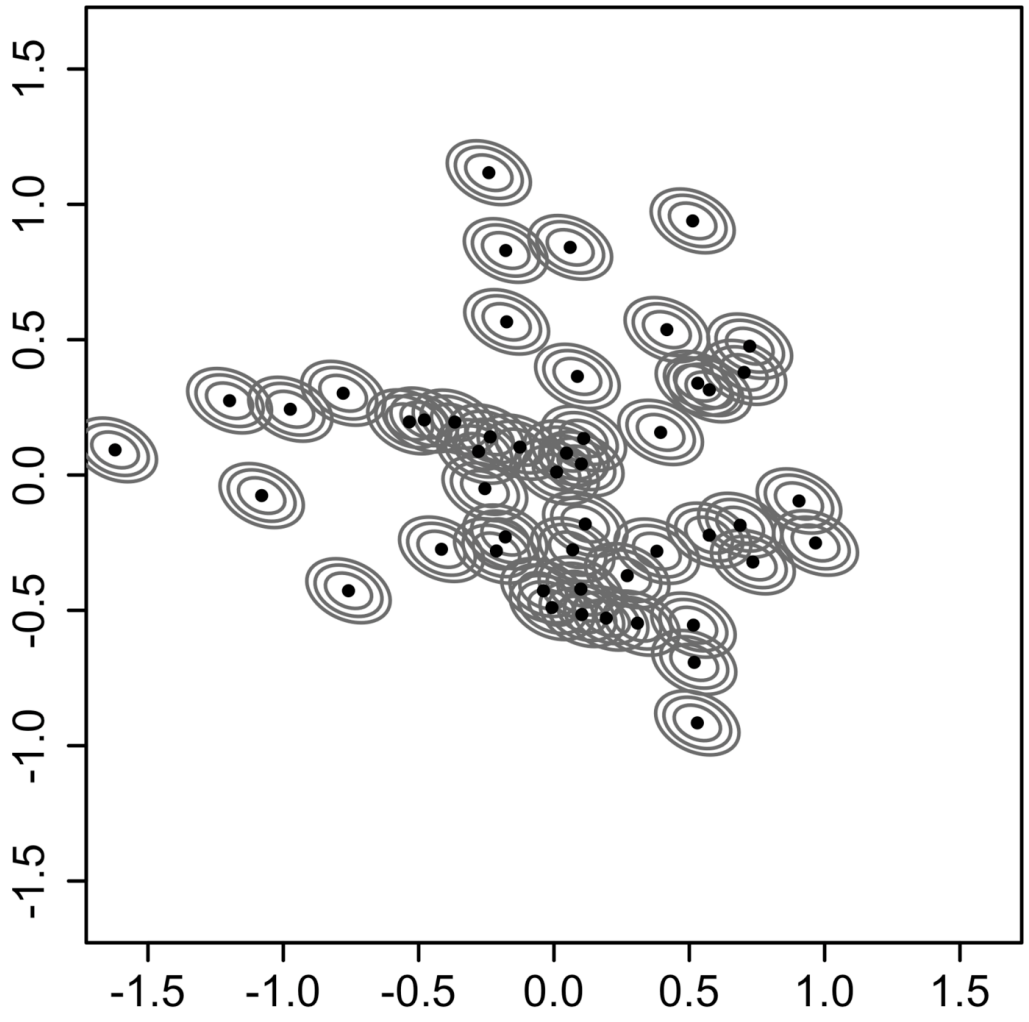




Data Point	Best fixed bandwidth (least squared CV)	Adaptive bandwidth
-1.67	0.70	1.09
-0.35	0.70	0.77
-0.34	0.70	0.77
0.37	0.70	0.86
2.56	0.70	0.72
3.16	0.70	0.58
3.23	0.70	0.57
3.52	0.70	0.59
3.74	0.70	0.59
3.96	0.70	0.64

Is adaptive bandwidth actually better in this case? Clearly better on the tails, but a little worse in the meat of the distribution. MISE is actually worse.

Multi-dimensional KDE



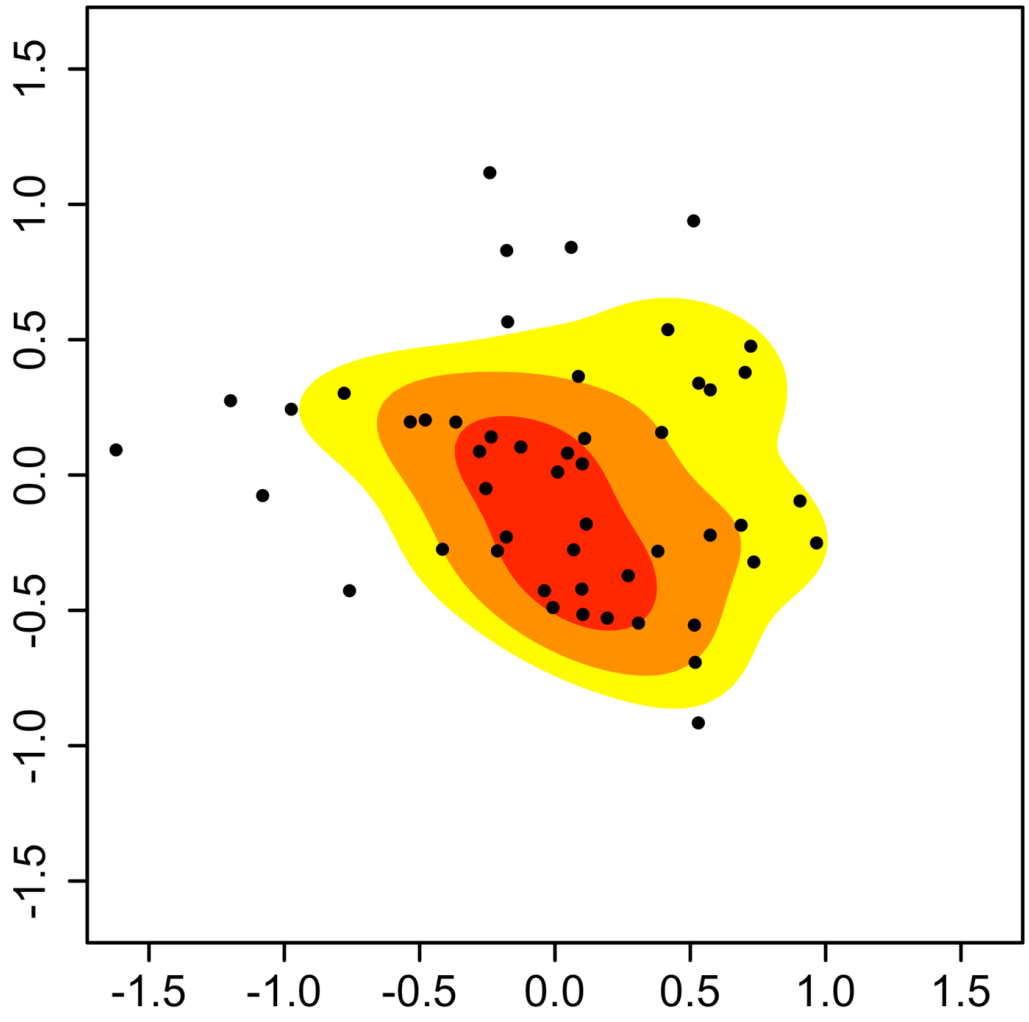
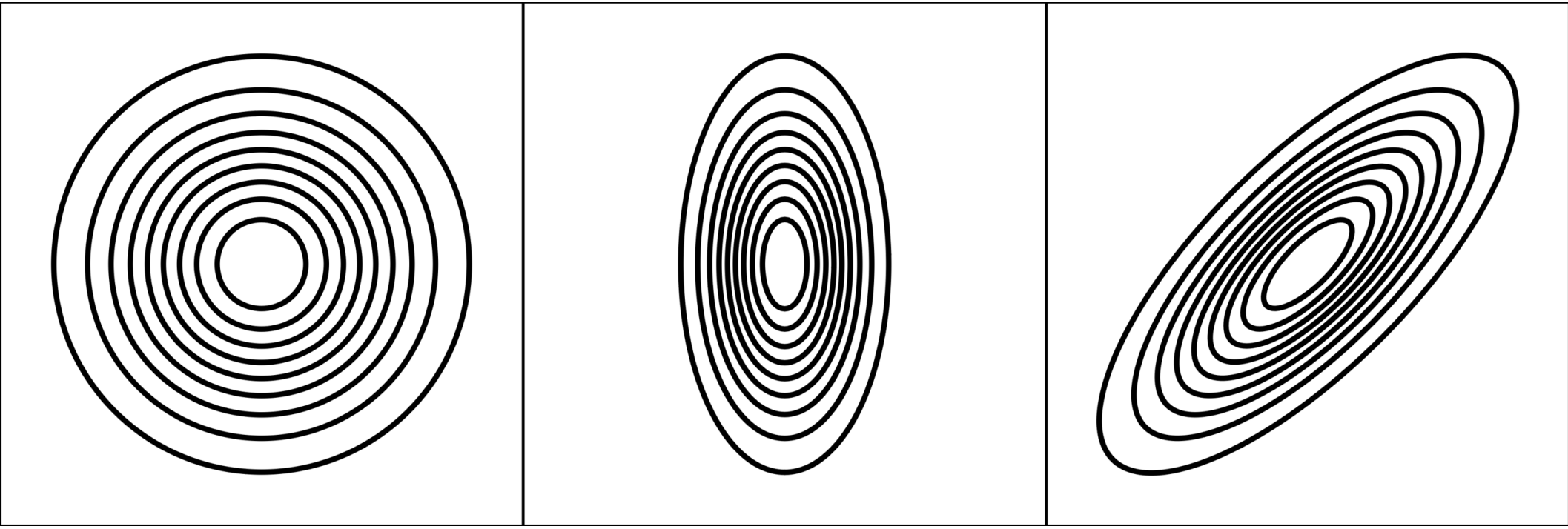


Figure by Drleft (talk) 00:04, 16 September 2010 (UTC) - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=11500097

Kernel choice for multi-dimensional KDE



$H = hI$

$H_{ij} = h_i \delta_{ij}$

$H =$ symmetric positive definite matrix

Figure credit: Drleft [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], from Wikimedia Commons

Kernel choice for multi-dimensional KDE

$$K\left(\vec{x}\right)=\frac{1}{\left(2\pi\right)^{d/2}\sqrt{\det H}}e^{-\frac{1}{2}\vec{x}^{T}\cdot H^{-1}\cdot\vec{x}}$$

The bandwidth matrix **H** defines this multi-dimensional Gaussian kernel. It is a positive definite symmetric matrix. Analogous matrices exist for other kernel choices, but as usual the bandwidth is much more important than the functional form of the kernel.

As before we can try to optimize the choice of **H** by minimizing the MISE. Many of the same issue arise.

Silverman's rule of thumb (in this case a diagonal matrix):

$$H_{ij}=\delta_{ij}\sigma_{i}^{2}\left(\frac{4}{n\left(d+2\right)}\right)^{\frac{2}{d+4}}$$

Smoothed cross validation for multi-dim. KDE

Start with a pilot bandwidth matrix **G** (an initial guess). Then defined the smoothed cross validation statistic SCV as follows:

$$SCV\left(H\right)=n^{-1}\left(\det H\right)^{-1/2}R\left(K\right)+n^{-2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(K_{2H+2G}-2K_{H+2G}+K_{2G}\right)\left(\vec{x}_i-\vec{x}_j\right)$$

Here

$$R\left(K\right)=\int K\left(\vec{x}\right)^2 d\vec{x}=\left(4\pi\right)^{-d/2} \text{ for the normal kernel}$$

Minimize this expression as a function of the matrix **H** to get the best bandwidth matrix estimate.

This sounds like a nightmare---anyone have a ready-built function for minimizing as a function of a matrix? I guess parametrize it in terms of d standard deviations σ_i and ½d(d-1) correlation coefficients ρ_{ij}, and minimize with respect to all of these ½d(d+1) parameters.

Closing Thoughts

Multivariate KDE with variable bandwidth exists, but is beyond the scope of these notes.

Well-developed software packages exist. Don't re-invent the wheel. The Wikipedia articles on KDE and multivariate KDE give links to good resources.

A promising new approach is the "fastKDE" that is based on defining the kernel and bandwidth in Fourier space:

https://bitbucket.org/lbl-cascade/fastkde