Introduction to Kernel Density Estimation

A graphical tutorial

tommyod @ GitHub

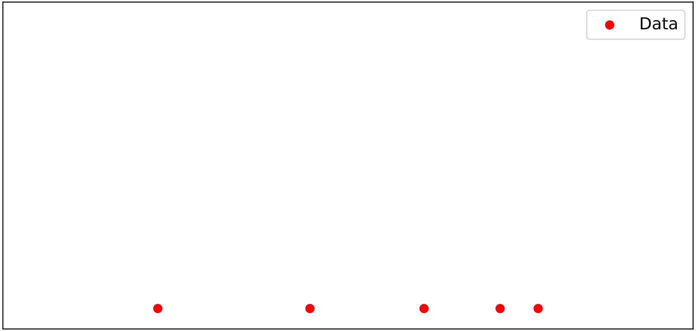
September 24, 2018

Introduction

What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

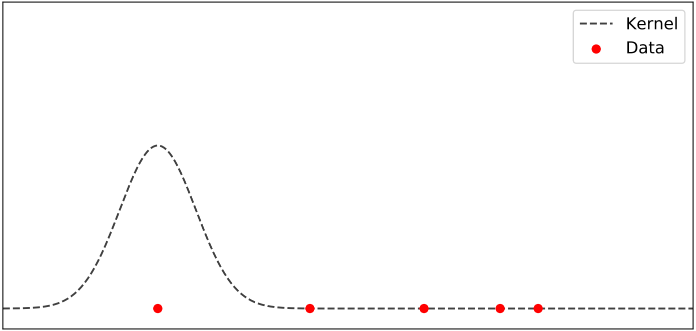
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

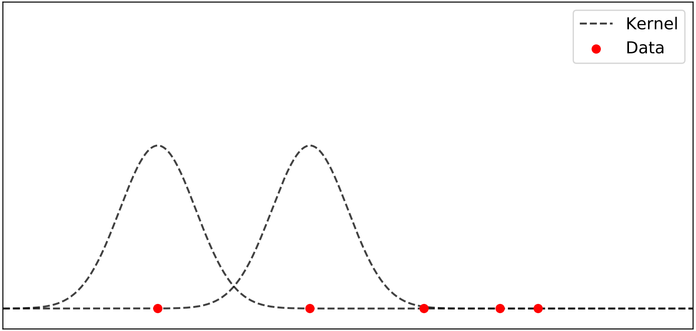
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

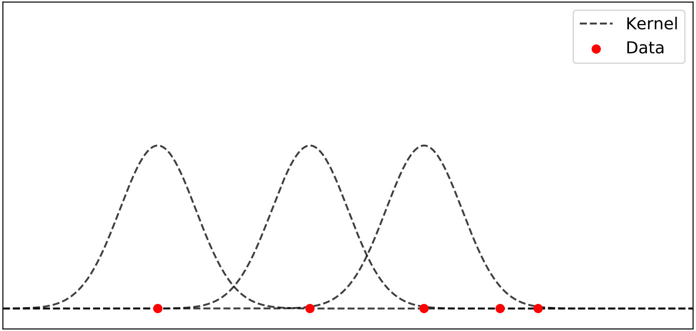
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

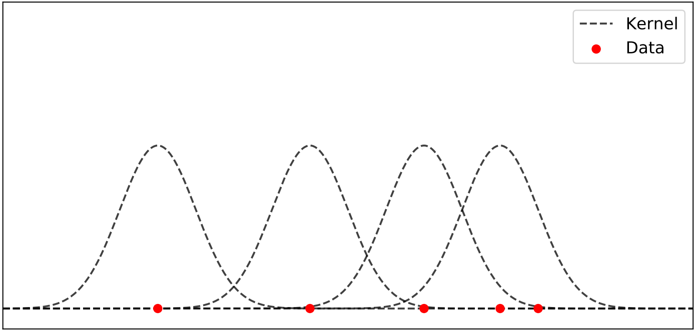
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

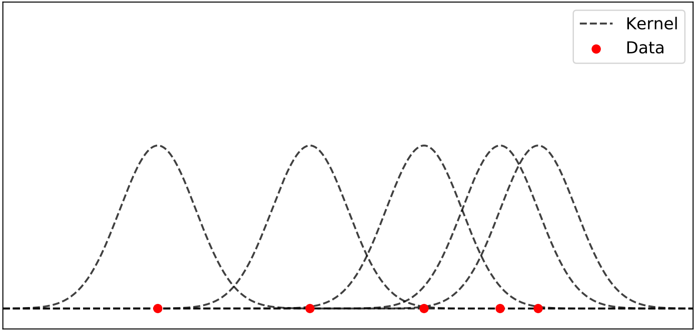
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

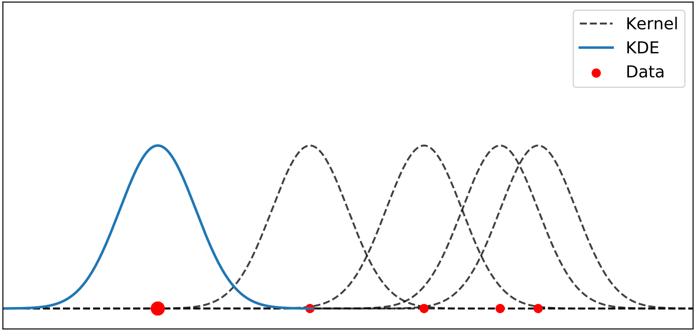
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

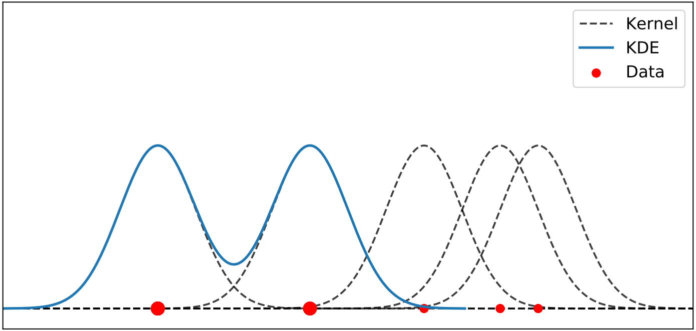
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

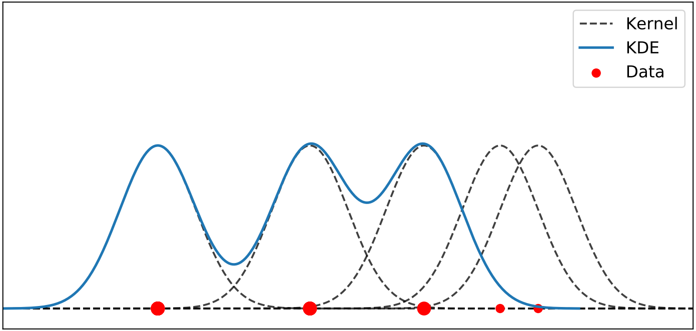
$$\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

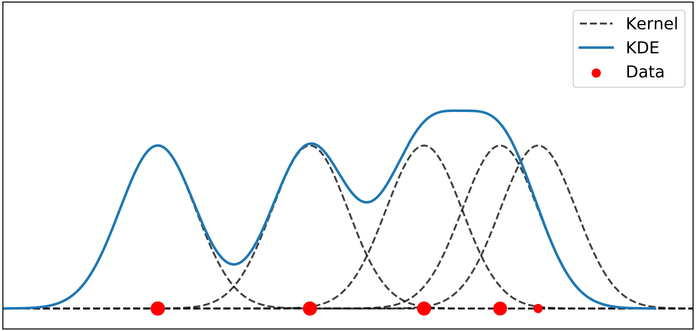
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

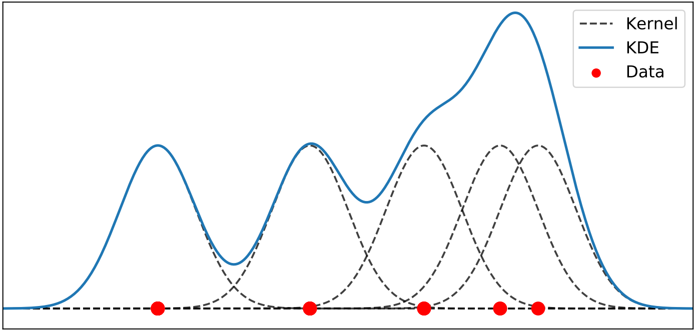
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



What is a kernel density estimate?

On every data point x_i, we place a kernel function K. The kernel density estimate is

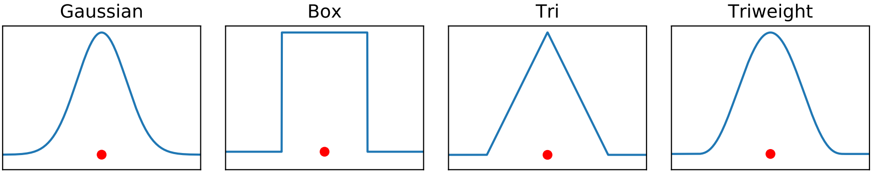
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



Choice of kernel

The kernel function K is typically

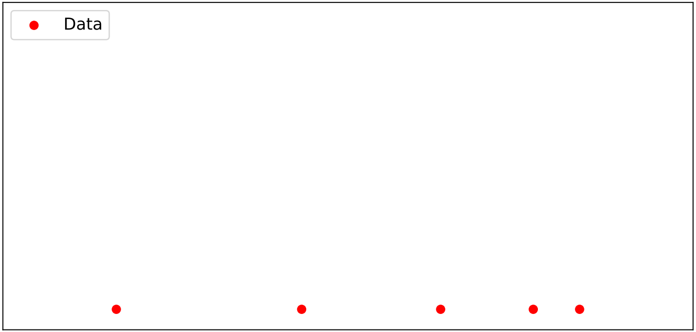
- everywhere non-negative: $K(x) \geq 0$ for every x
- symmetric: $K(x) = K(-x)$ for every x
- decreasing: $K'(x) \leq 0$ for every $x > 0$.



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

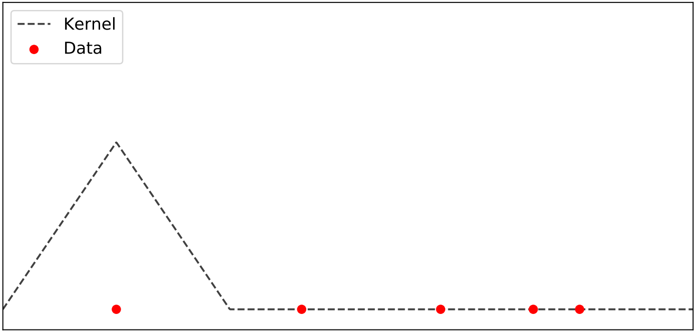
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

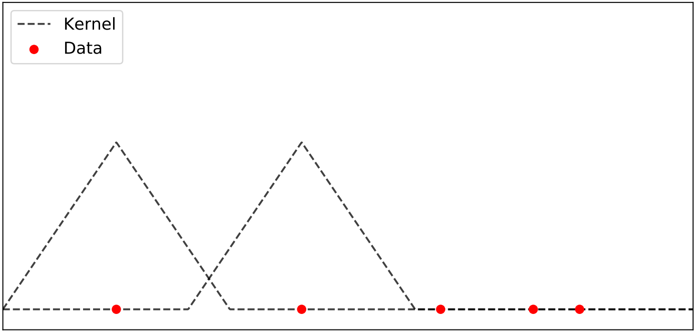
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

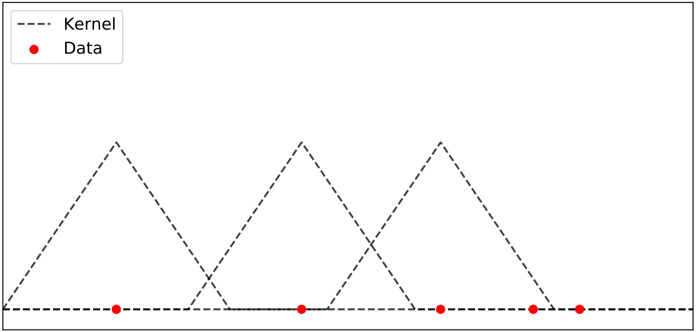
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

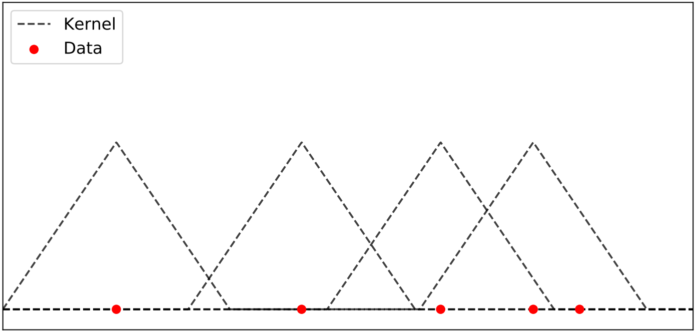
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

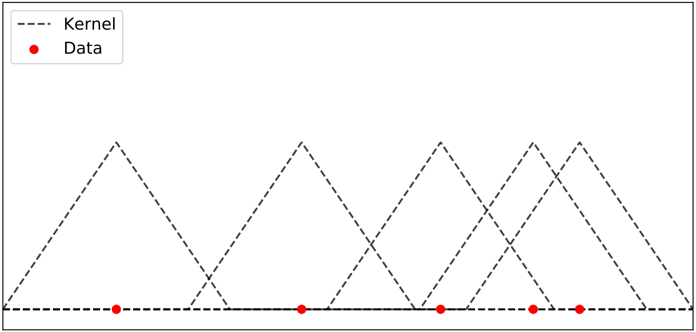
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

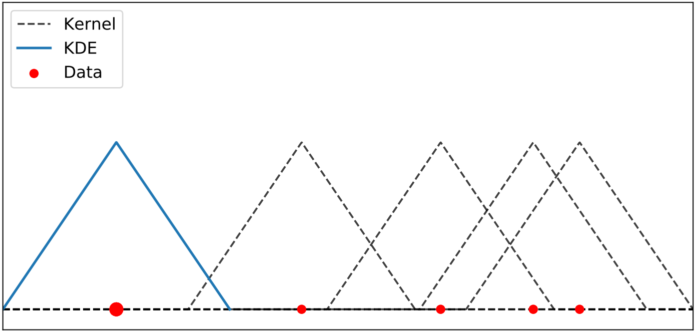
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

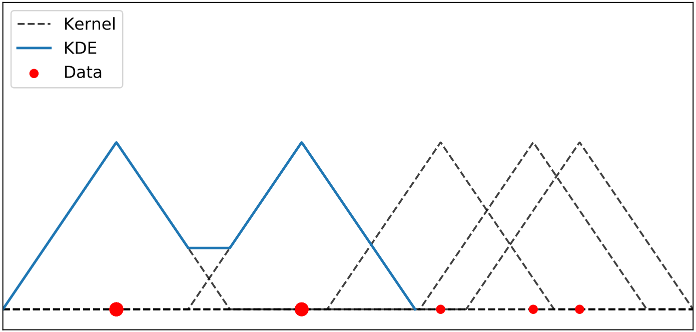
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

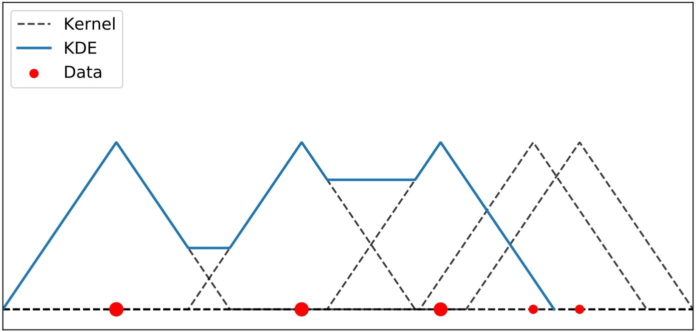
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

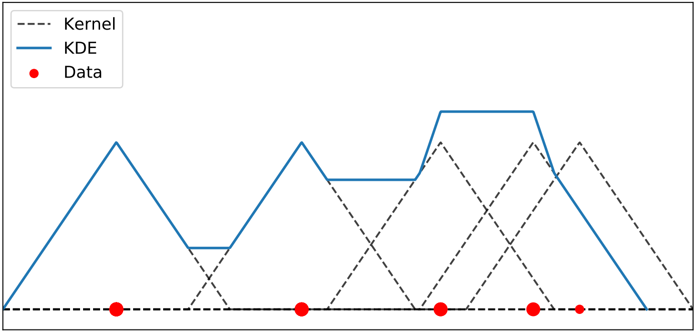
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

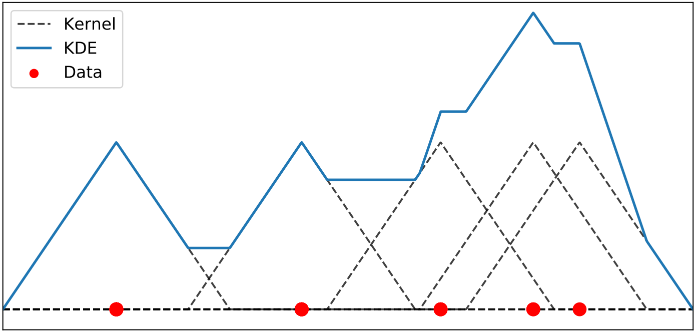
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of kernel

The *triangular* kernel (or *linear* kernel) is given by

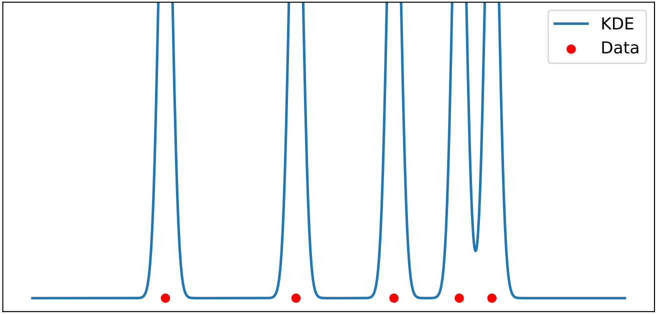
$$f(x) \propto \max(1 - |x|, 0).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

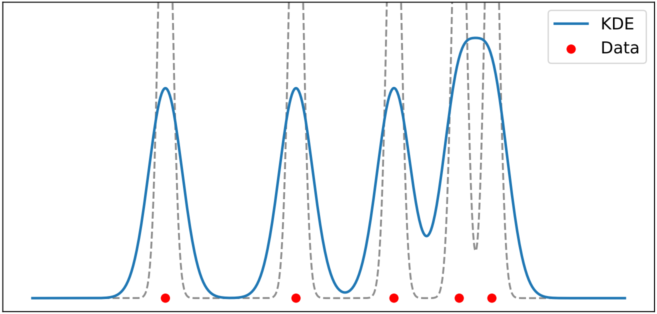
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

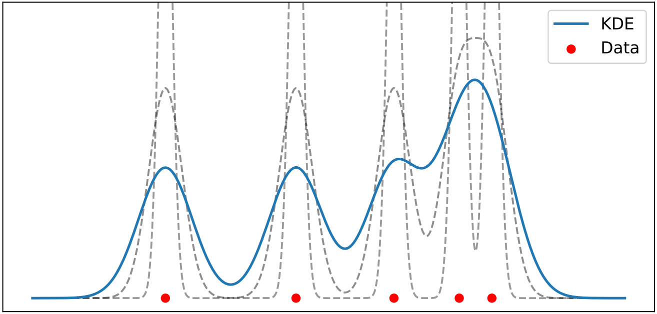
$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

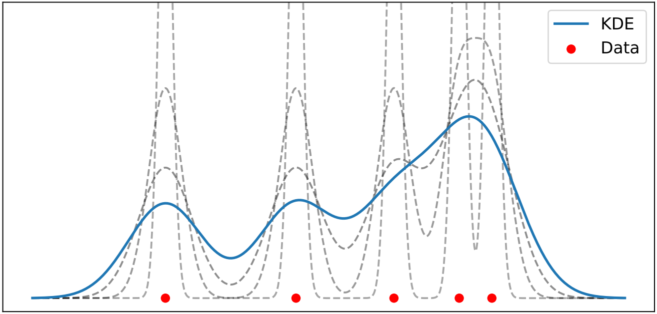
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

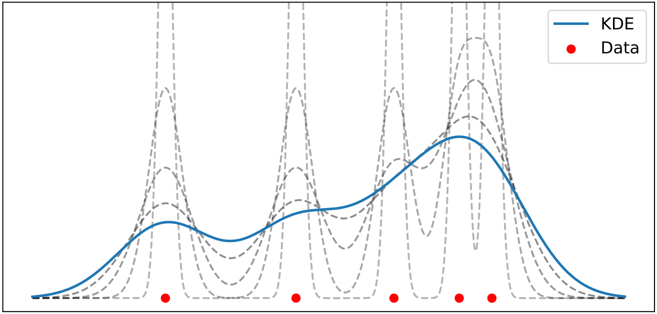
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

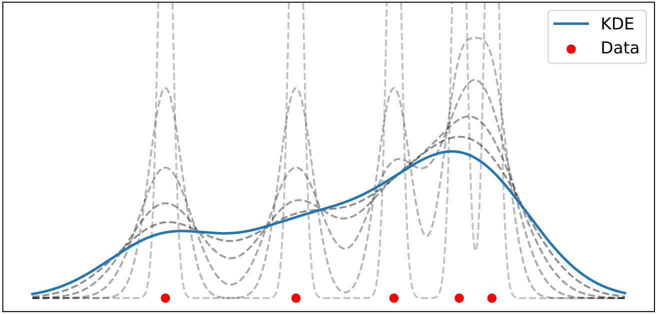
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

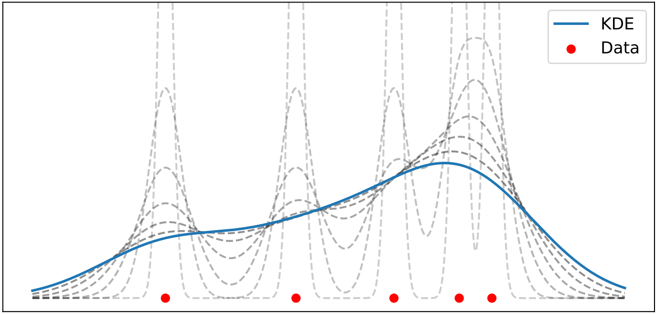
$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x-x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

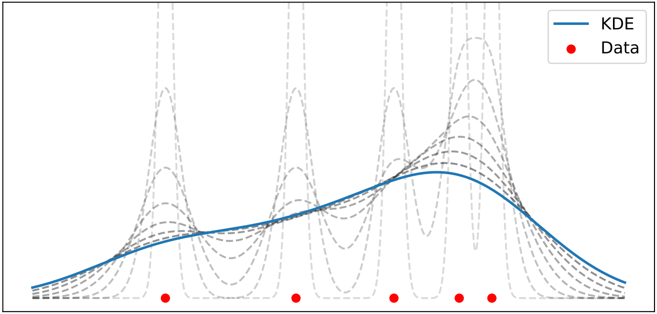
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

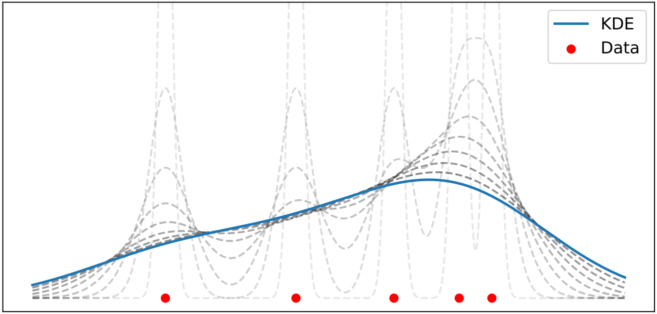
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

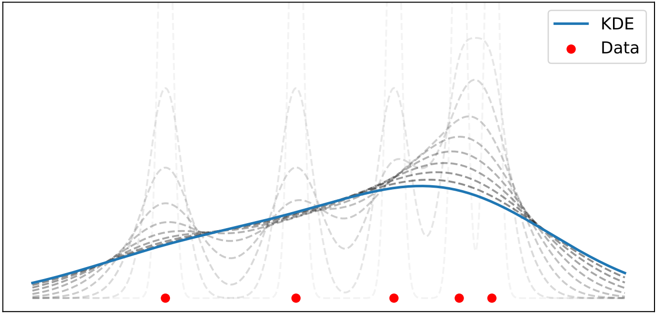
$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Choice of bandwidth

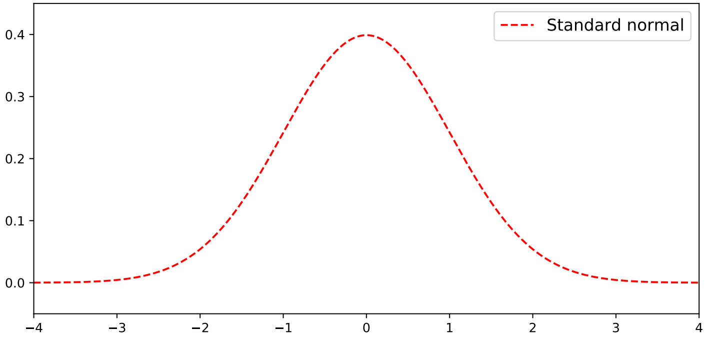
We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

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



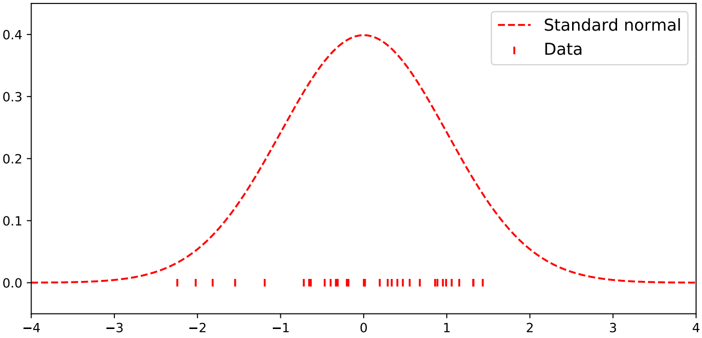
Choice of bandwidth - Silverman

Silverman's rule of thumb computes an optimal h by assuming that the data is normally distributed. Good starting point in many cases.



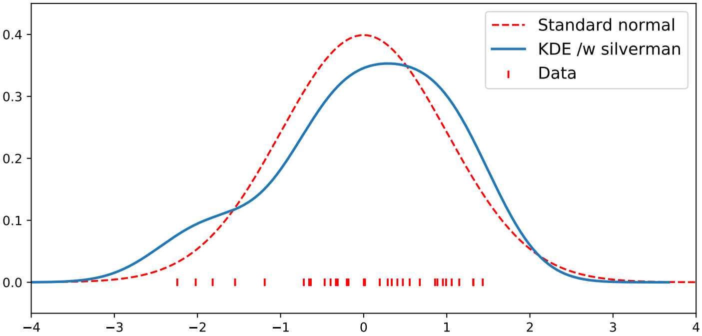
Choice of bandwidth - Silverman

Silverman's rule of thumb computes an optimal h by assuming that the data is normally distributed. Good starting point in many cases.



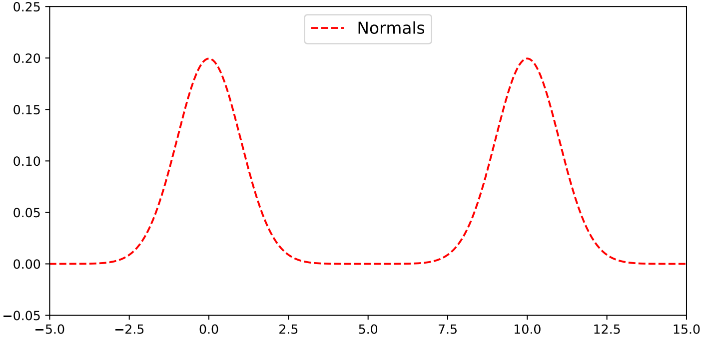
Choice of bandwidth - Silverman

Silverman's rule of thumb computes an optimal h by assuming that the data is normally distributed. Good starting point in many cases.



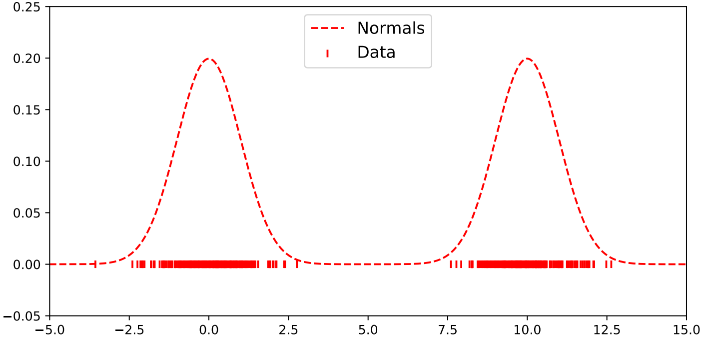
Choice of bandwidth - ISJ

The *Improved Sheather Jones* (ISJ) algorithm is more robust with respect to *multimodality*.



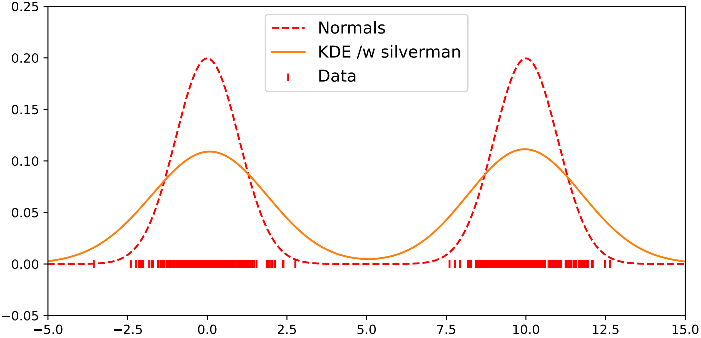
Choice of bandwidth - ISJ

The *Improved Sheather Jones* (ISJ) algorithm is more robust with respect to *multimodality*.



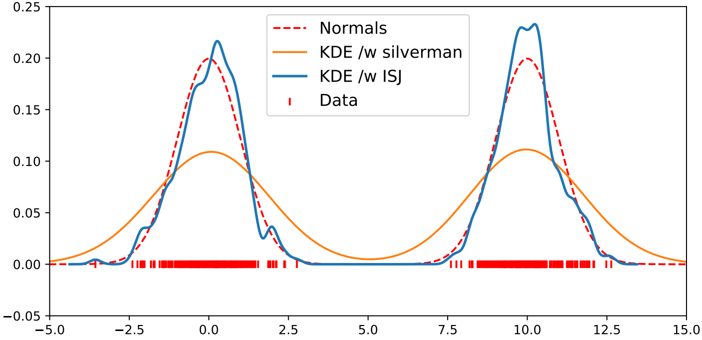
Choice of bandwidth - ISJ

The *Improved Sheather Jones* (ISJ) algorithm is more robust with respect to *multimodality*.



Choice of bandwidth - ISJ

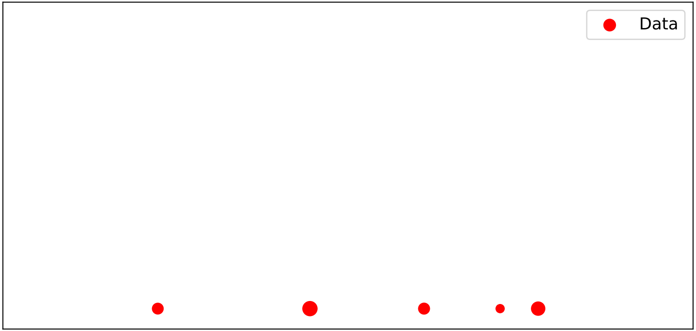
The *Improved Sheather Jones* (ISJ) algorithm is more robust with respect to *multimodality*.



Weighting data

It's possible to add weights w_i to data points x_i by writing

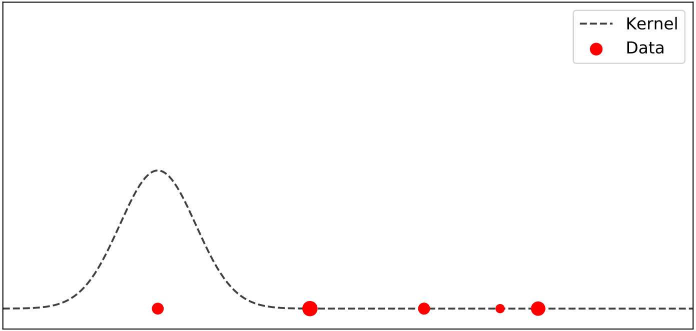
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

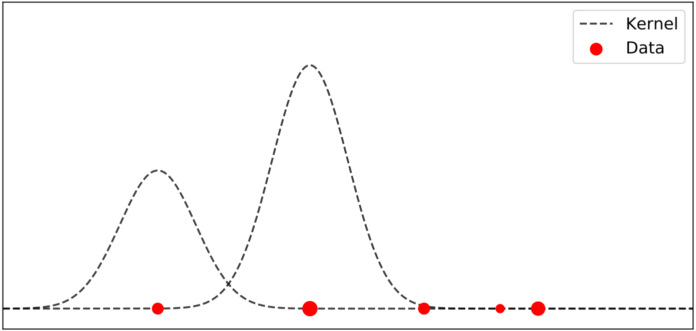
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

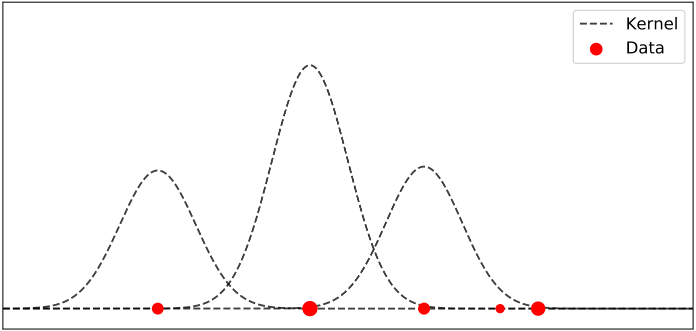
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

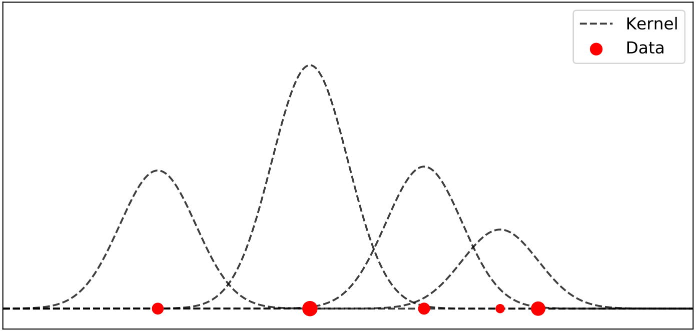
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

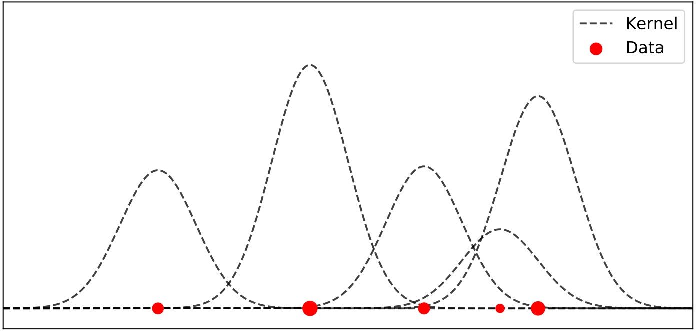
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K \left(\frac{x - x_i}{h} \right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

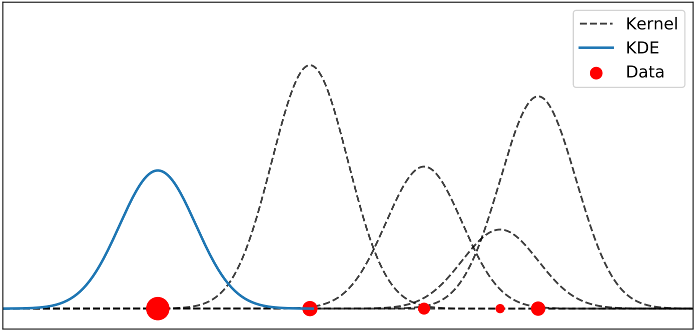
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

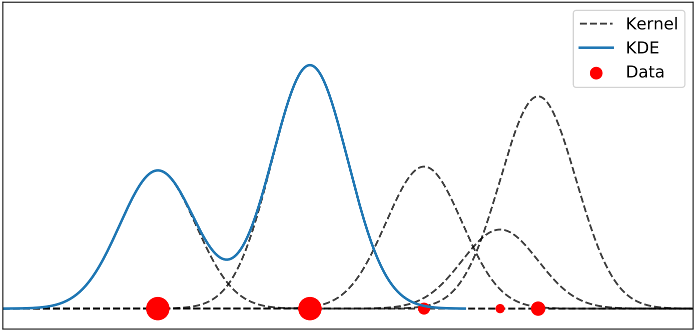
$$\hat{f}(x) = \frac{1}{h}\sum_{i=1}^{N} w_i K\left(\frac{x-x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

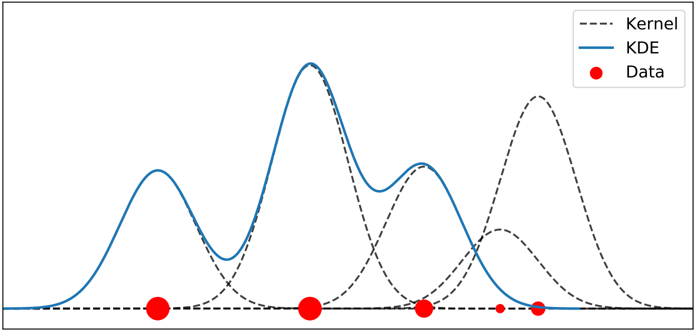
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

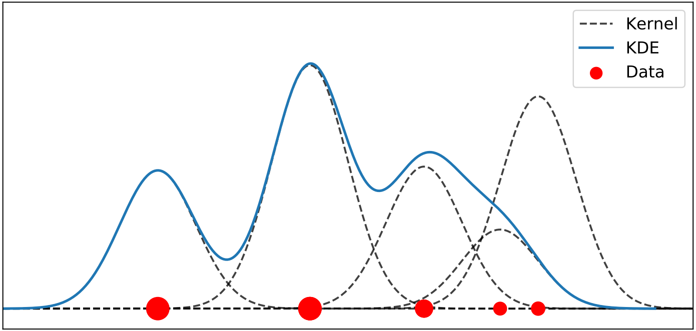
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

It's possible to add weights w_i to data points x_i by writing

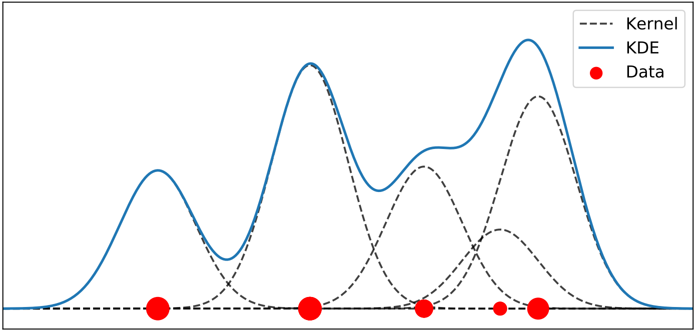
$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



Weighting data

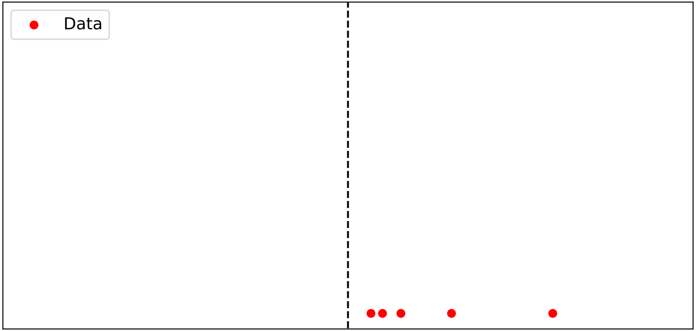
It's possible to add weights w_i to data points x_i by writing

$$\hat{f}(x) = \frac{1}{h} \sum_{i=1}^{N} w_i K\left(\frac{x - x_i}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



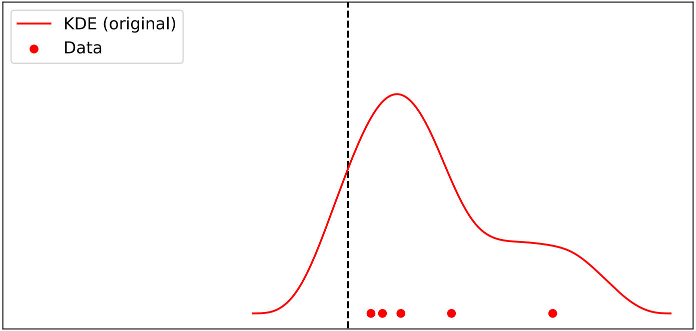
Bounded domains

A simple trick to overcome bias at boundaries is to mirror the data. This ensures that $\hat{f}'(x) = 0$ at the boundary.



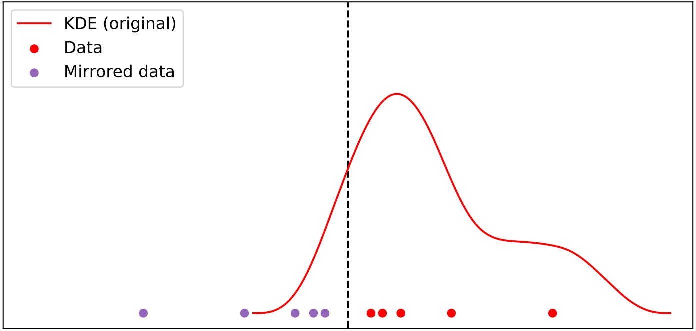
Bounded domains

A simple trick to overcome bias at boundaries is to mirror the data. This ensures that $\hat{f}'(x) = 0$ at the boundary.



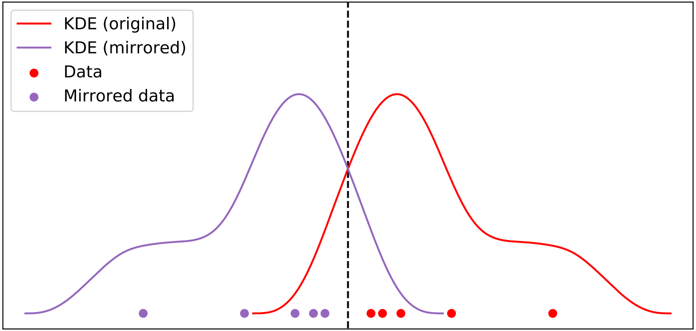
Bounded domains

A simple trick to overcome bias at boundaries is to mirror the data.
This ensures that $\hat{f}'(x) = 0$ at the boundary.



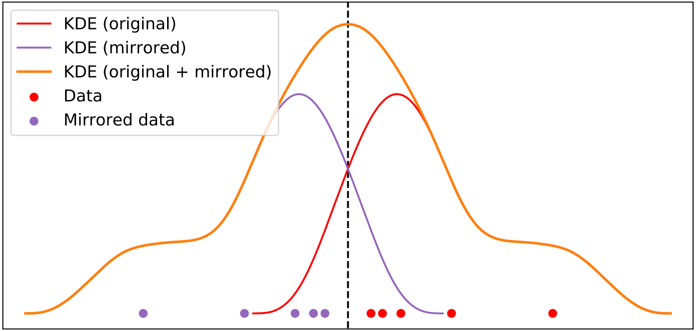
Bounded domains

A simple trick to overcome bias at boundaries is to mirror the data. This ensures that $\hat{f}'(x) = 0$ at the boundary.



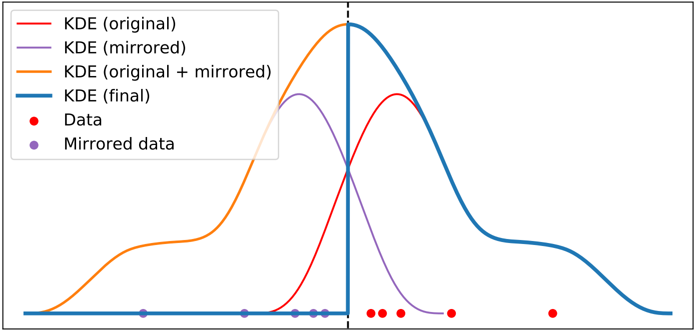
Bounded domains

A simple trick to overcome bias at boundaries is to mirror the data. This ensures that $\hat{f}'(x) = 0$ at the boundary.



Bounded domains

A simple trick to overcome bias at boundaries is to mirror the data. This ensures that $\hat{f}'(x) = 0$ at the boundary.

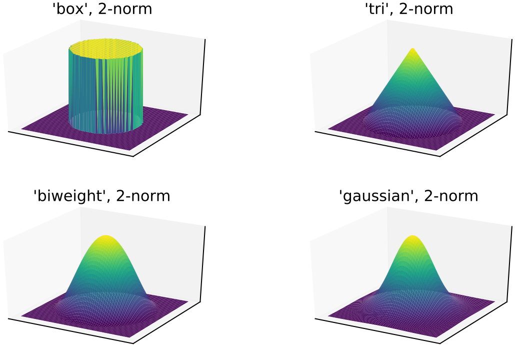


Extension to d dimensions

Kernels in 2D

An approach to d-dimensional estimates is to write

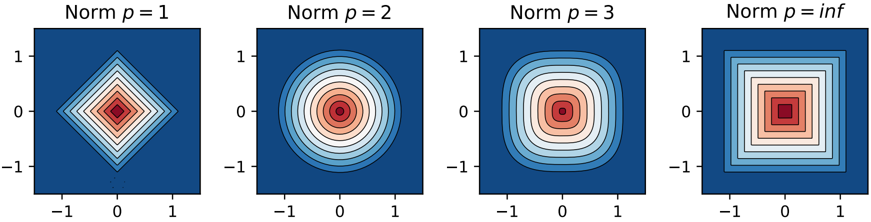
$$\hat{f}(x) = \frac{1}{h^d} \sum_{i=1}^{N} w_i K\left(\frac{\|x - x_i\|_p}{h}\right), \text{ where } \sum_{i=1}^{N} w_i = 1.$$



The effect of norms

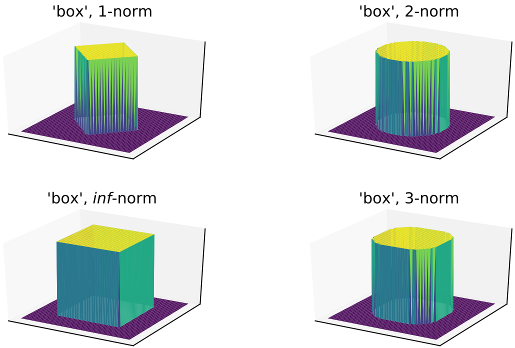
The choice of norm comes in to play when $d \geq 2$, the p-norm is

$$\|x\|_p := \left(\sum_{i=1} |x_i|^p \right)^{1/p}.$$



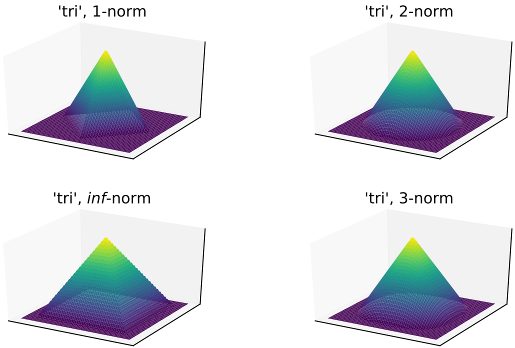
The effect of norms

The shape of kernel functions in higher dimensions depend on the value of p in the p norm.



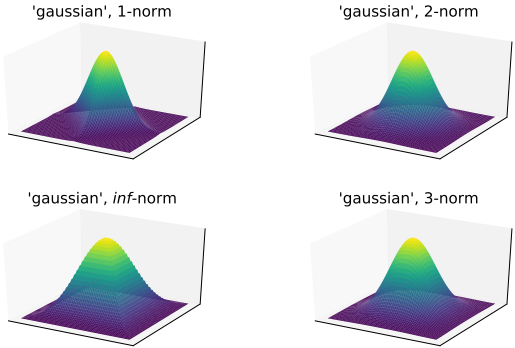
The effect of norms

The shape of kernel functions in higher dimensions depend on the value of p in the p norm.



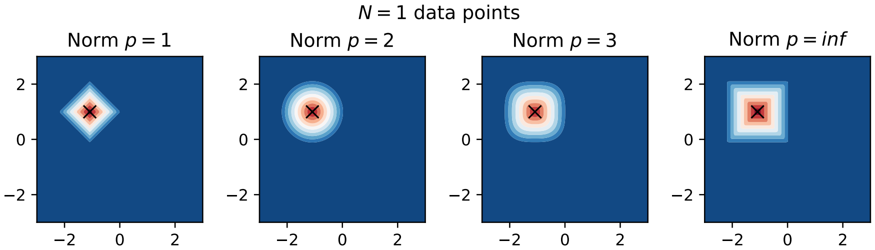
The effect of norms

The shape of kernel functions in higher dimensions depend on the value of p in the p norm.



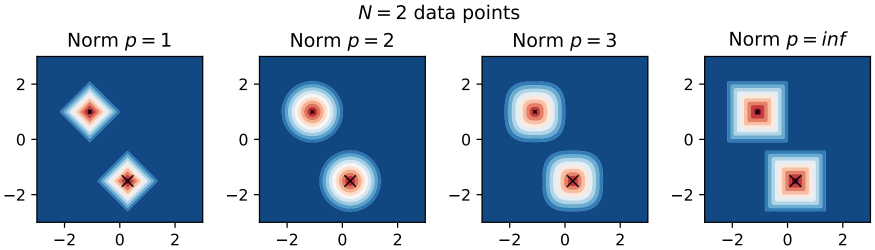
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



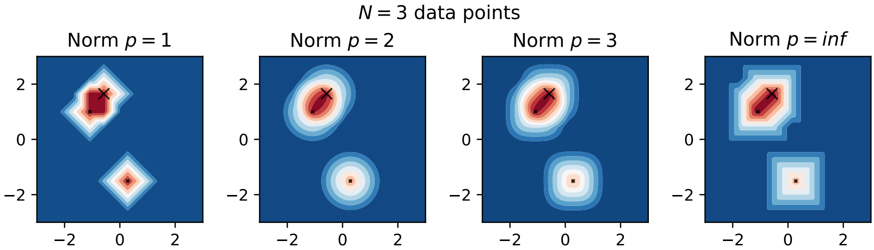
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



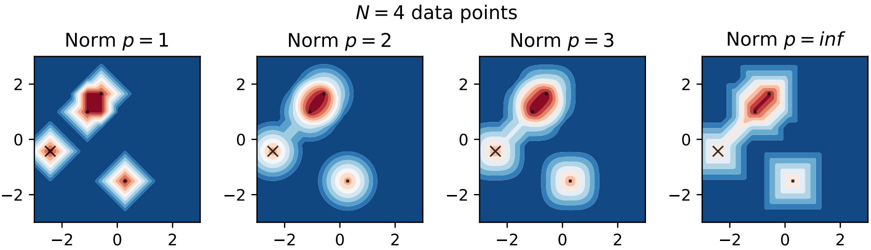
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



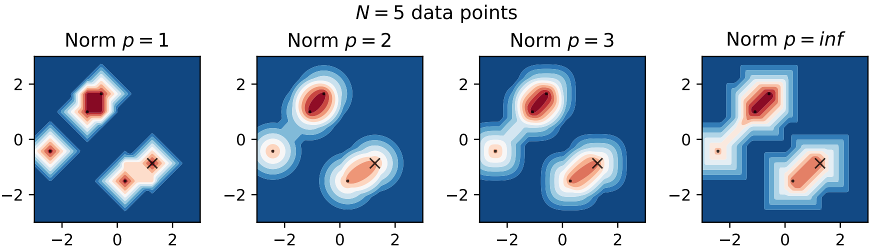
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



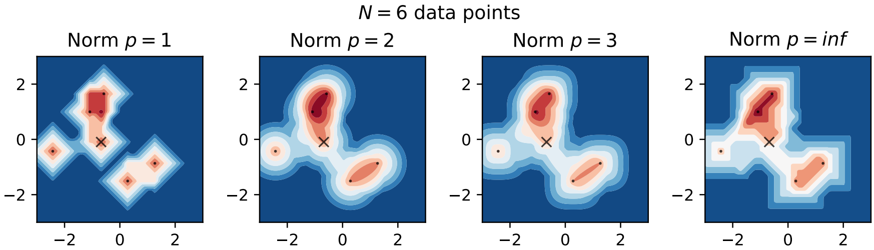
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



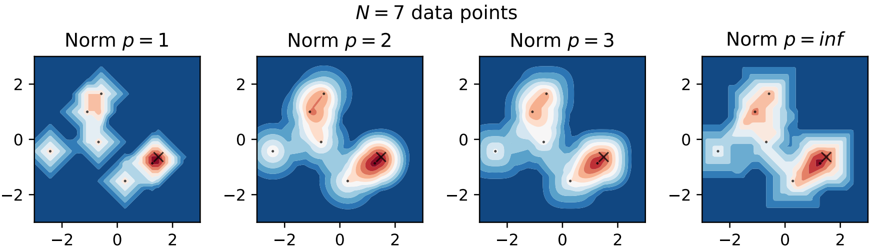
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



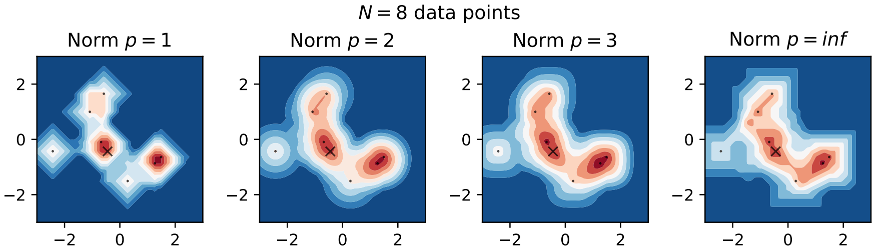
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



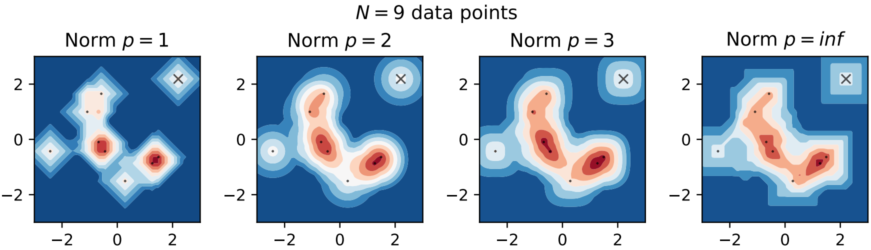
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



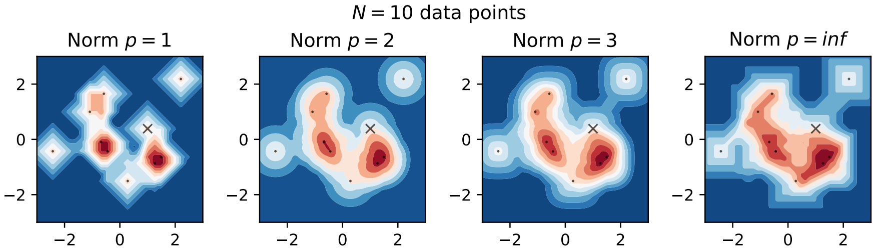
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



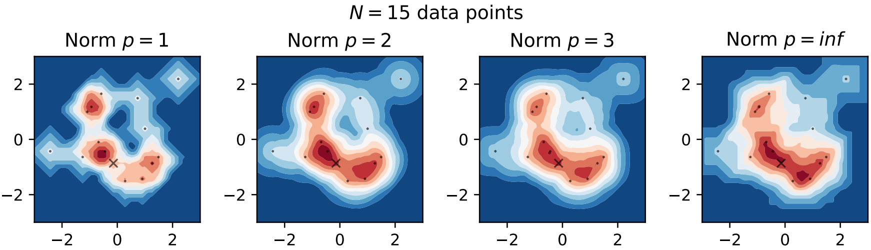
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



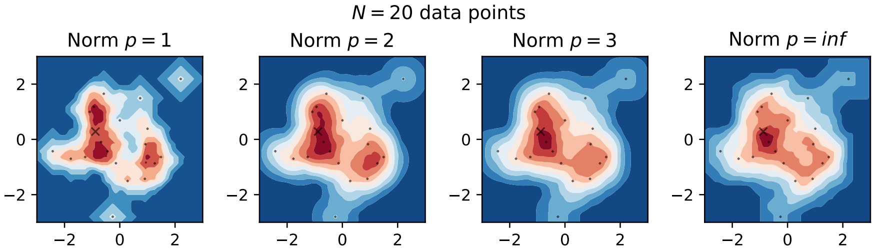
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



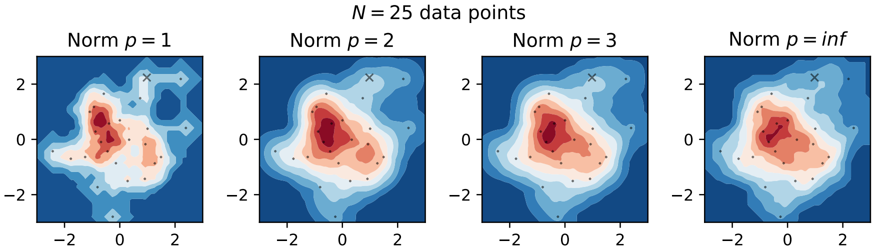
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



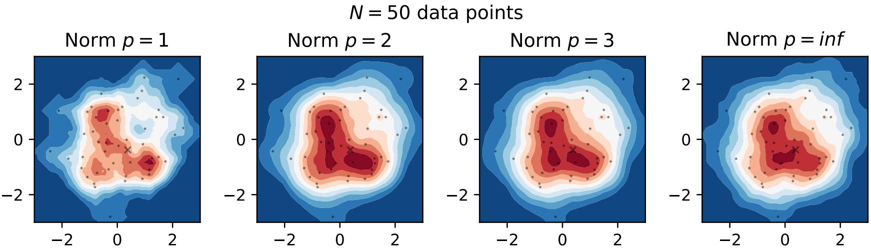
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



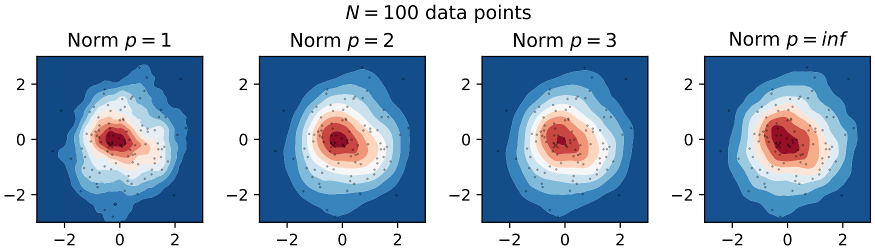
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



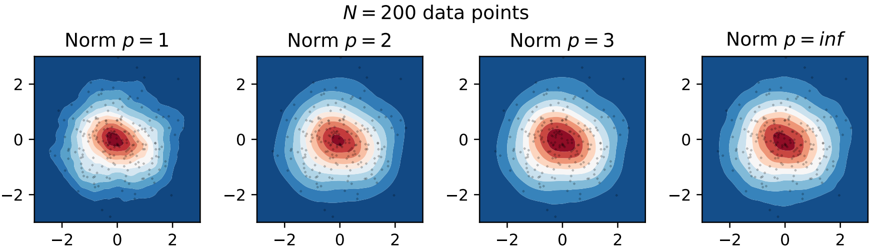
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



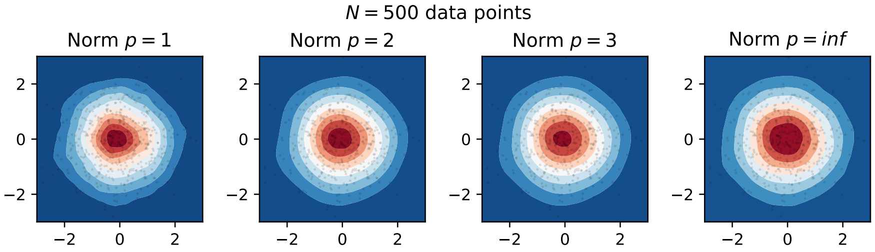
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



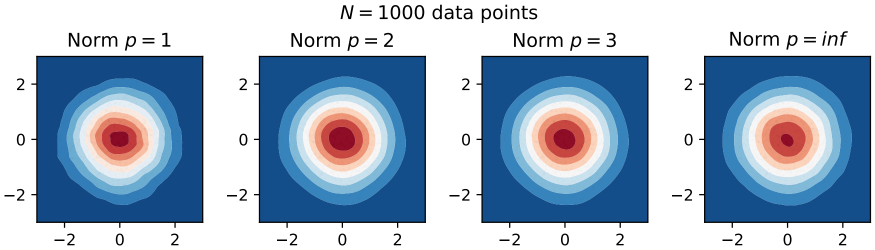
Example with data

As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



Example with data

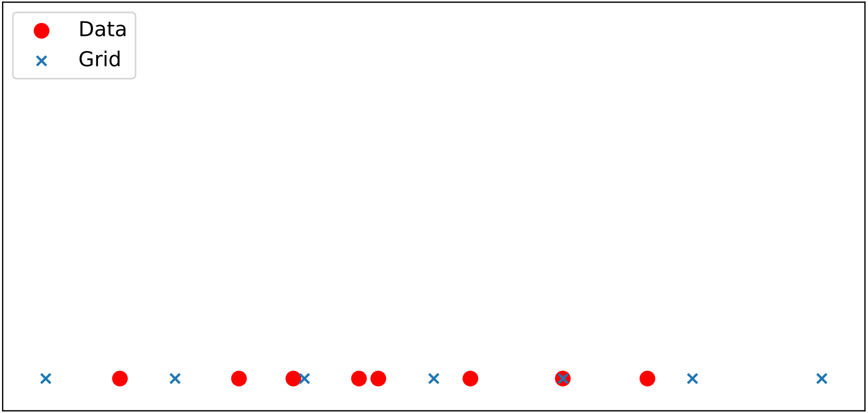
As the number of samples grow, the choice of both kernel K and norm p becomes unimportant. The bandwidth H is still important.



A fast algorithm

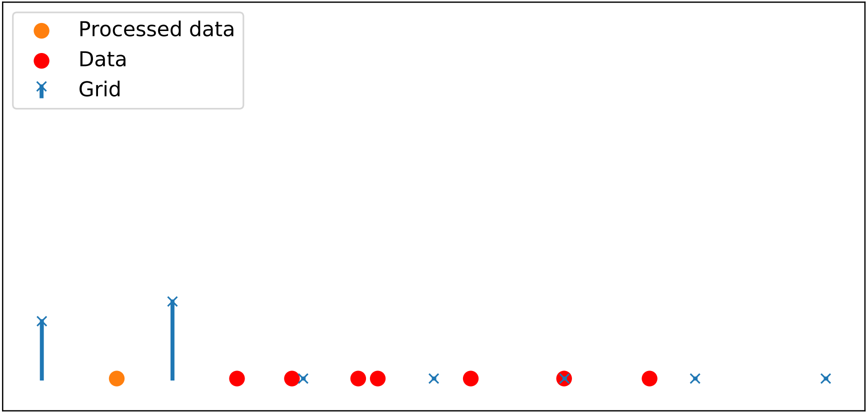
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



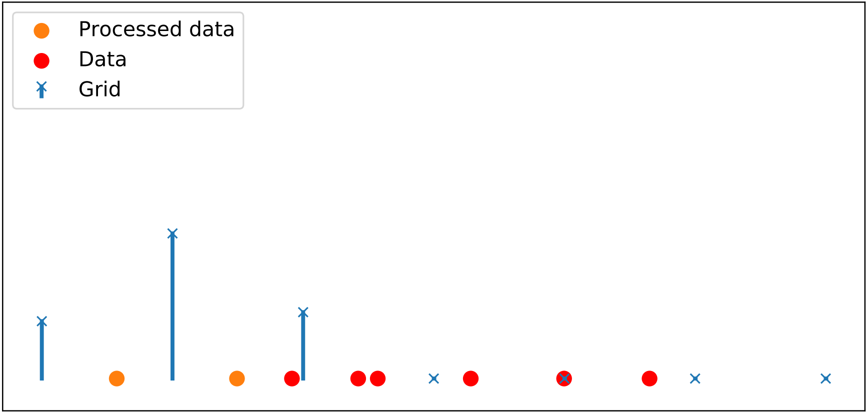
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



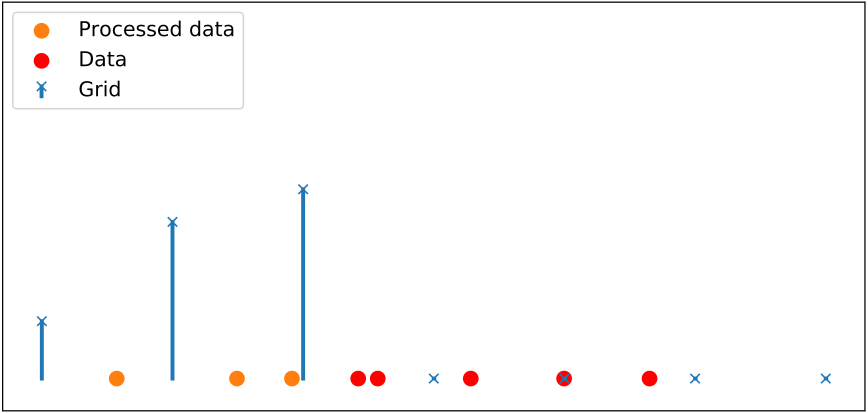
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



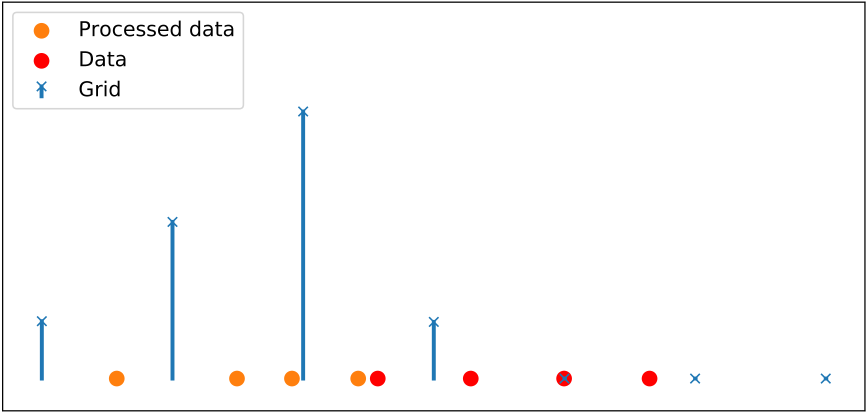
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



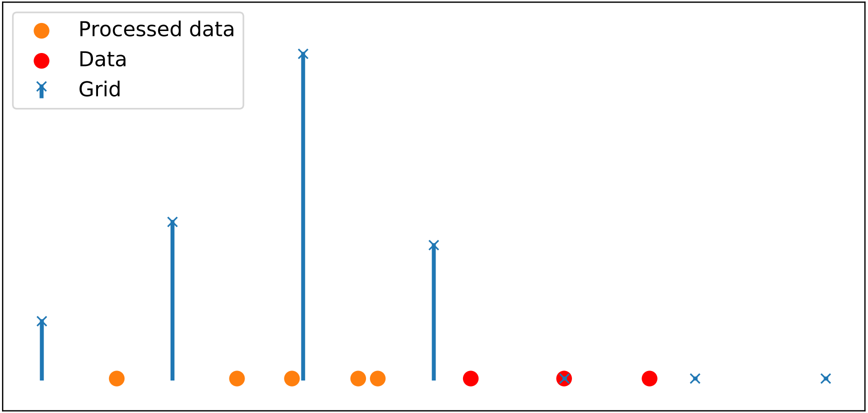
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



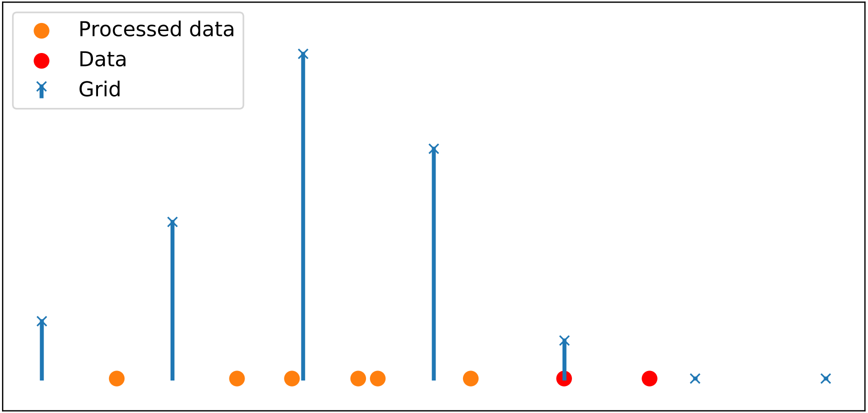
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



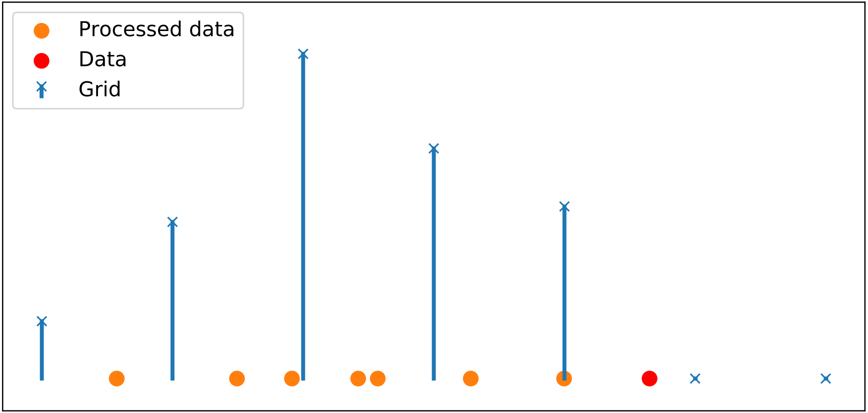
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



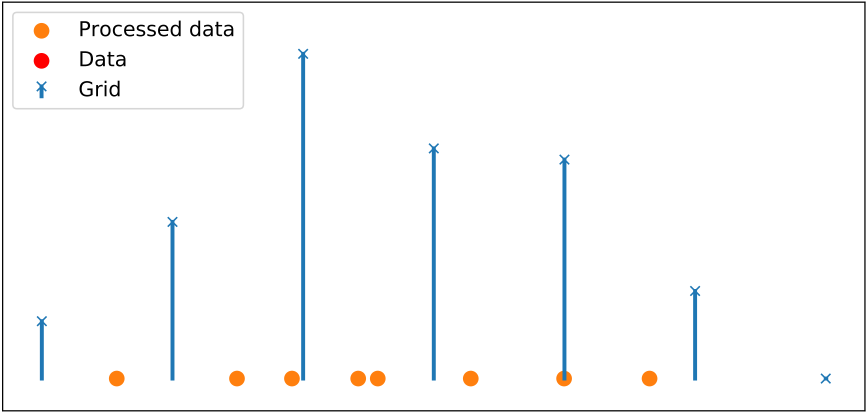
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



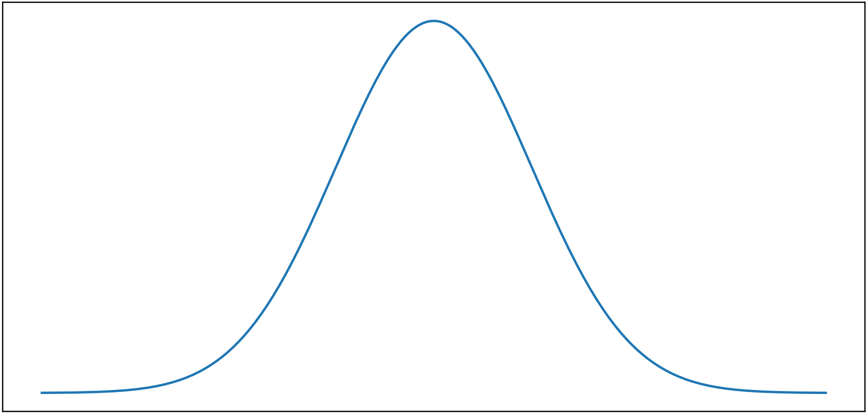
Linear binning

Go through N data points and assign weights to n equidistant grid points. The algorithm runs in $\mathcal{O}(N2^d)$ time in d dimensions.



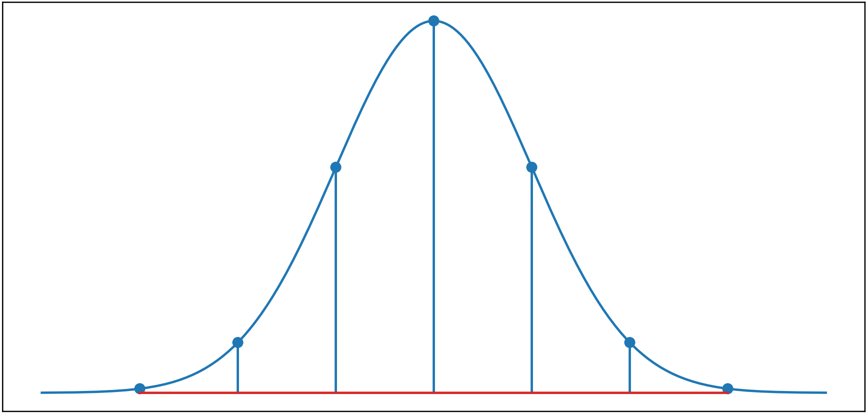
Sample the kernel

Sample the kernel function K at equidistant points. The n binned data points and the kernel are then convolved, this runs in $\mathcal{O}(n \log n)$ time, for a total time of $\mathcal{O}(N2^d + n \log n)$.



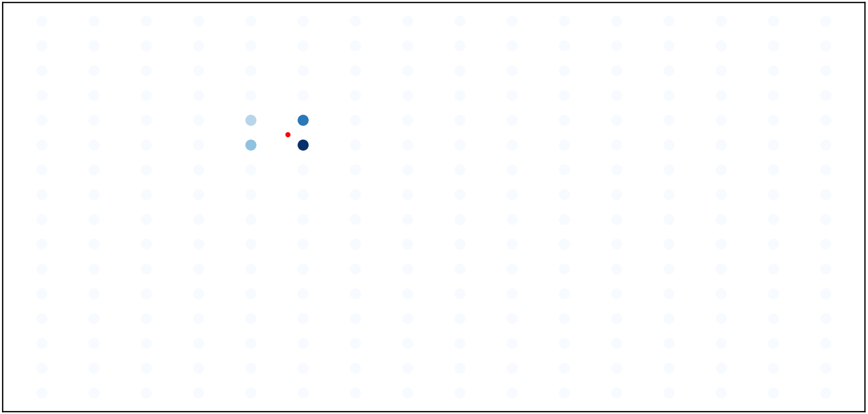
Sample the kernel

Sample the kernel function K at equidistant points. The n binned data points and the kernel are then convolved, this runs in $\mathcal{O}(n \log n)$ time, for a total time of $\mathcal{O}(N 2^d + n \log n)$.



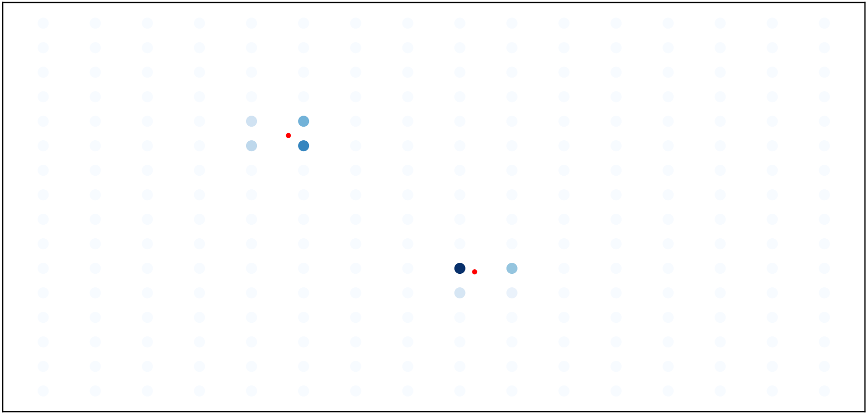
Linear binning in higher dimensions

The extension to d dimensions is relatively straightforward.



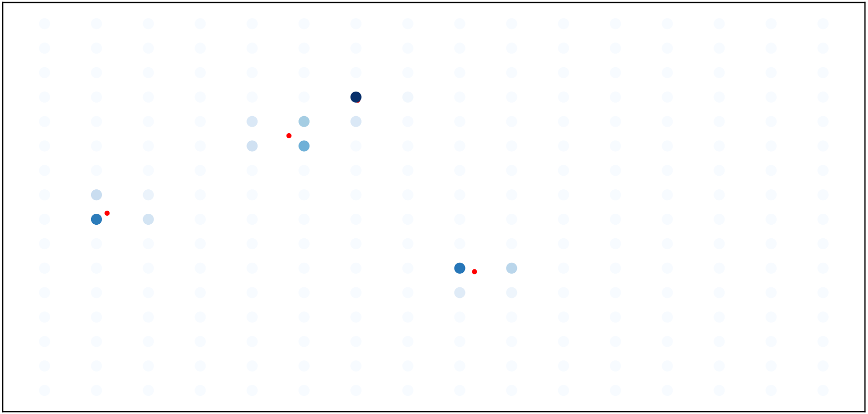
Linear binning in higher dimensions

The extension to d dimensions is relatively straightforward.



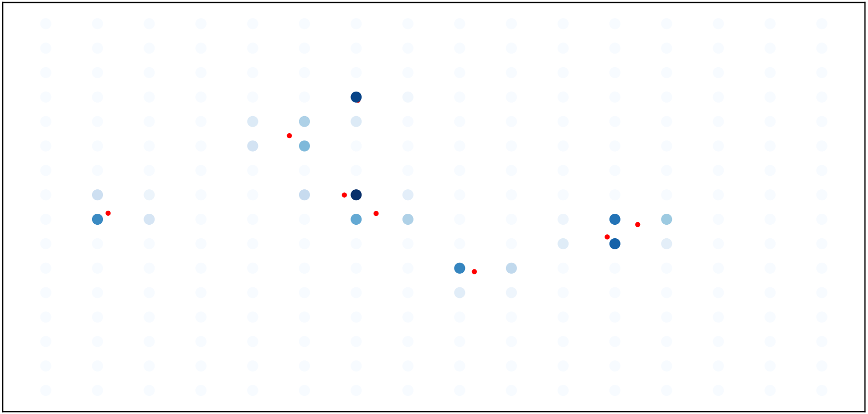
Linear binning in higher dimensions

The extension to d dimensions is relatively straightforward.



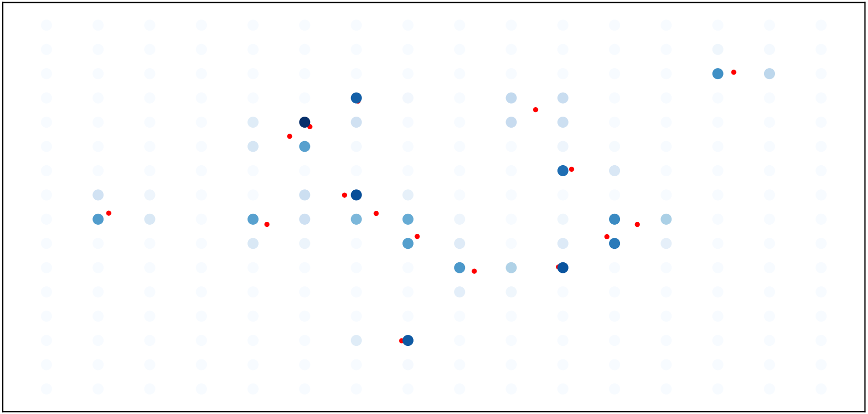
Linear binning in higher dimensions

The extension to d dimensions is relatively straightforward.



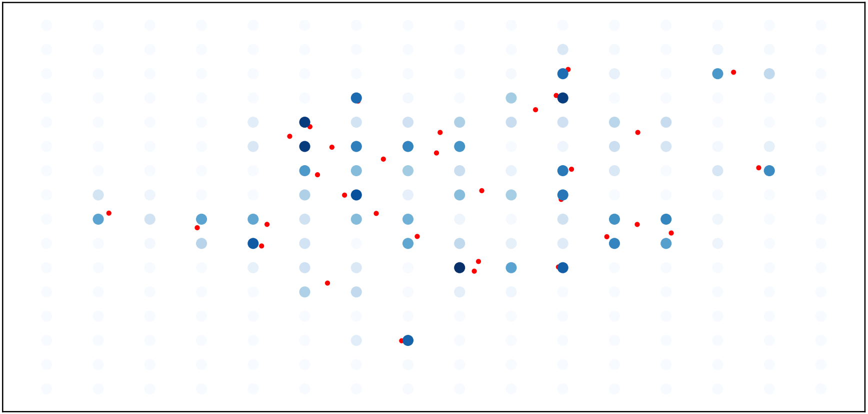
Linear binning in higher dimensions

The extension to d dimensions is relatively straightforward.



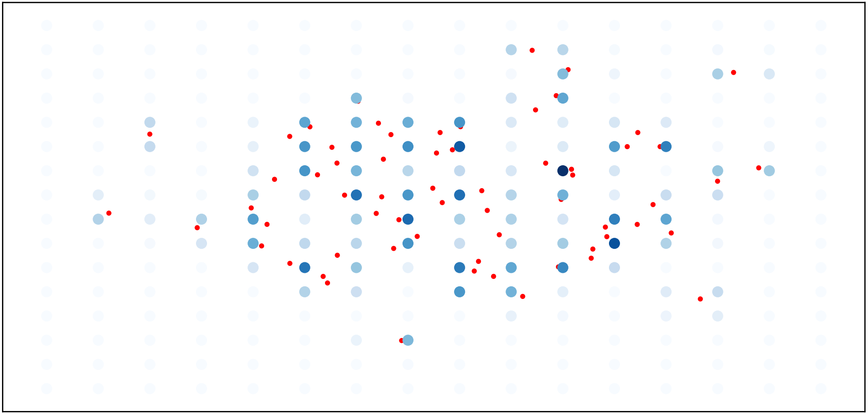
Linear binning in higher dimensions

The extension to d dimensions is relatively straightforward.



Linear binning in higher dimensions

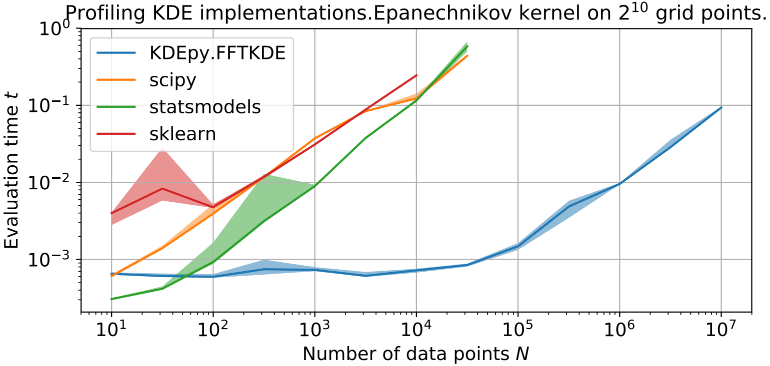
The extension to d dimensions is relatively straightforward.



KDEpy

If you're interested in KDE in Python, I've written a library.

- GitHub: `https://github.com/tommyod/KDEpy`



References

References for further reading.

- Silverman, B. W. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
- Wand, M. P., and M. C. Jones. *Kernel Smoothing.* Chapman and Hall, 1995.
- Jake VanderPlas. *Kernel Density Estimation in Python.* 2013 `https://jakevdp.github.io/blog/2013/12/01/kernel-density-estimation/`