COMP 432 Machine Learning

Kernel Density Estimation

Computer Science & Software Engineering
Concordia University, Fall 2021

UNIVERSITÉ
Concordia
UNIVERSITY

Parametric vs non-parametric

Parametric models have a fixed number of adaptable parameters, independent of the amount of data.

- Logistic regression $\mathbf{w} \in \mathbb{R}^D$
- *K*-component mixture of Guassians $\{\boldsymbol{\pi}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$

Non-parametric models have a variable number of parameters, adapting to the amount of data.

- Kernel density estimators (this lecture)
- Support vector machines (next lecture)
- Nearest neighbour classifiers (next lecture after that)

Aside: In practical terms, the distinction is somewhat fuzzy. Example: if a model selection procedure tends to select larger parametric models given more data, are we ‘adapting’ the number of parameters to the data and therefore “less parametric”? 🤷‍♂️ Histograms are considered “non-parametric” because they make fewer assumptions, despite having a fixed number of parameters (bin frequencies).

Kernel Density Estimation

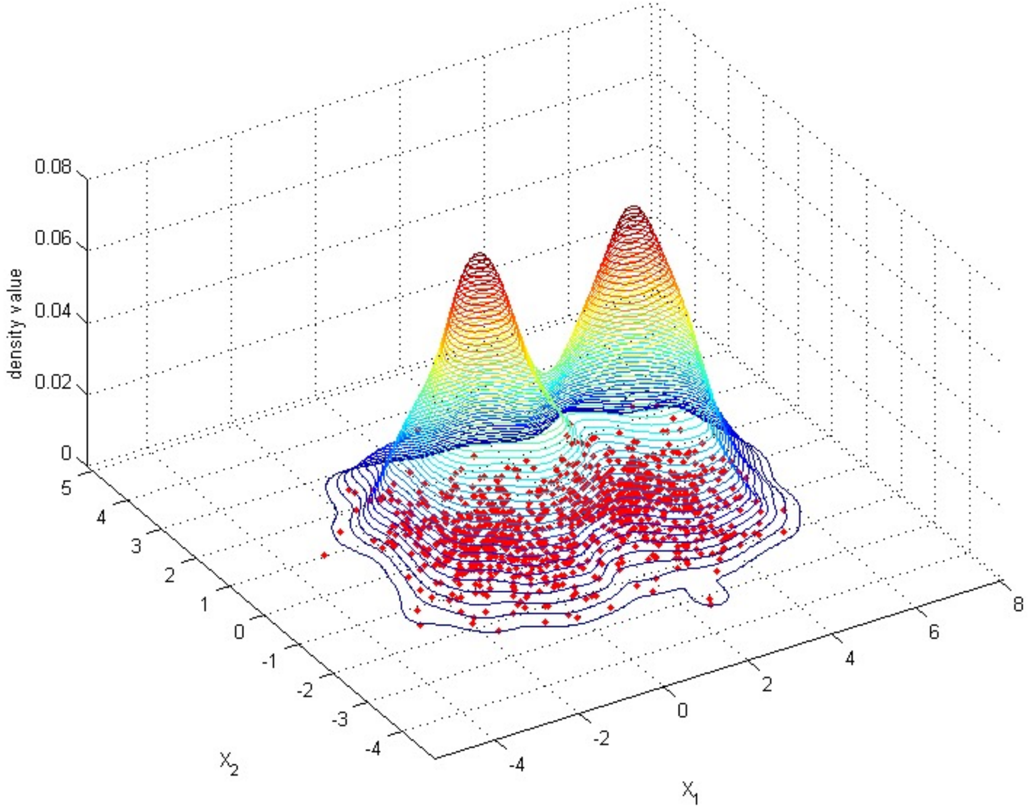
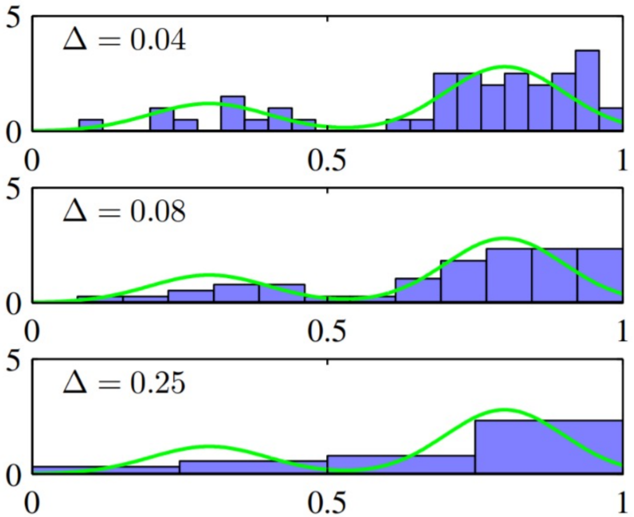


Image source: Wikipedia

Density Estimation

- **Goal:** Given samples $\mathcal{D} = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ we want to estimate $p(\mathbf{x})$ at a new point $\mathbf{x}$.
  - for scoring examples, or for novelty/anomaly detection...
  - for classification by comparing $p(\mathbf{x})$ to some $p(\mathbf{y})$
- Example: Fit a GMM to $\mathcal{D}$ and evaluate $p(\mathbf{x} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Example: Fit a histogram to $\mathcal{D}$ and evaluate density of bin that $\mathbf{x}$ falls into.



too many bins
(overfit)

OK number of bins

not enough bins
(underfit)

An illustration of the histogram approach to density estimation, in which a data set of 50 data points is generated from the distribution shown by the green curve. Histogram density estimates are shown for various values of bin width Δ

Image credit: Christpher M. Bishop

Kernel Density Estimation

- **Idea:** Estimate $p(\mathbf{x})$ by "smoothing" the data $\mathcal{D}$ itself. Do this by convolving the $\mathbf{x}_i$ with some "kernel."
  - The model of $p(\mathbf{x})$ is expressed directly in terms of data $\mathbf{x}_i$
  - The complexity of the density estimate can scale with the amount of data (more data => more terms in $p(\mathbf{x})$)

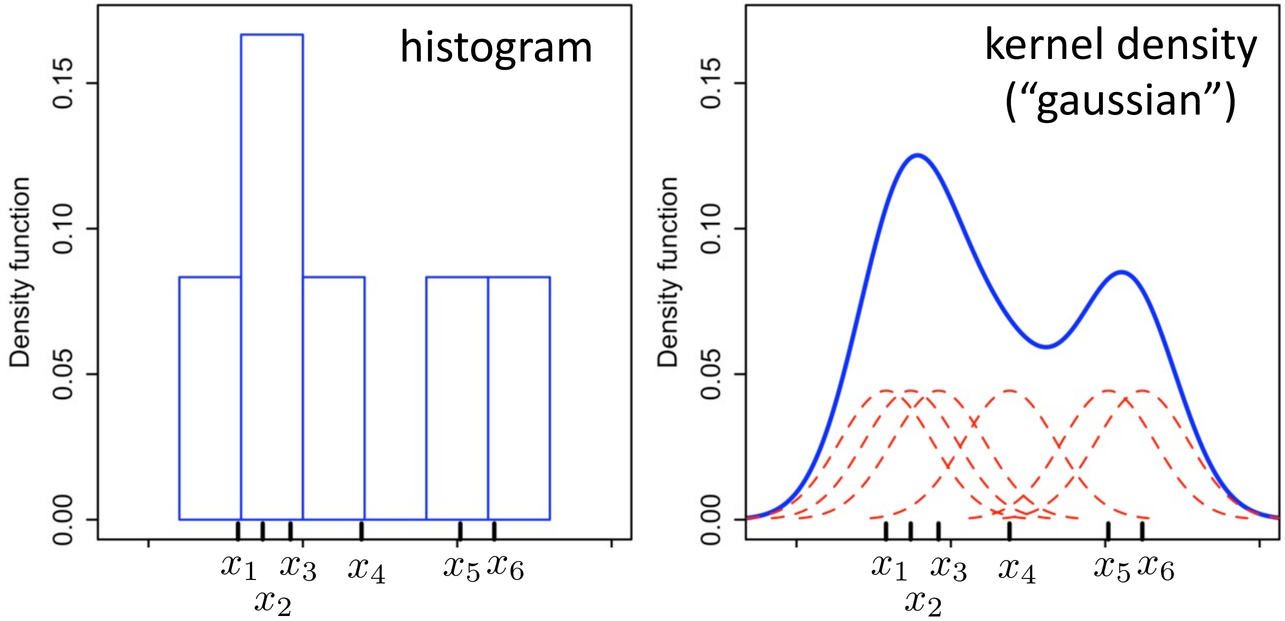
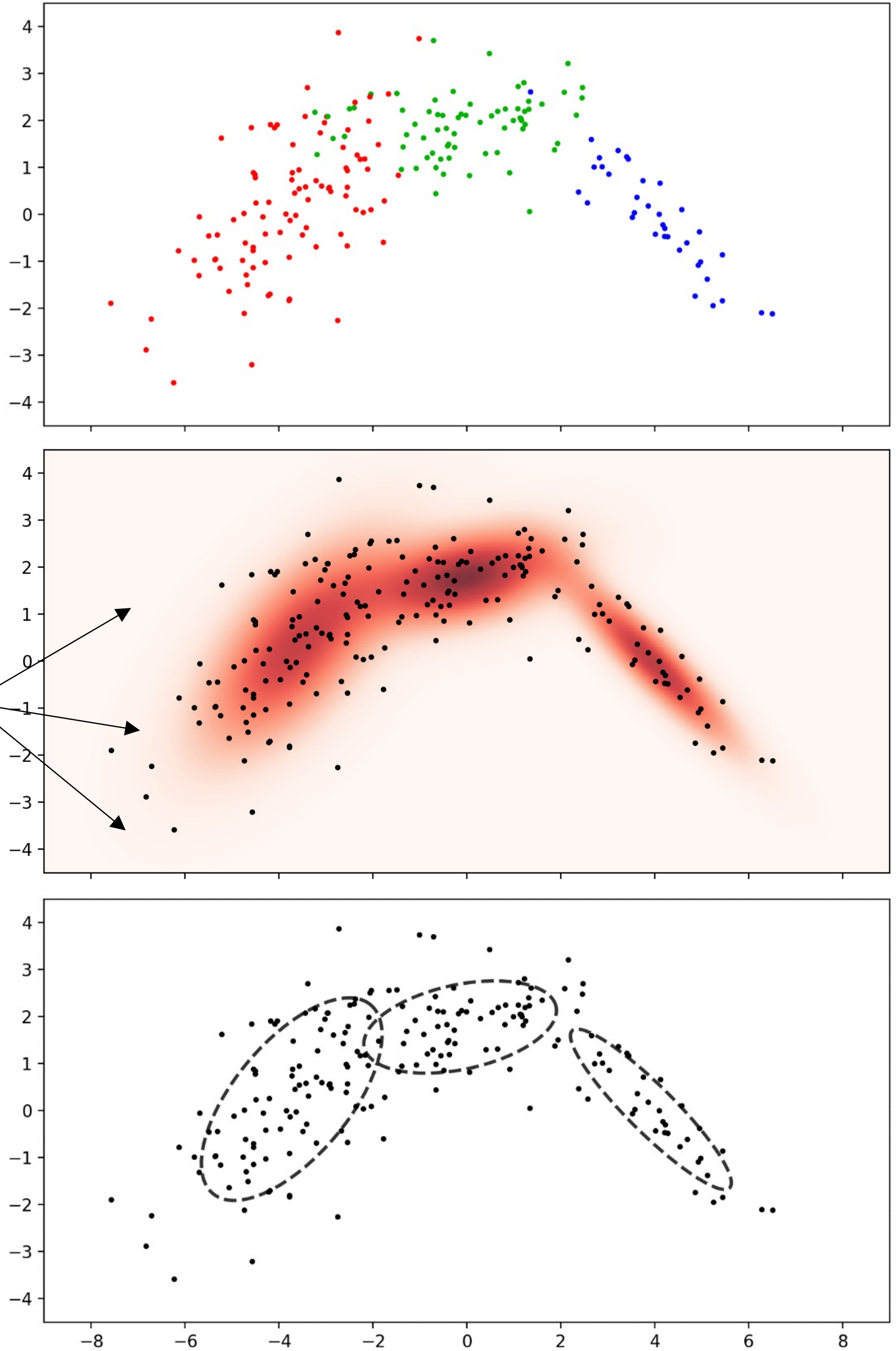


Image source: Wikipedia

GMM densities



Synthetic data from GMM

$p(\mathbf{x})$

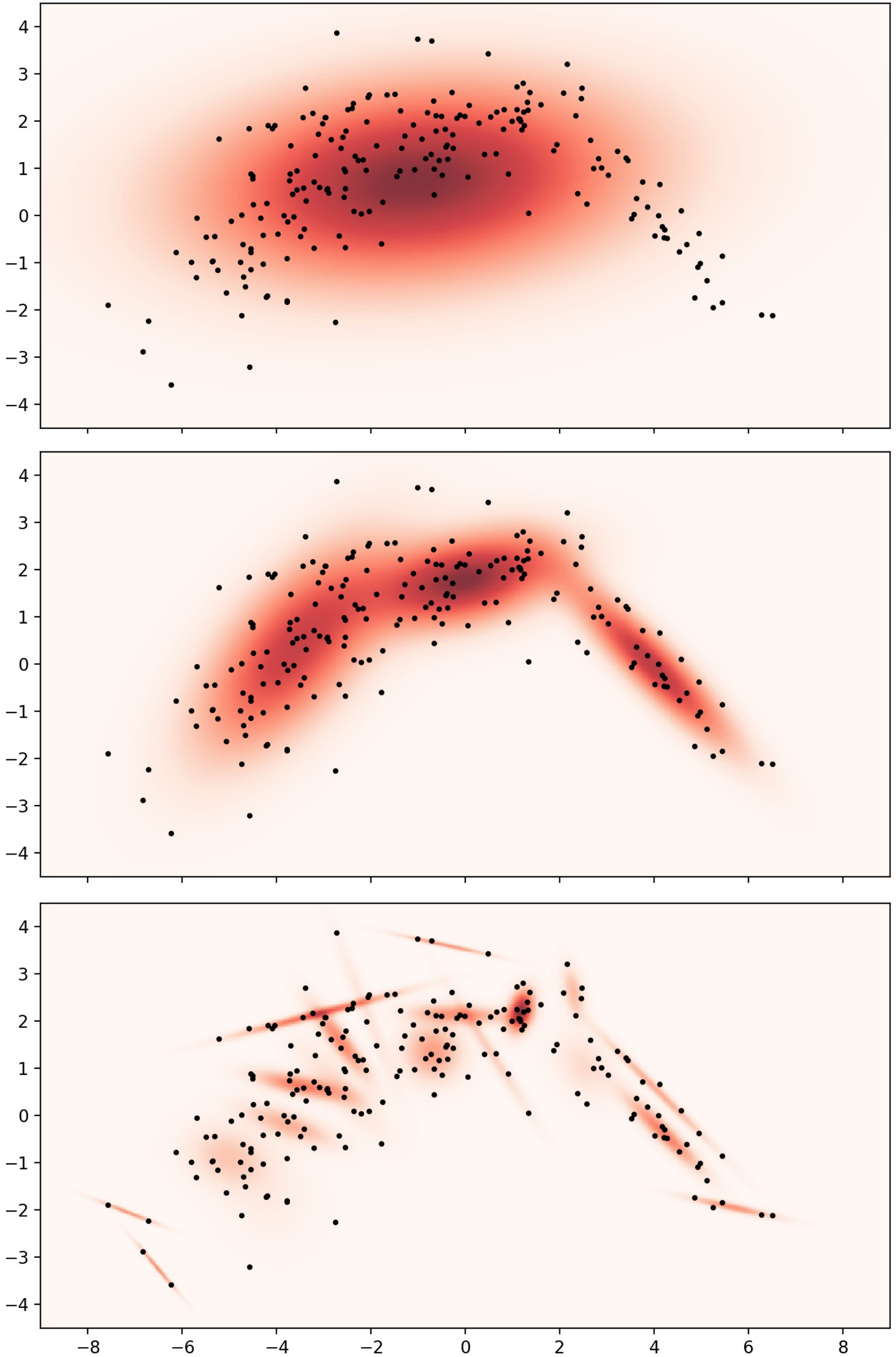
(really $p(\mathbf{x} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$)

Density from GMM fit to the synthetic data

$\boldsymbol{\pi}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$

Model parameters of the GMM that was fit

GMM densities (parametric)



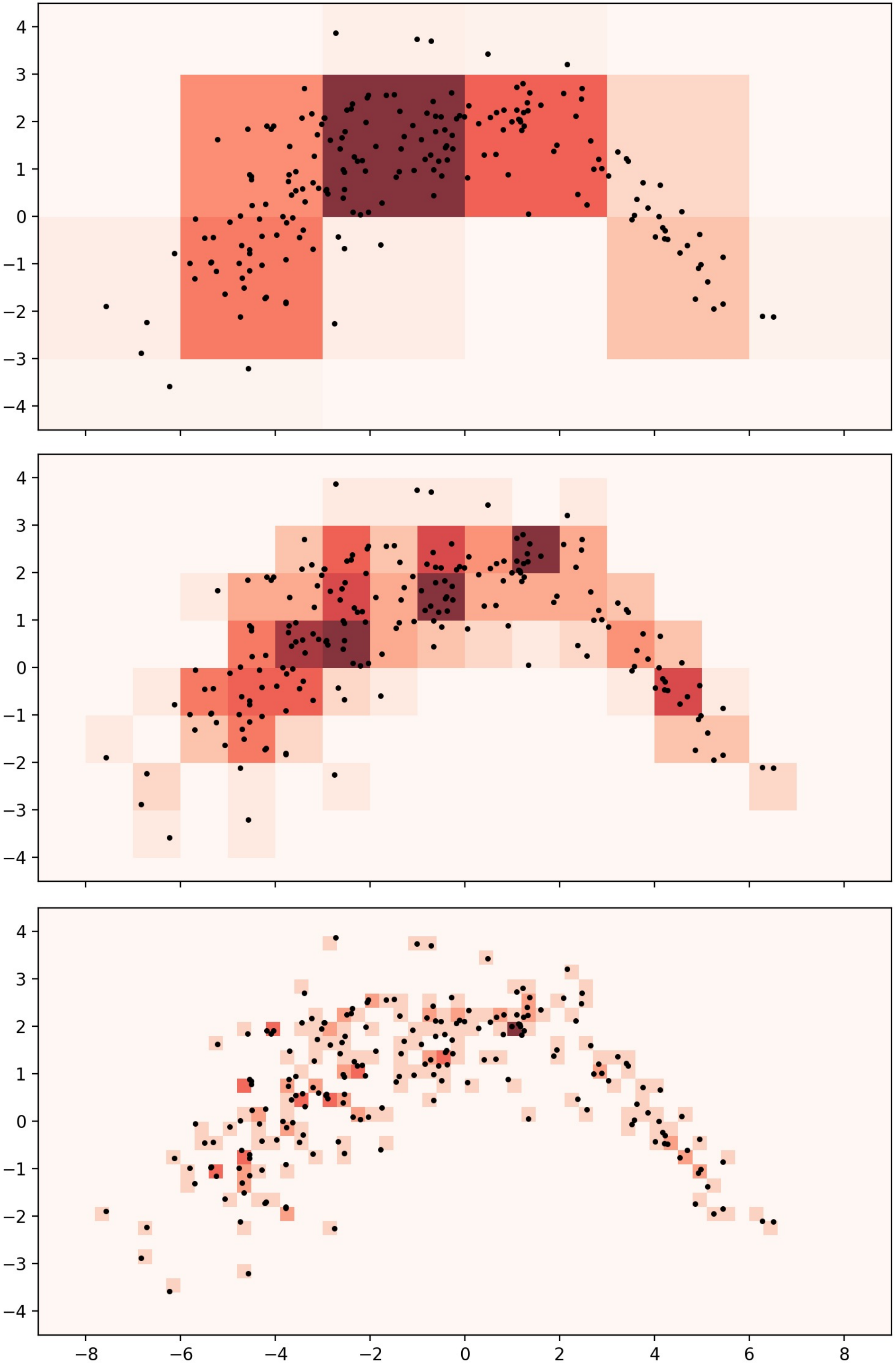
K=1 component

K=3 components

K=20 components

Histogram densities

(non-parametric, but not kernel density)

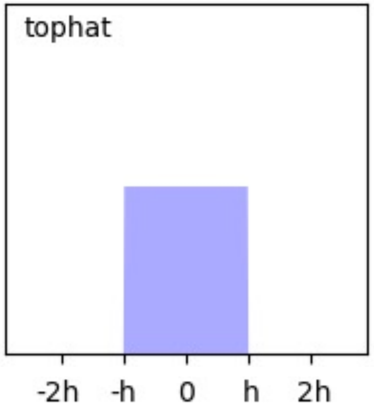


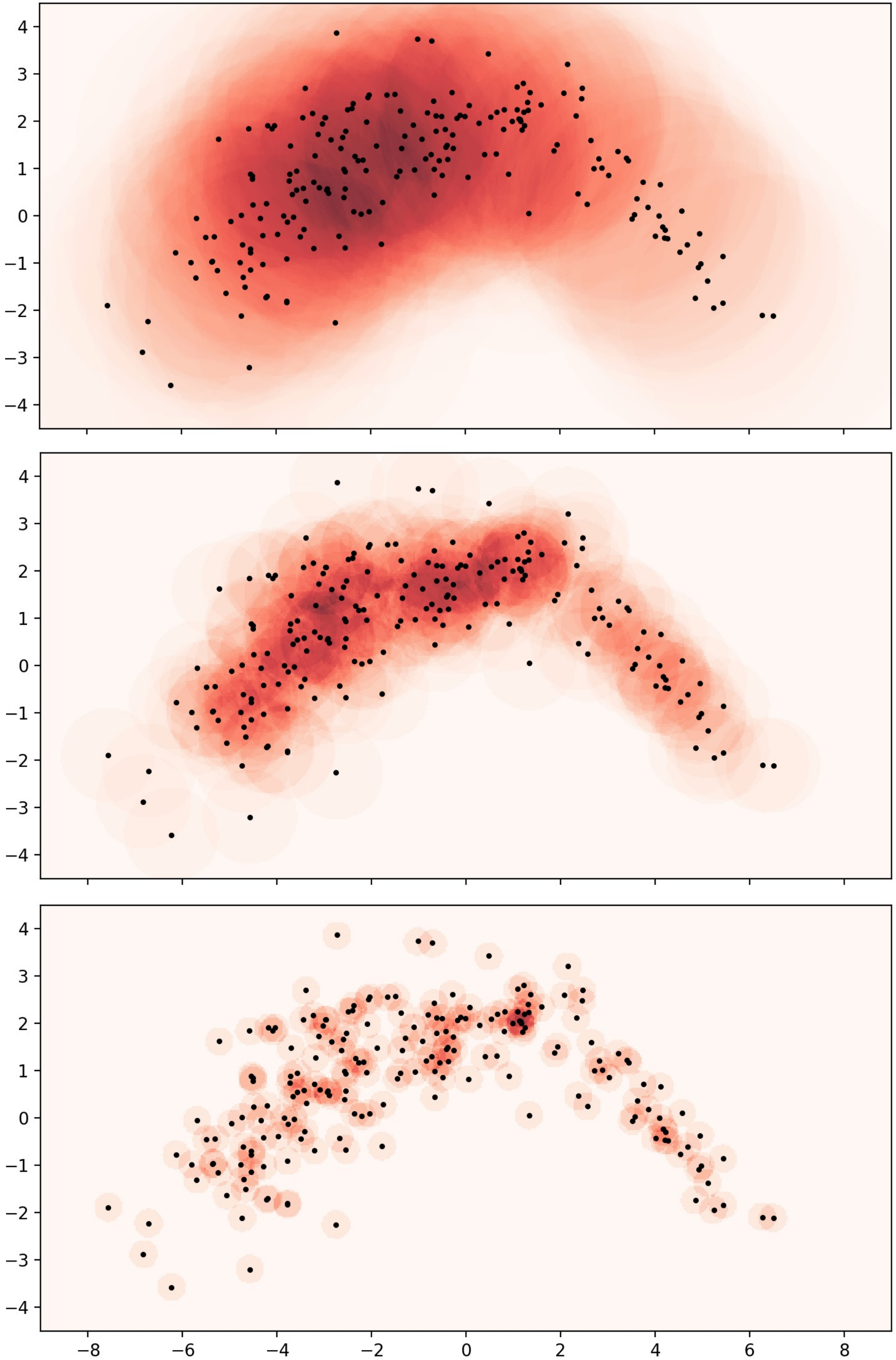
binwidth=3.0

binwidth=1.0

binwidth=0.3

Kernel densities (“tophat”)



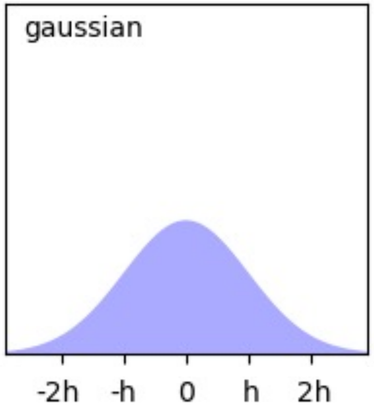


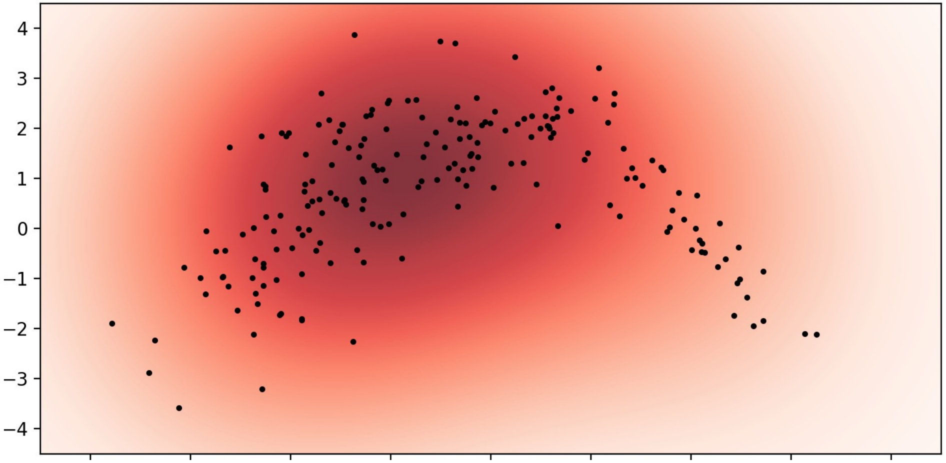
h=3.0 bandwidth

h=1.0 bandwidth

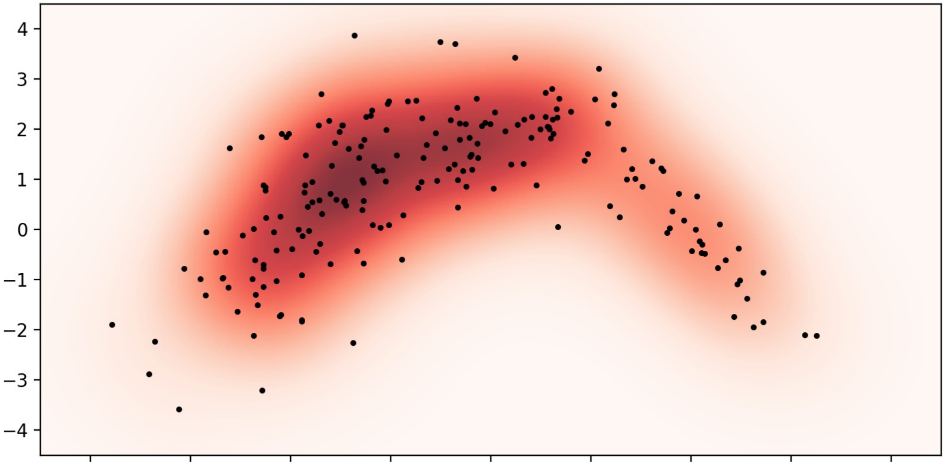
h=0.3 bandwidth

Kernel densities (“gaussian”)

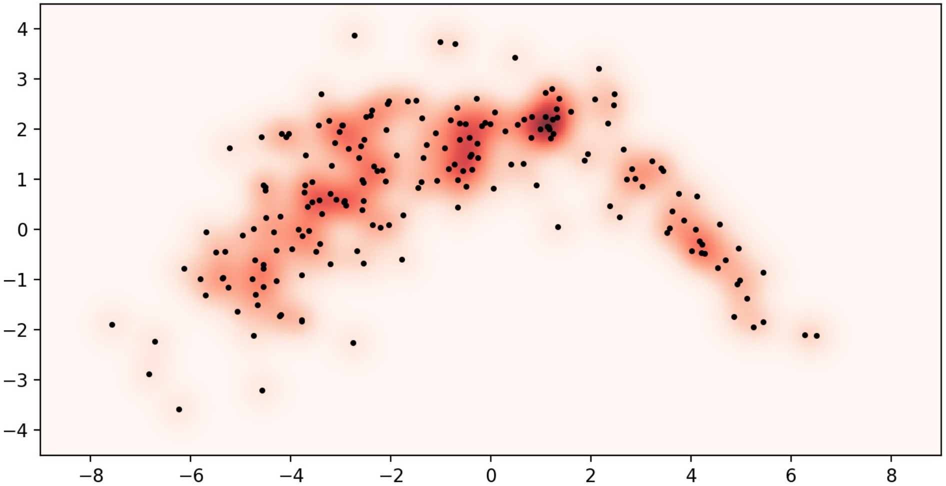




h=3.0 bandwidth



h=1.0 bandwidth

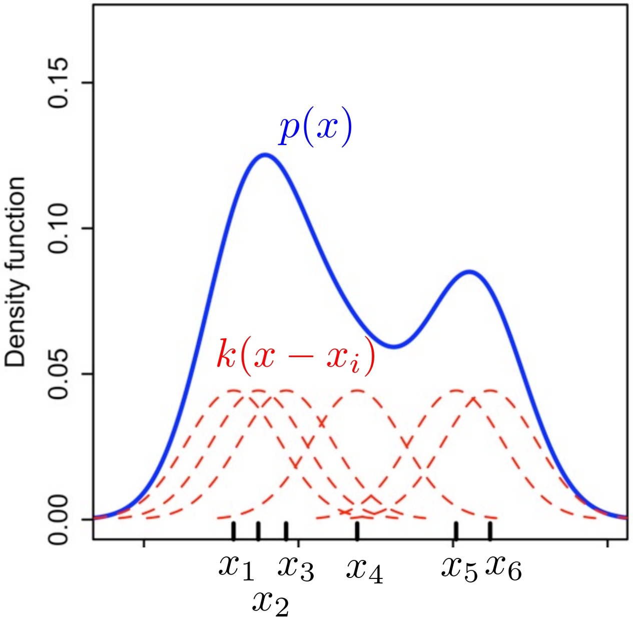


h=0.3 bandwidth

Kernel density estimator

A **kernel density estimator** (or **Parzen estimator**) is computed by adding an instance of the kernel function *centered at each data point* $\mathbf{x}_i$ and then normalizing.

$$p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x} - \mathbf{x}_i)$$



A valid "kernel function" must satisfy:

$$k(\mathbf{x}) \geq 0 \quad \text{(non-negative)}$$

$$\int_{\mathbb{R}^D} k(\mathbf{x})\mathrm{d}\mathbf{x} = 1 \quad \text{(normalized)}$$

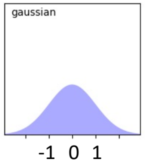
With Gaussian kernel, it's just an *N*-component GMM with equal weights and variances! "non-parametric"

Kernels for density estimation

Gaussian kernel

$$k(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \exp\left(-\frac{1}{2}\left\|\mathbf{x}\right\|^2\right)$$

i.e. $\mathcal{N}(\mathbf{x} \mid \mathbf{0}, \mathbf{I})$



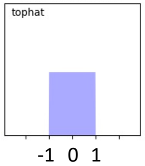
Tophat kernel

$$k(\mathbf{x}) = \begin{cases} \frac{\Gamma(\frac{D}{2}+1)}{\sqrt{\pi}^D} & \left\|\mathbf{x}\right\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

where Γ is the gamma function.

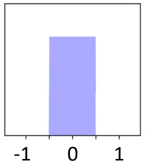
e.g. when $D = 1$

$$\frac{\Gamma(\frac{3}{2})}{\sqrt{\pi}} = \frac{1}{2}$$



Parzen window

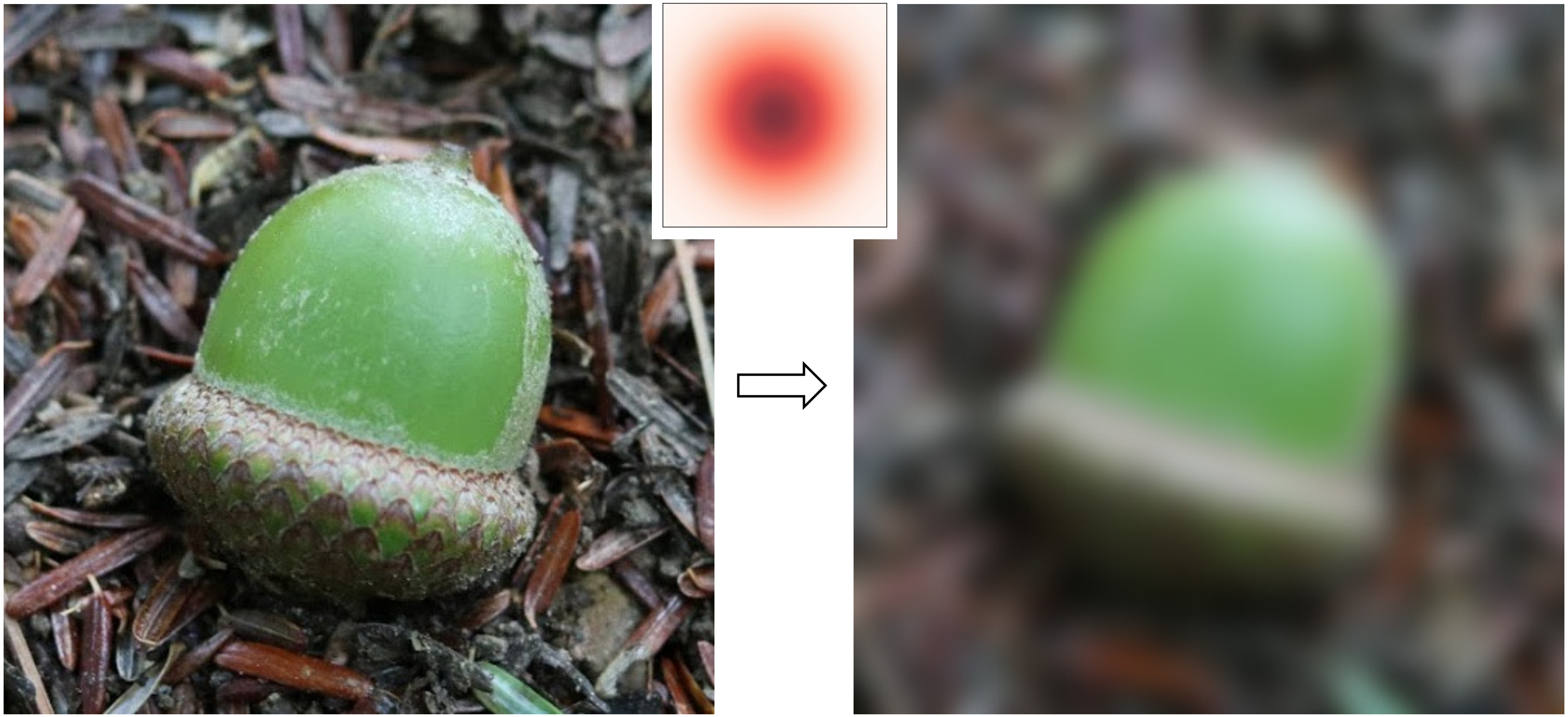
$$k(\mathbf{x}) = \begin{cases} 1 & |x_i| \leq \frac{1}{2} \text{ for } i = 1, \ldots, D \\ 0 & \text{otherwise} \end{cases}$$



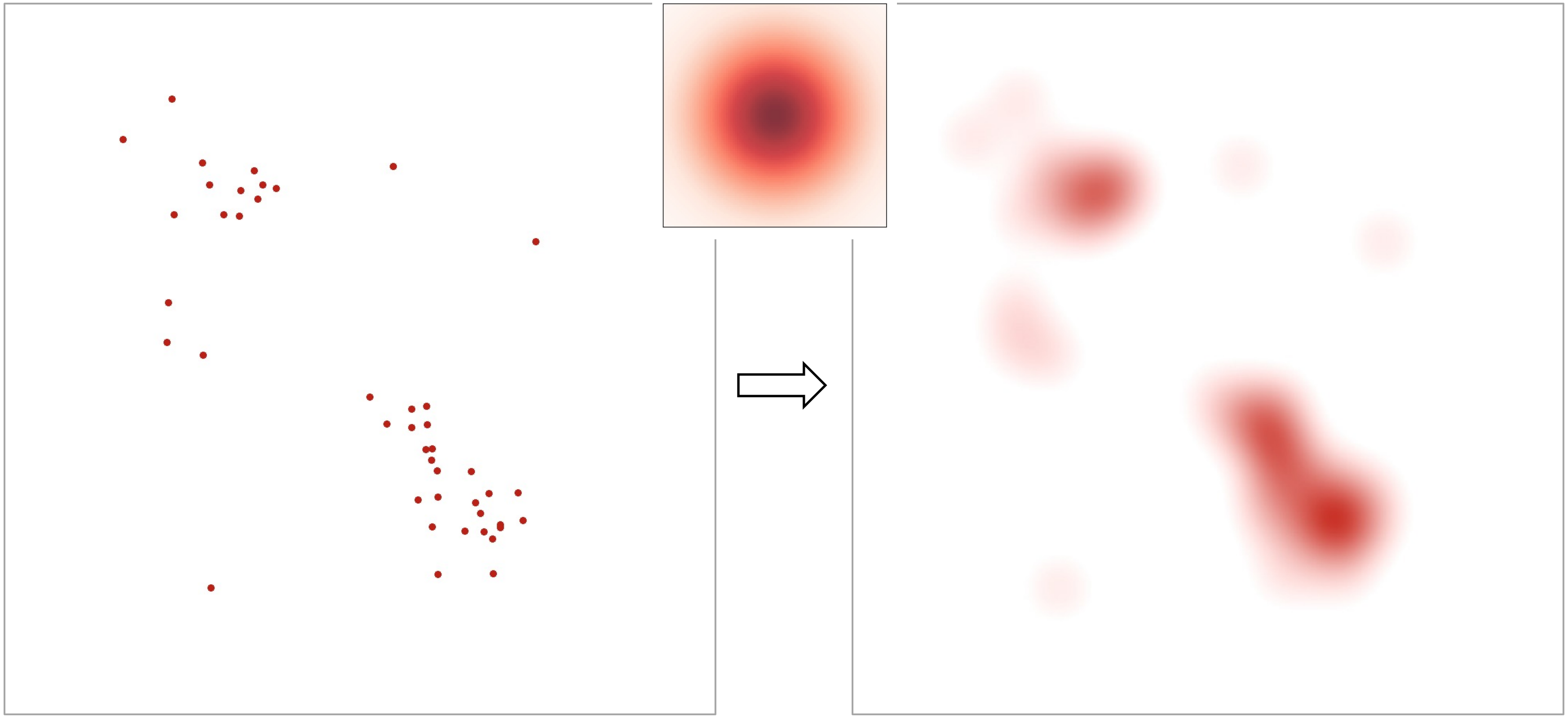
We can "add bandwidth *h*" to any kernel $k(\mathbf{x})$ by taking $k_h(\mathbf{x}) = \frac{1}{h^D} k\left(\frac{\mathbf{x}}{h}\right)$

Smoothing as convolution

$f(x, y)$ $k(x, y)$ $(f * k)(x, y)$



Smoothing as convolution



$f(x,y)$ $k(x,y)$ $(f * k)(x,y)$

"*f* is zero everywhere except at the data points"

Kernel density as convolution (conceptual!)

The **empirical density** is zero everywhere except the at the empirically observed data:

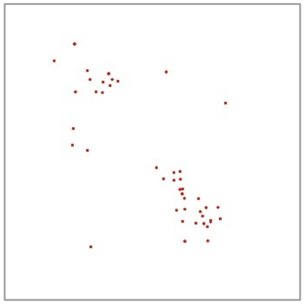
$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{x}_i)$$

The "Dirac delta function" satisfies:

$$\delta(\mathbf{x}) = \begin{cases} +\infty & \mathbf{x}=\mathbf{0} \\ 0 & \mathbf{x}\neq\mathbf{0} \end{cases}$$

such that: $\int_{\mathbb{R}^D}\delta(\mathbf{x})\mathrm{d}\mathbf{x} = 1$

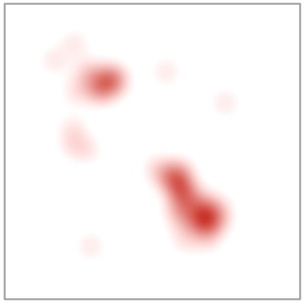
Data points in feature space



$f(\mathbf{x})$

$\Downarrow$

$p(\mathbf{x})$



Density estimate in feature space

The **kernel density** is the empirical density convolved with a smoothing kernel *k:*

$$p(\mathbf{x}) = (f * k)(\mathbf{x}) \qquad \text{“convolution of } f \text{ by kernel } k\text{”}$$

$$= \frac{1}{N}\sum_{i=1}^{N}k(\mathbf{x}-\mathbf{x}_i)$$

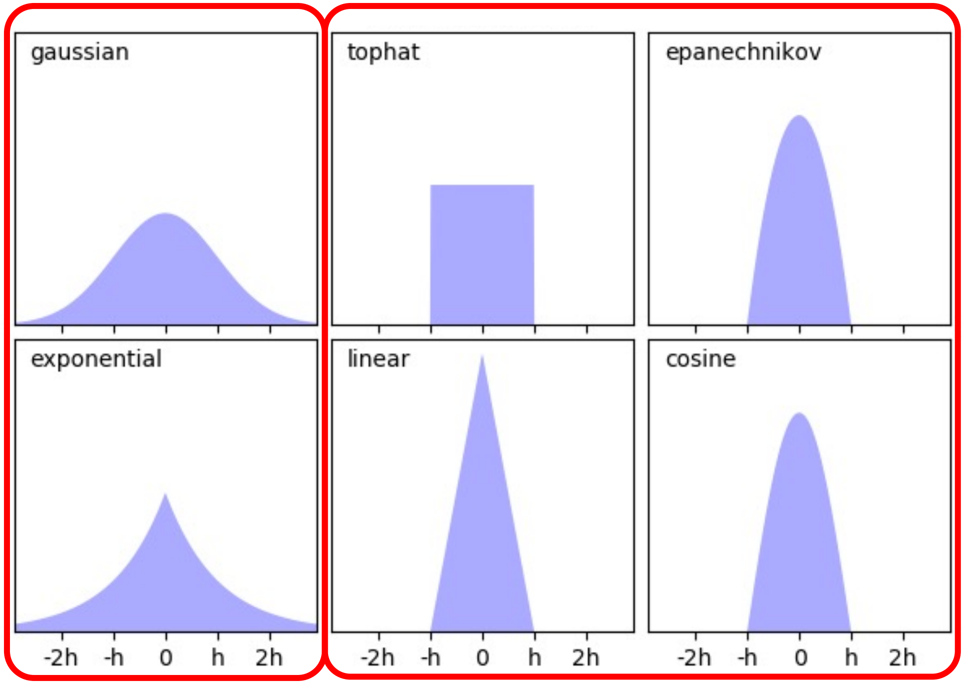
Again, $p(\mathbf{x})$ is short for $p(\mathbf{x}\mid\mathcal{D},k)$ given data $\mathcal{D}$ and choice of kernel k.

Kernel Parameters

🤷‍♂️

Two free parameters in this “non-parametric” model

1. Kernel function:

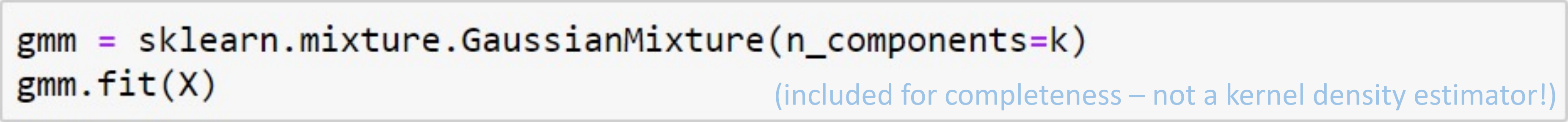


These work well and have nice probabilistic interpretations, but may be slow to estimate predict new $p(\mathbf{x})$ because *all* training points contribute to density, even if very small contribution.

These are faster at predicting a new density $p(\mathbf{x})$ because the kernels have a limited range of support so only a small subset of training points will contribute to density.

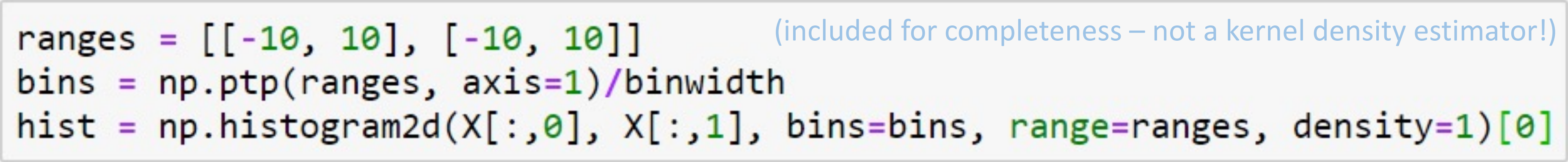
2. Kernel function bandwidth *h*

Scikit-learn and Numpy



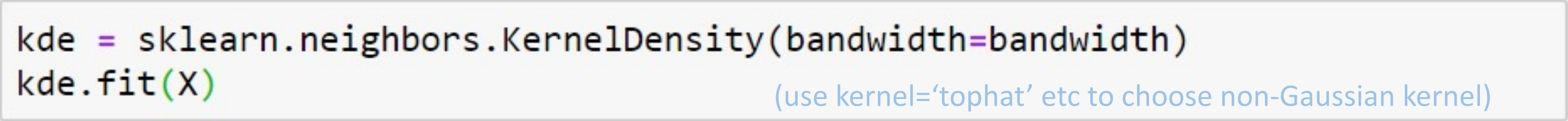
```
gmm = sklearn.mixture.GaussianMixture(n_components=k)
gmm.fit(X)
```

(included for completeness – not a kernel density estimator!)



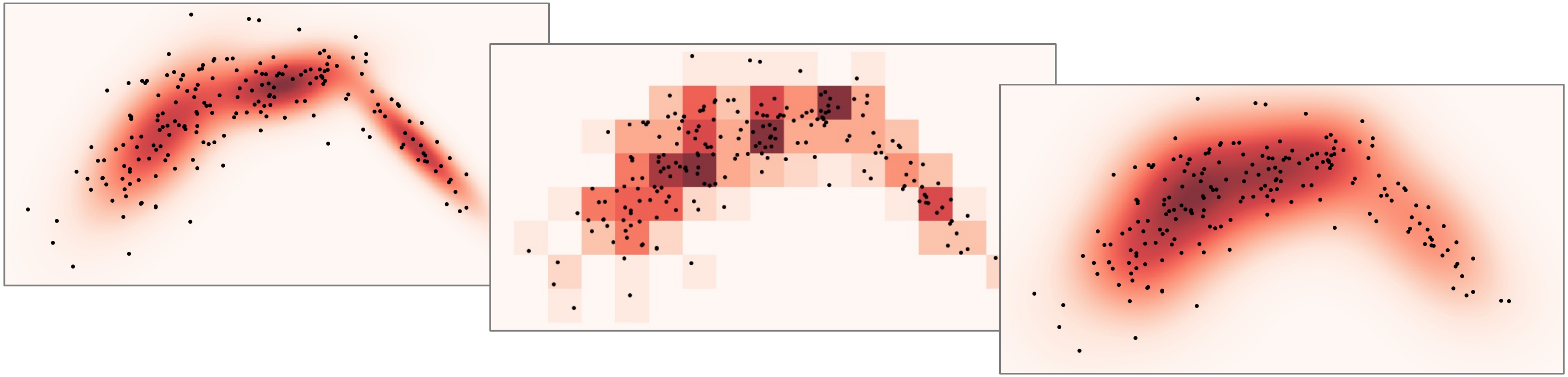
```
ranges = [[-10, 10], [-10, 10]]
bins = np.ptp(ranges, axis=1)/binwidth
hist = np.histogram2d(X[:,0], X[:,1], bins=bins, range=ranges, density=1)[0]
```

(included for completeness – not a kernel density estimator!)



```
kde = sklearn.neighbors.KernelDensity(bandwidth=bandwidth)
kde.fit(X)
```

(use kernel='tophat' etc to choose non-Gaussian kernel)



PRML Readings

§2.5.0 Nonparametric methods

§2.5.1 Kernel density estimators