18.02.2020

Statistical Methods in AI (CSE/ECE 471)

Lecture-12: Nonparametric Density Estimation

Ravi Kiran (ravi.kiran@iiit.ac.in)

Vineet Gandhi (v.gandhi@iiit.ac.in)

Center for Visual Information Technology (CVIT)

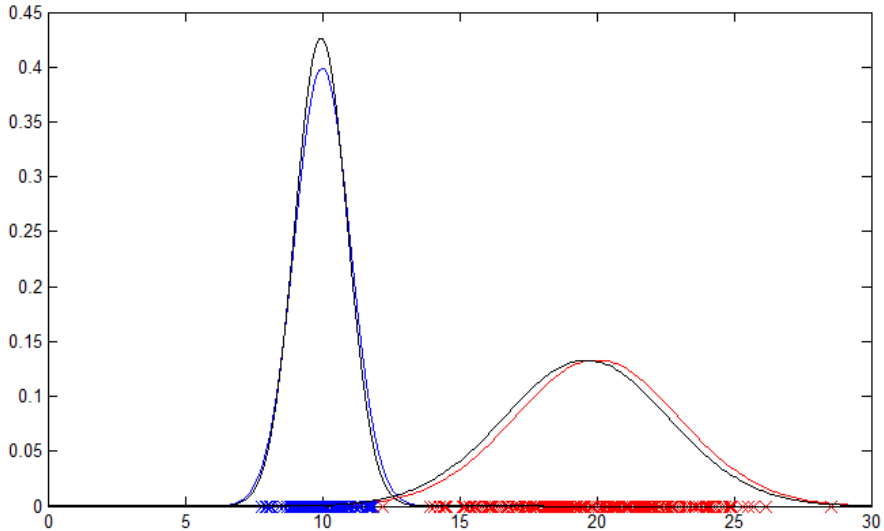
IIIT Hyderabad

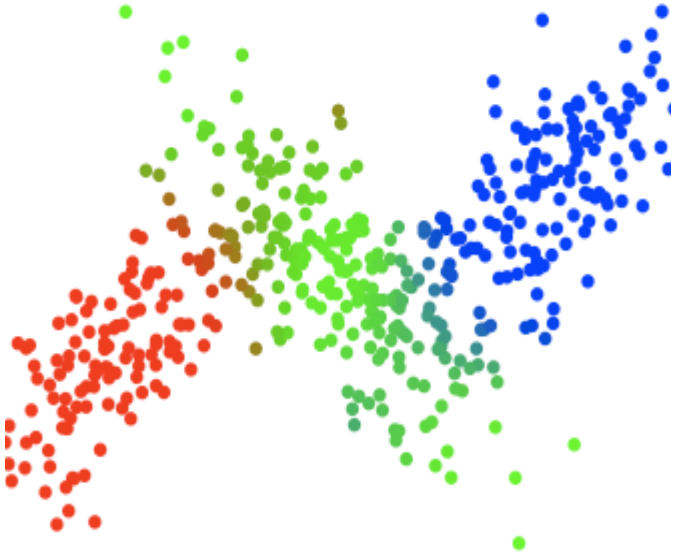
Unsupervised Learning → Density Estimation

Task: Given $X \in \mathcal{X}$, learn $f(X)$.

Gaussian Mixture Model

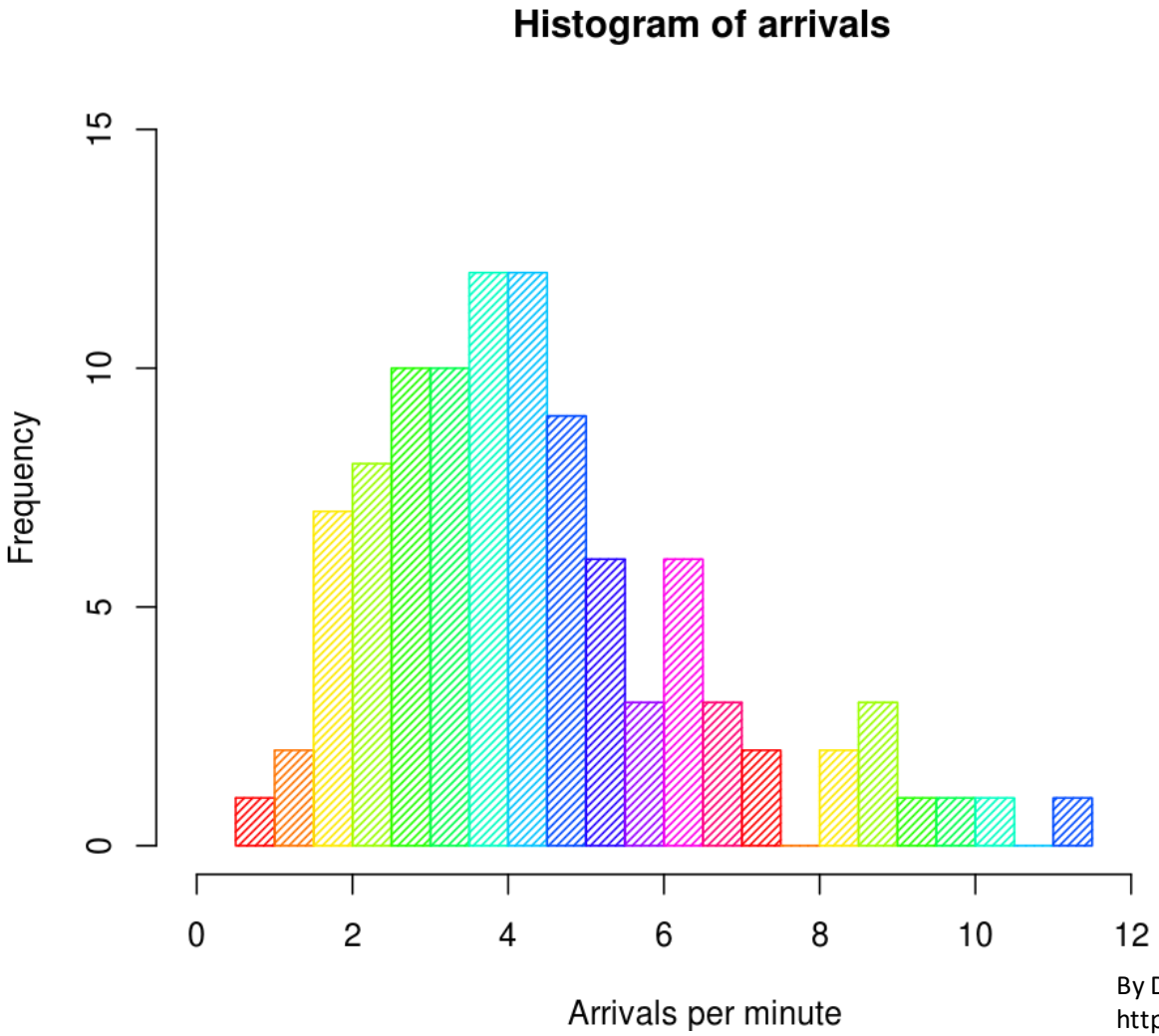
$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

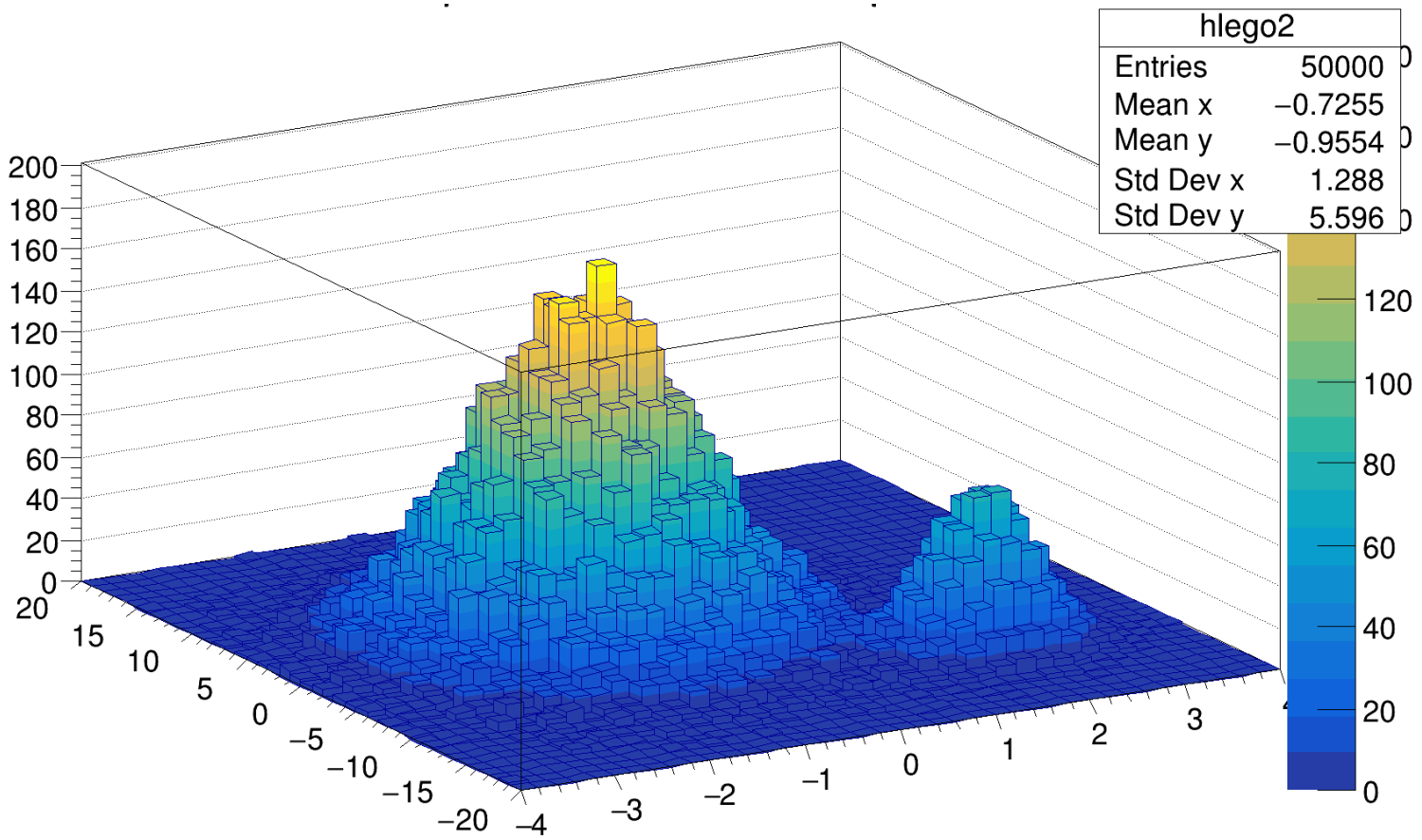




- What if component densities are not unimodal Gaussian ?
- Can we get rid of K ?

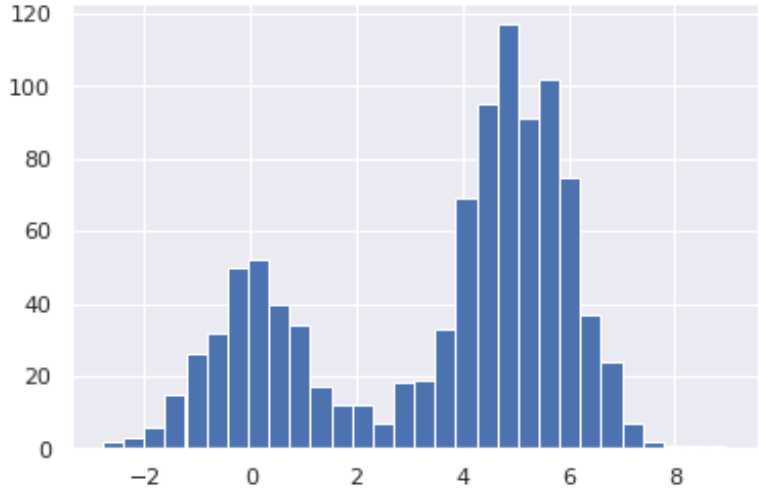
Non-parametric density estimation





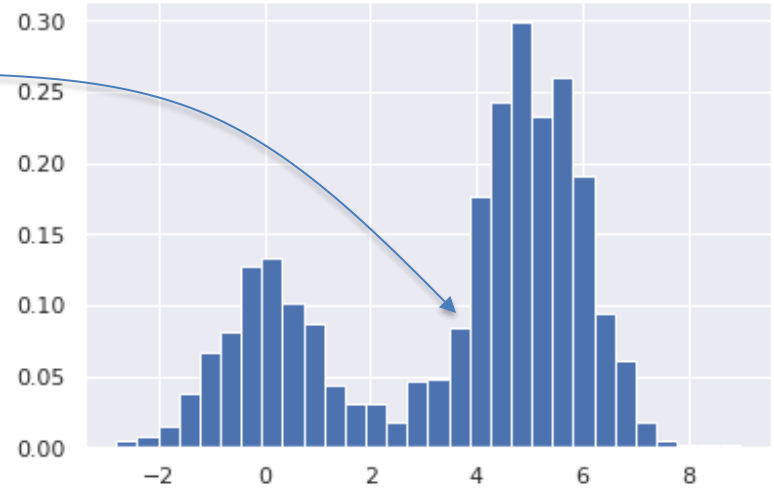
https://root.cern/doc/master/classTHistPainter.html

Normalizing a histogram → probability density

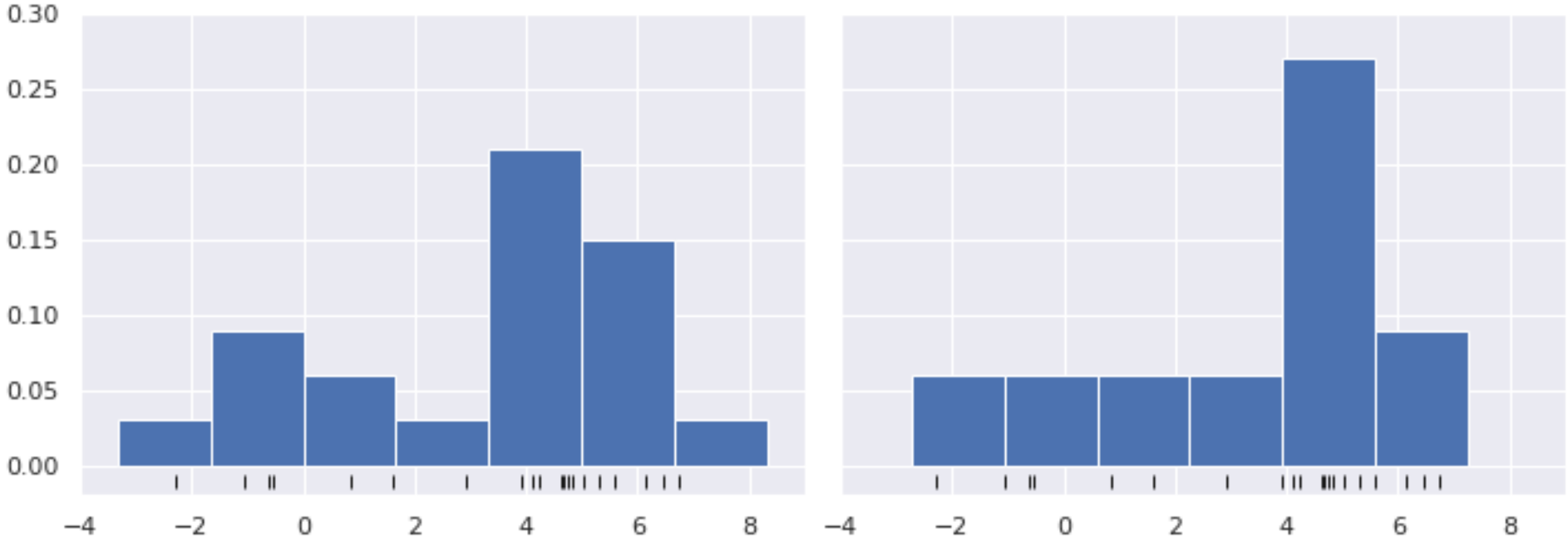


Probability that a random sample x will fall in this bin

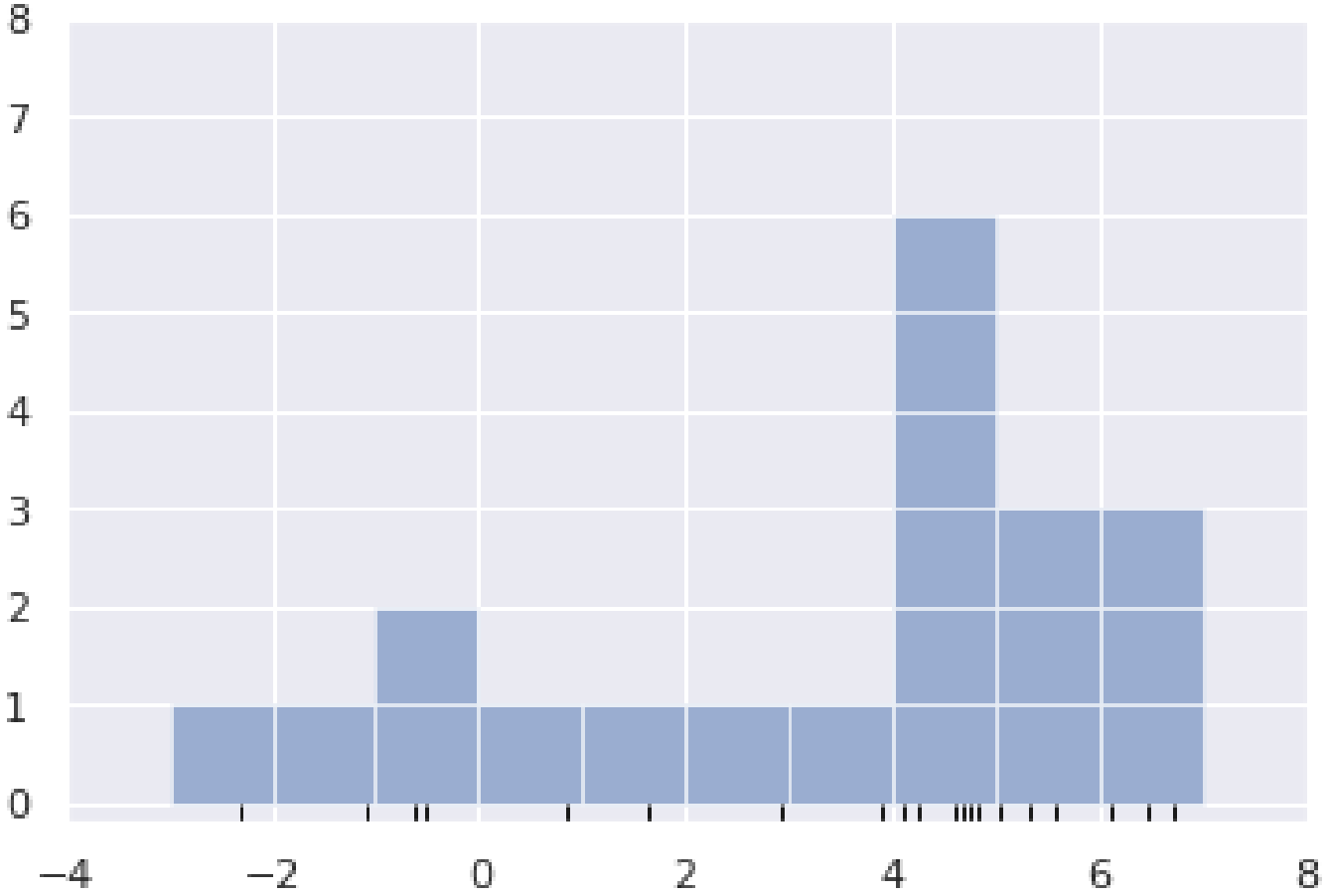
$$p_H(x) = \frac{1}{N}\frac{[\#\ of\ x^{(k}\ in\ same\ bin\ as\ x]}{[width\ of\ bin]}$$



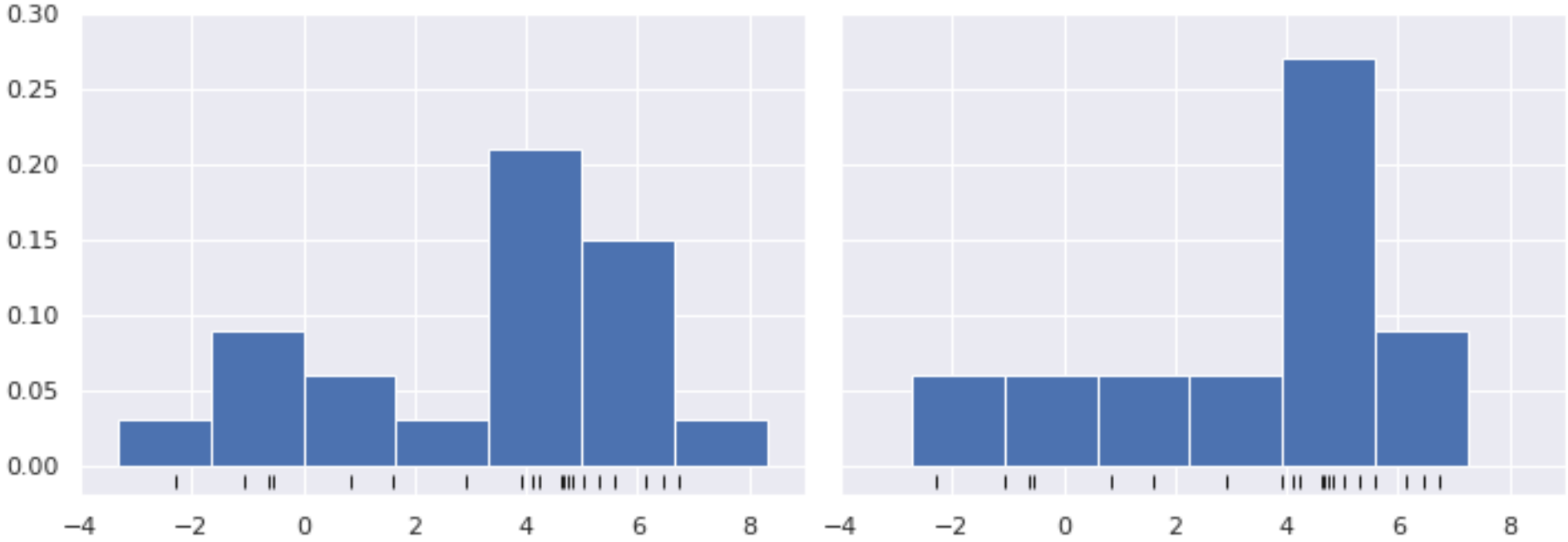
Choice of bin sizes and bin edge locations can lead to different looking histograms for same data



Histogram as a stack of unit sized blocks. Each data sample contributes towards a block.



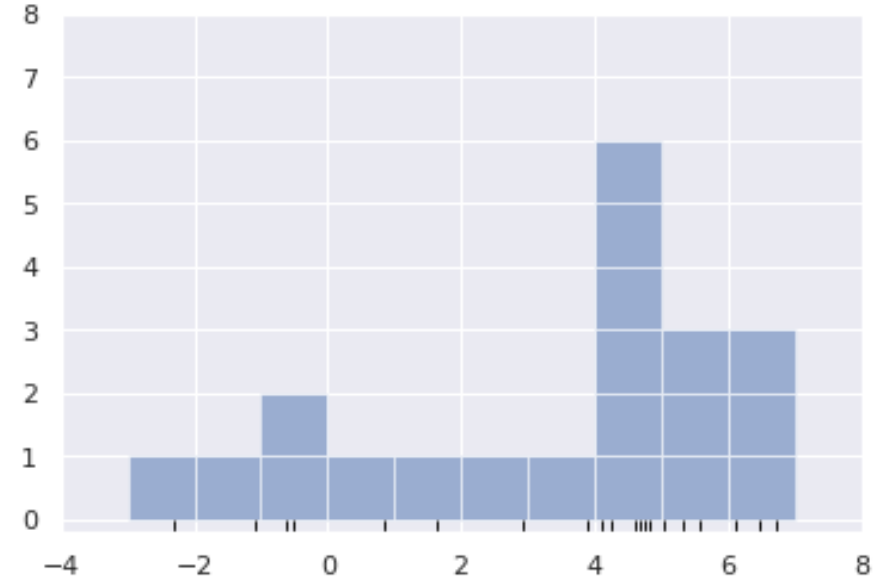
Choice of bin sizes and bin edge locations can lead to different looking histograms for same data

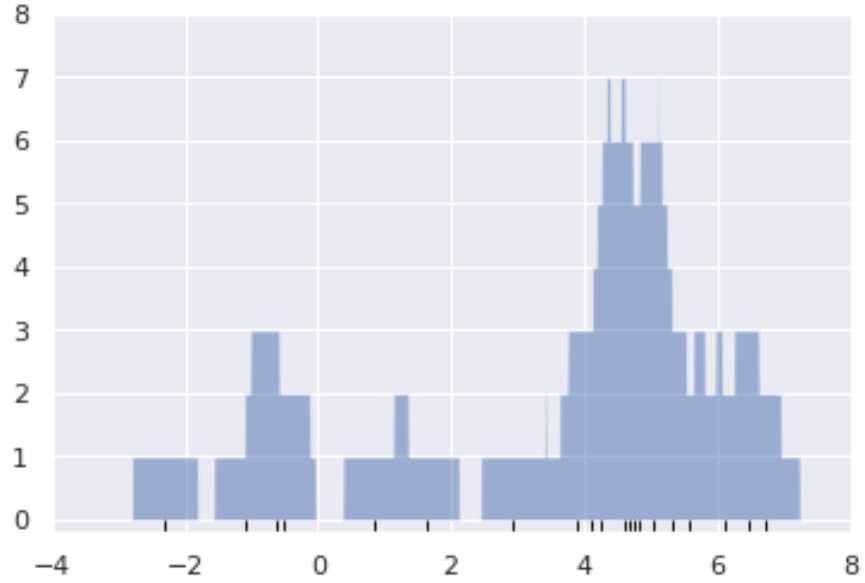


Why be at the mercy of binning parameters which are decided in a “data-blind” manner ?

Choice of bin sizes and bin edge locations can lead to different looking histograms for same data

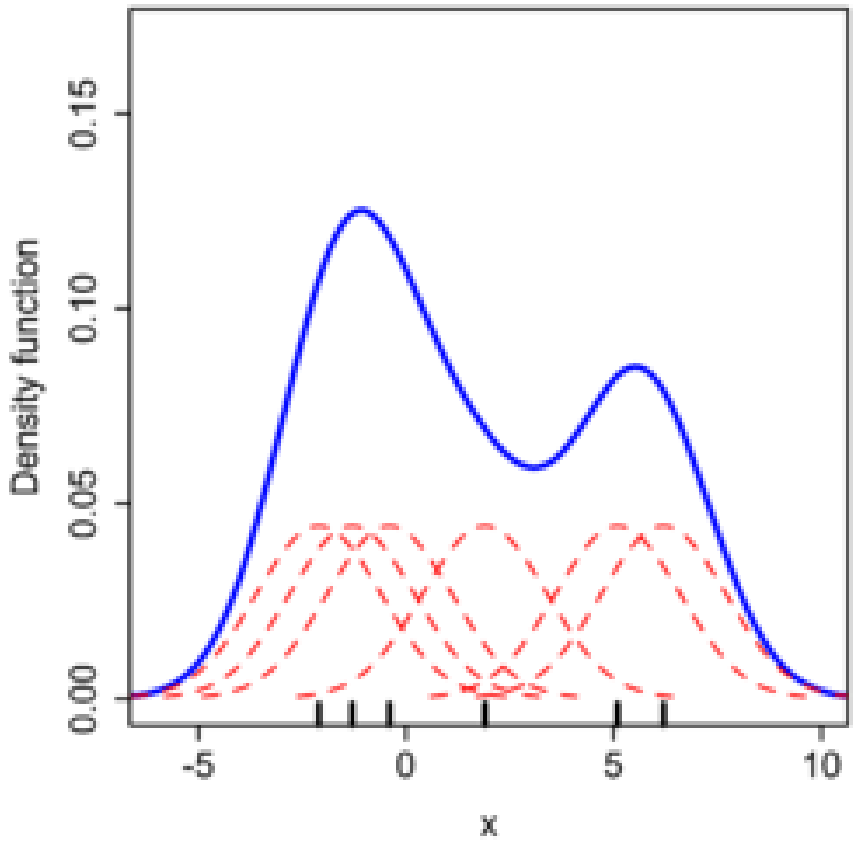
Alternative: Let data determine bin locations (and density at a location x).



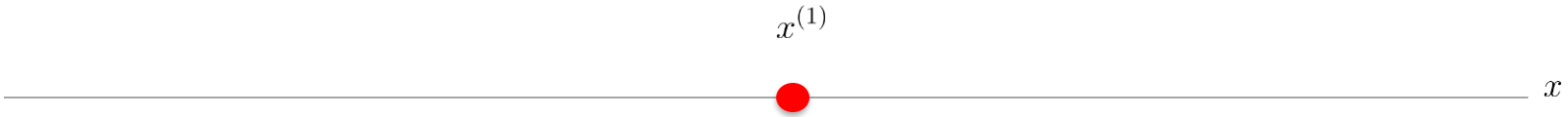


Choice of bin sizes and bin edge locations can lead to different looking histograms for same data

Alternative: Let data determine bin locations (and density at a location x).

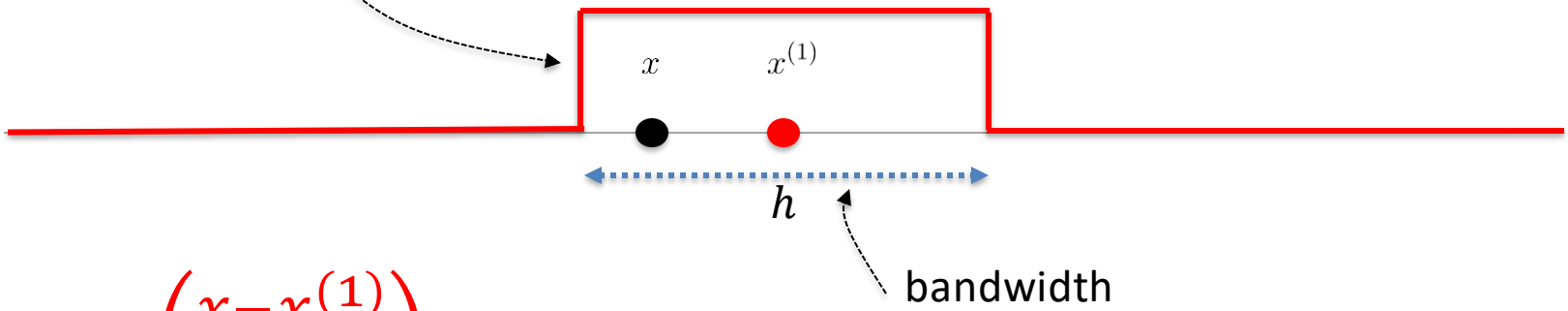


Kernel Density Estimation



- $p(x)$?

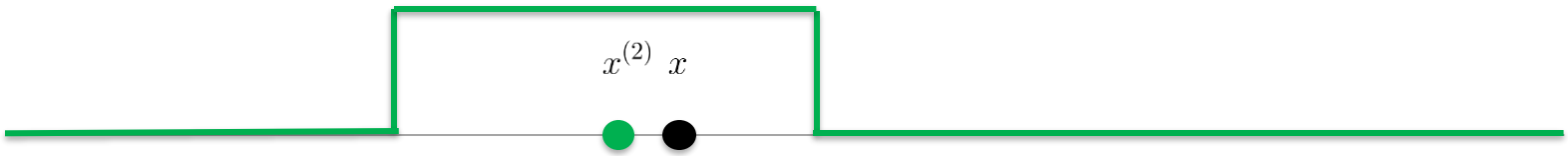
Kernel Density Estimation



- $K\left(\frac{x-x^{(1)}}{h}\right) = 1$

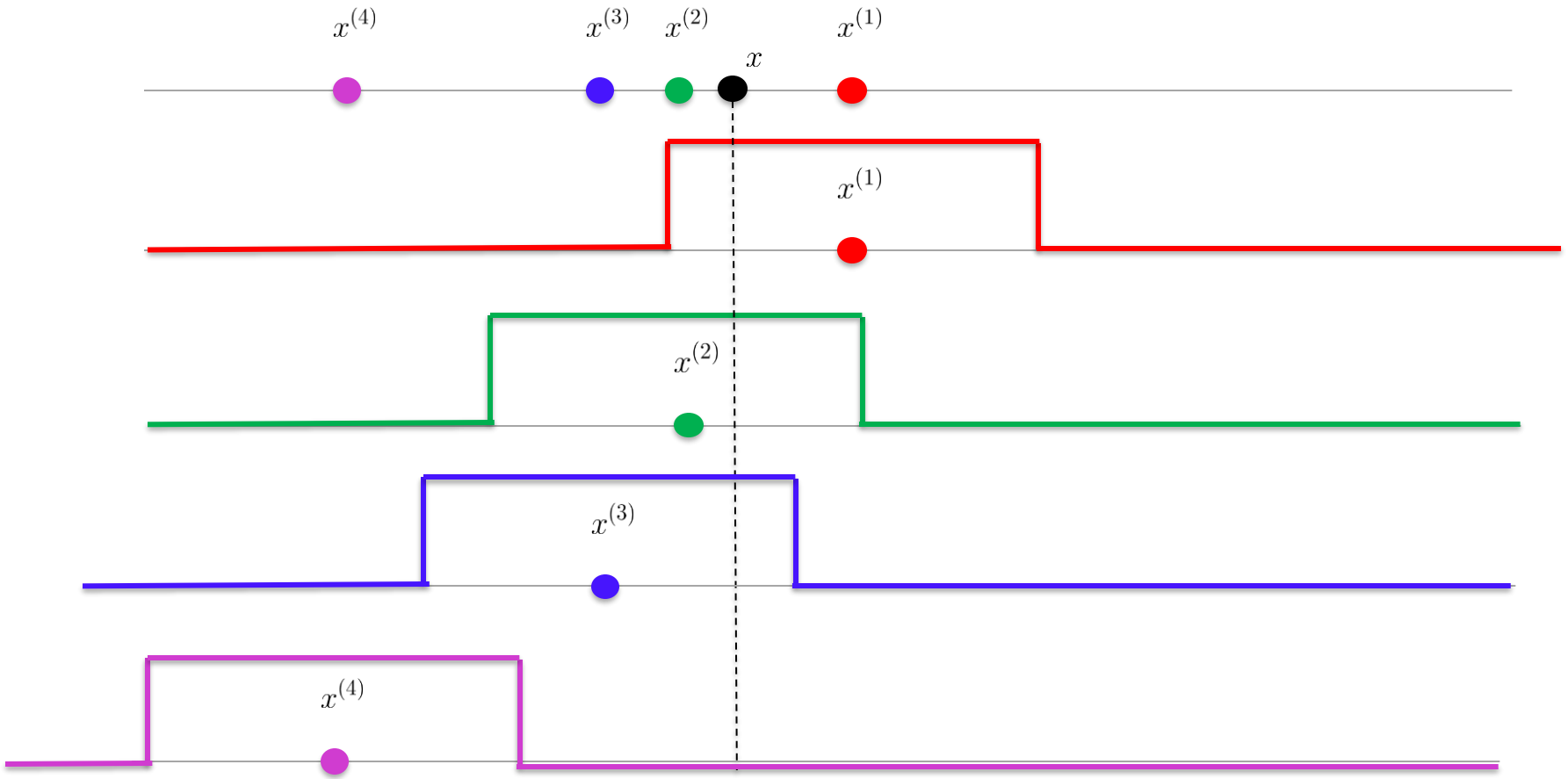
Kernel Density Estimation

Kernel Density Estimation

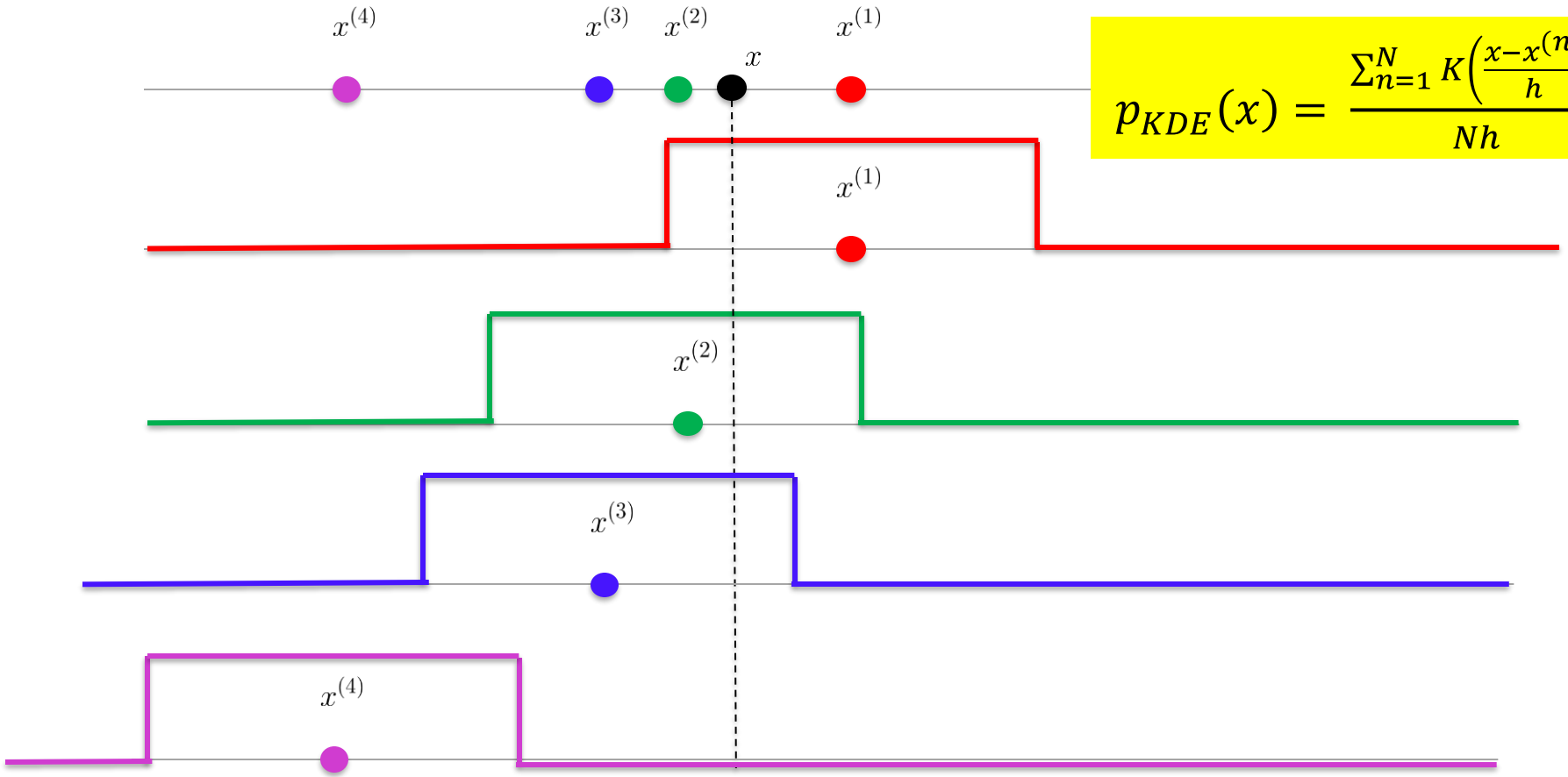


- $K\left(\frac{x-x^{(2)}}{h}\right) = 1$

Parzen Window Density Estimation



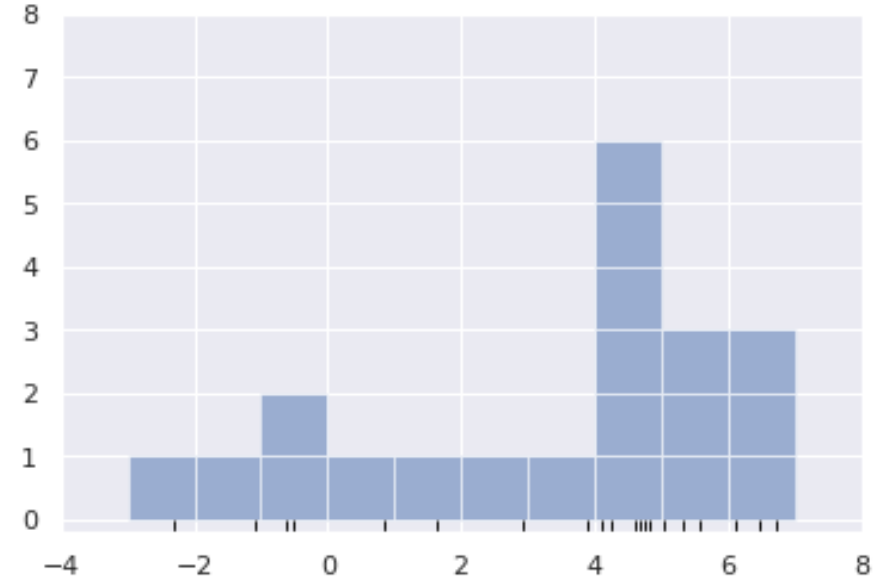
Parzen Window Density Estimation

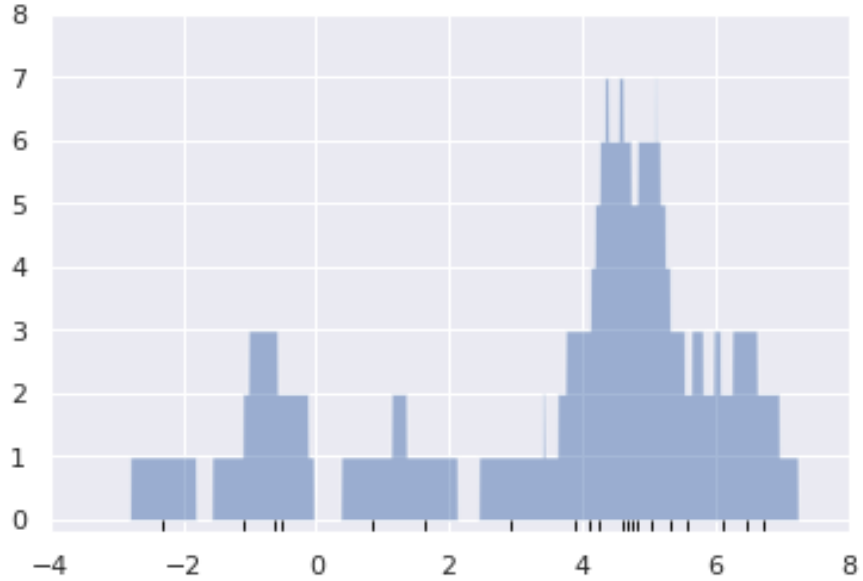


$$p_{KDE}(x) = \frac{\sum_{n=1}^{N} K\left(\frac{x-x^{(n)}}{h}\right)}{Nh}$$

Choice of bin sizes and bin edge locations can lead to different looking histograms for same data

Alternative: Let data determine bin locations (and density at a location x).

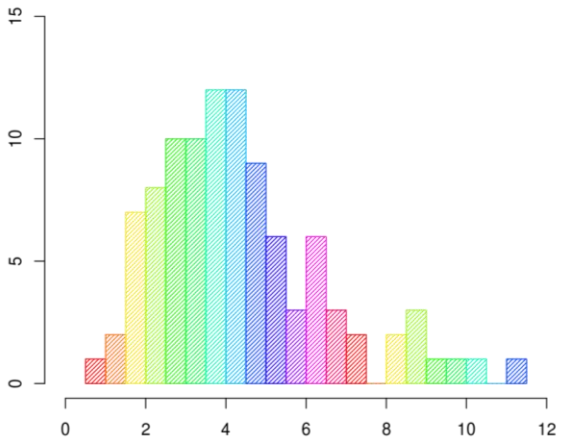


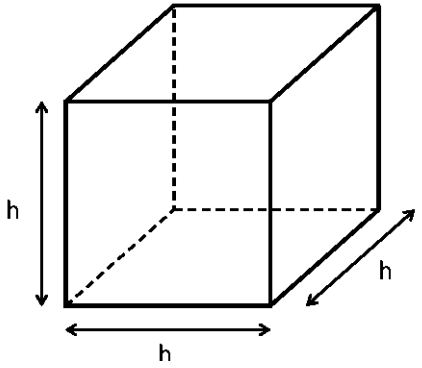


$$p_{KDE}(x) = \frac{\sum_{n=1}^{N} K\left(\frac{x-x^{(n)}}{h}\right)}{Nh}$$

Parzen Window Density Estimation

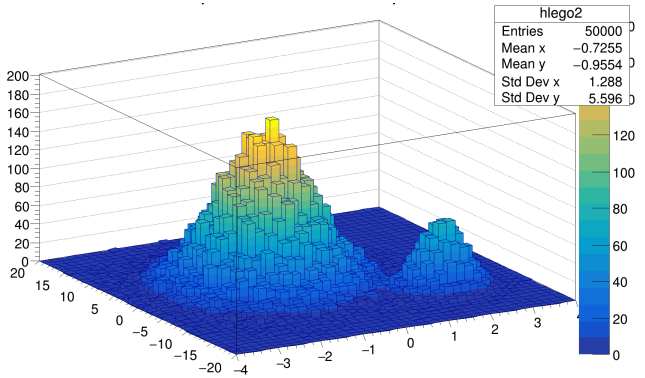
- $V = h^D$





$$p_H(x) = \frac{1}{N}\frac{[\#\ of\ x^{(k}\ in\ same\ bin\ as\ x]}{[width\ of\ bin]}$$

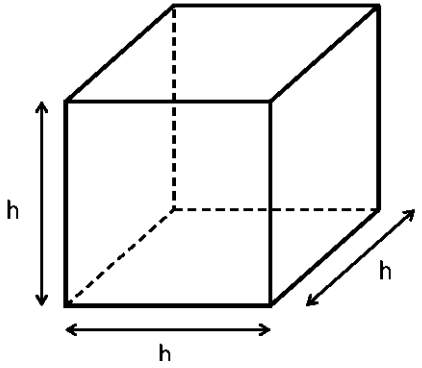
$$p_{KDE}(x) = \frac{1}{N}\frac{\sum_{n=1}^{N} K\left(\frac{x-x^{(n)}}{h}\right)}{V}$$

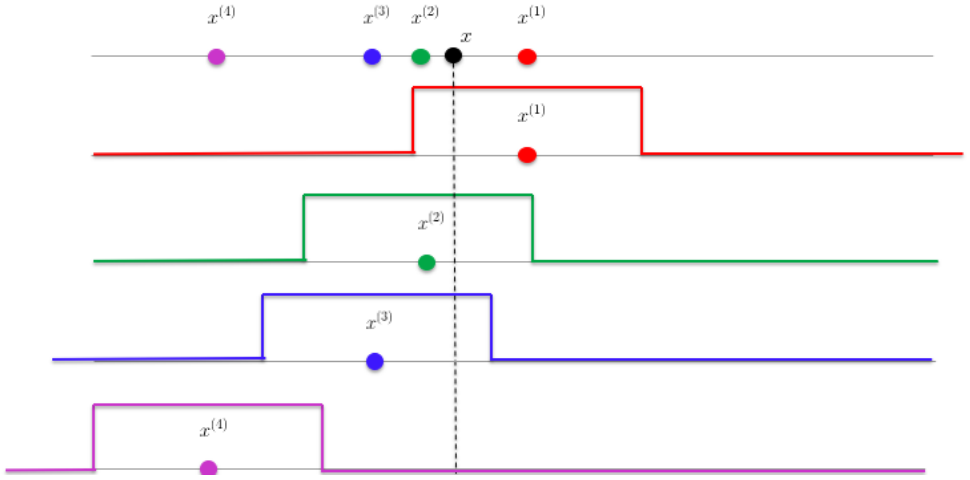


Parzen Window Density Estimation

$$K(u) = \begin{cases} 1 & |u_j| < 1/2 \quad \forall j = 1 \dots D \\ 0 & otherwise \end{cases}$$

$$p_{KDE}(x) = \frac{1}{N} \frac{\sum_{n=1}^{N} K\left(\frac{x - x^{(n)}}{h}\right)}{V}$$



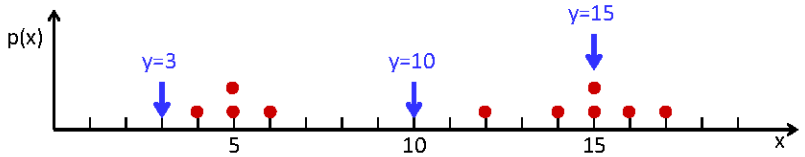


$$p_{KDE}(x) = \frac{1}{N} \frac{\sum_{n=1}^{N} K\left(\frac{x-x^{(n)}}{h}\right)}{V}$$

$$V = h^D$$

Exercise

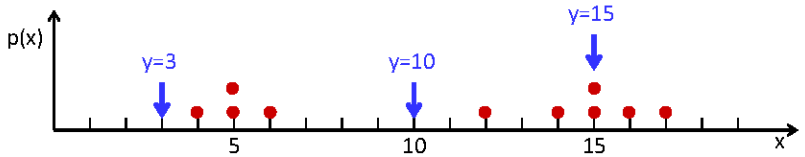
- Given dataset $X = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$, use Parzen windows to estimate the density $p(x)$ at $y = 3,10,15$; use $h = 4$
- Solution
  - Let's first draw the dataset to get an idea of the data



  - Let's now estimate $p(y = 3)$

Exercise

- Given dataset $X = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$, use Parzen windows to estimate the density $p(x)$ at $y = 3,10,15$; use $h = 4$
- Solution
  - Let's first draw the dataset to get an idea of the data

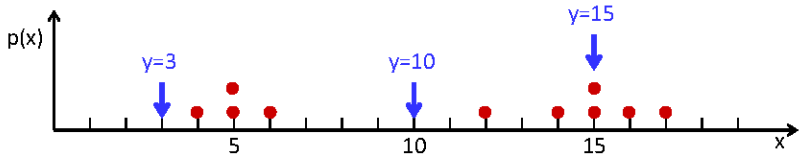


  - Let's now estimate $p(y = 3)$

$$p(y = 3) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(n)}}{h}\right)$$

Exercise

- Given dataset $X = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$, use Parzen windows to estimate the density $p(x)$ at $y = 3,10,15$; use $h = 4$
- Solution
  - Let's first draw the dataset to get an idea of the data

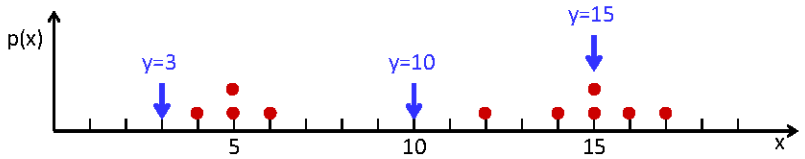


  - Let's now estimate $p(y = 3)$

$$p(y = 3) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{x - x^{(n)}}{h}\right) = \frac{1}{10 \times 4^1}\left[K\left(\frac{3-4}{4}\right) + K\left(\frac{3-5}{4}\right) + \cdots K\left(\frac{3-17}{4}\right)\right] = 0.0025$$

Exercise

- Given dataset $X = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$, use Parzen windows to estimate the density $p(x)$ at $y = 3,10,15$; use $h = 4$
- Solution
  - Let's first draw the dataset to get an idea of the data



  - Let's now estimate $p(y = 3)$

$$p(y = 3) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{x - x^{(n)}}{h}\right) = \frac{1}{10 \times 4^1}\left[K\left(\frac{3-4}{4}\right) + K\left(\frac{3-5}{4}\right) + \cdots K\left(\frac{3-17}{4}\right)\right] = 0.0025$$

  - Similarly

$$p(y = 10) = \frac{1}{10 \times 4^1}[0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0] = 0$$

$$p(y = 15) = \frac{1}{10 \times 4^1}[0 + 0 + 0 + 0 + 0 + 1 + 1 + 1 + 1 + 0] = 0.1$$

Smooth kernels

The Parzen window has several drawbacks

- It yields density estimates that have discontinuities
- It weights equally all points x_i, regardless of their distance to the estimation point x

Smooth kernels

The Parzen window has several drawbacks

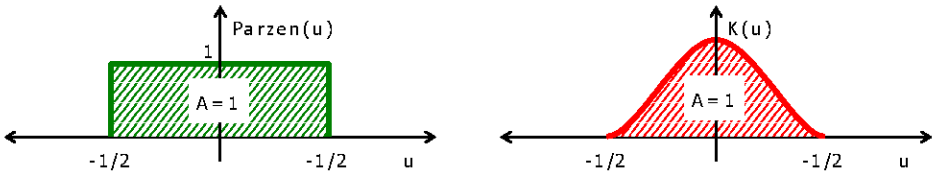
- It yields density estimates that have discontinuities
- It weights equally all points x_i, regardless of their distance to the estimation point x

For these reasons, the Parzen window is commonly replaced with a smooth kernel function $K(u)$

$$\int_{R^D} K(x)dx = 1$$

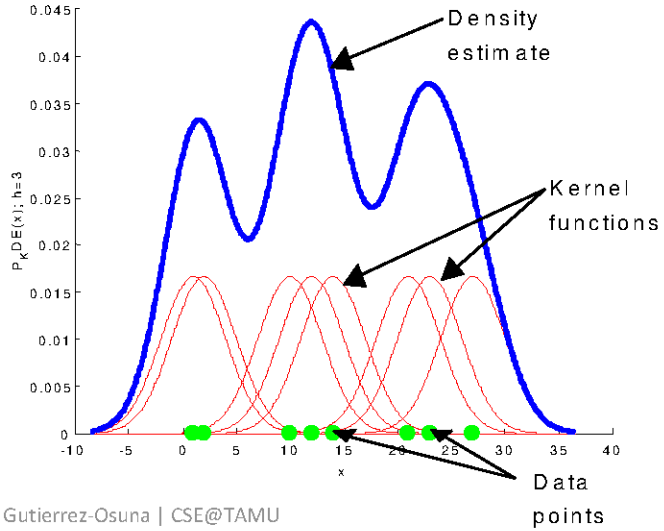
- Usually, but not always, $K(u)$ will be a radially symmetric and unimodal pdf, such as the Gaussian $K(x) = (2\pi)^{-D/2} e^{-\frac{1}{2}x^T x}$
- Which leads to the density estimate

$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(k}}{h}\right)$$



Interpretation

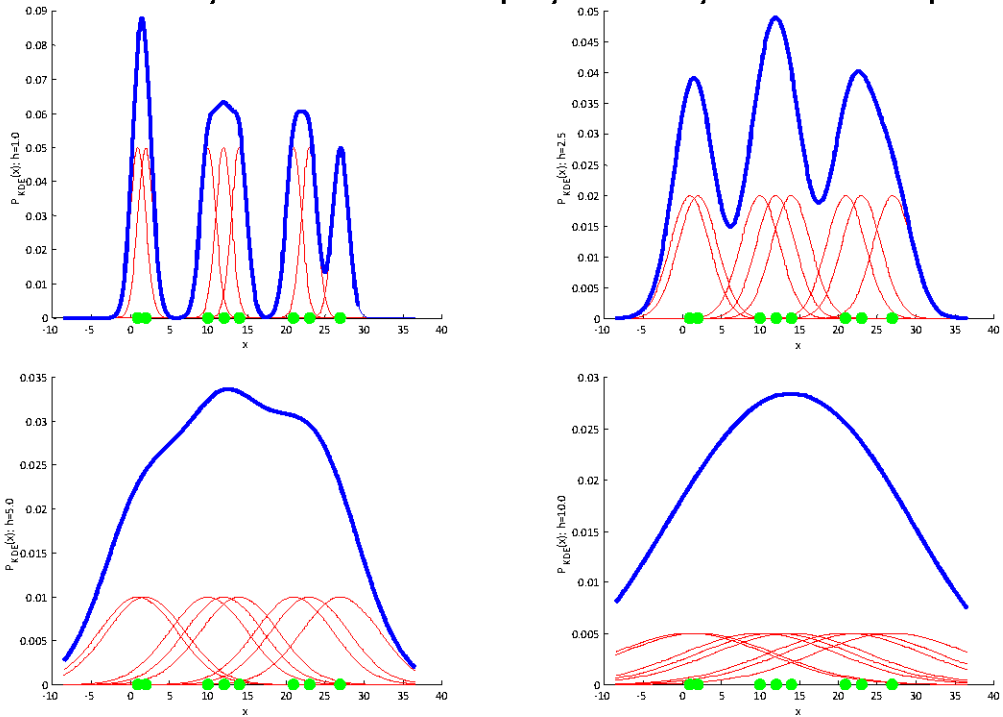
- Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of "bumps"
- The kernel function determines the shape of the bumps
- The parameter h, also called the smoothing parameter or bandwidth, determines their width



Bandwidth selection

The problem of choosing h is crucial in density estimation

- A large h will over-smooth the DE and mask the structure of the data
- A small h will yield a DE that is spiky and very hard to interpret



Maximum likelihood cross-validation

Maximum likelihood cross-validation

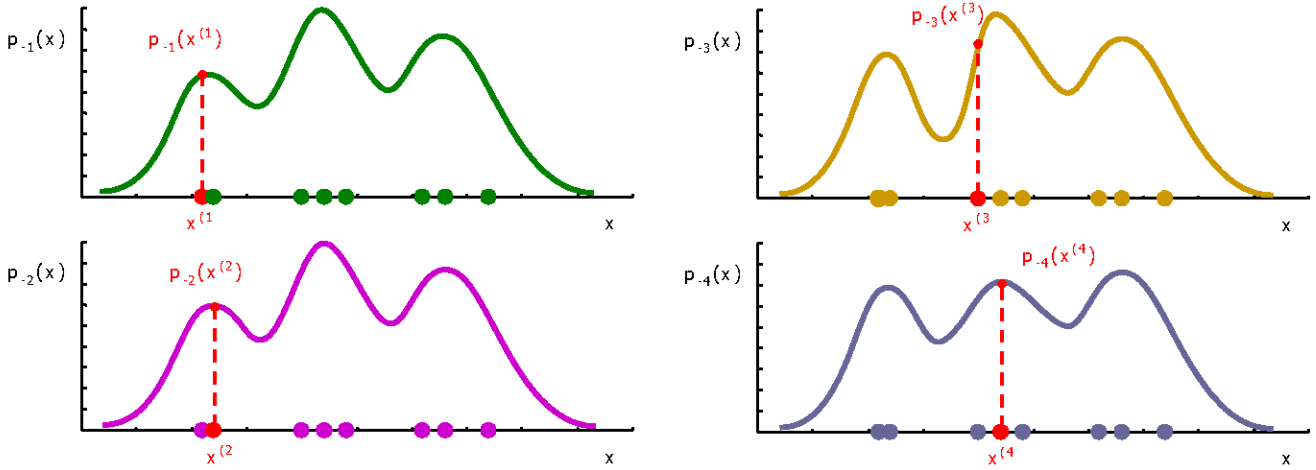
- The ML estimate of h is degenerate since it yields $h_{ML} = 0$, a density estimate with Dirac delta functions at each training data point

Maximum likelihood cross-validation

- The ML estimate of h is degenerate since it yields $h_{ML} = 0$, a density estimate with Dirac delta functions at each training data point
- A practical alternative is to maximize the "pseudo-likelihood" computed using leave-one-out cross-validation

$$h^* = \arg\max\left\{\frac{1}{N}\sum_{n=1}^{N} log p_{-n}\left(x^{(n)}\right)\right\}$$

$$where\ p_{-n}\left(x^{(n)}\right) = \frac{1}{(N-1)h}\sum_{\substack{m=1 \\ m\neq n}}^{N} K\left(\frac{x^{(n} - x^{(m}}{h}\right)$$


p-1(x)
p-1(x(1)
x(1
x
p-2(x)
p-2(x(2)
x(2
x
p-3(x)
p-3(x(3)
x(3
x
p-4(x)
p-4(x(4)
x(4
x

Multivariate density estimation

For the multivariate case, the KDE is

$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

- Notice that the bandwidth h is the same for all the axes, so this density estimate will be weight all the axis equally
- If one or several of the features has larger spread than the others, we should use a vector of smoothing parameters or even a full covariance matrix, which complicates the procedure

Multivariate density estimation

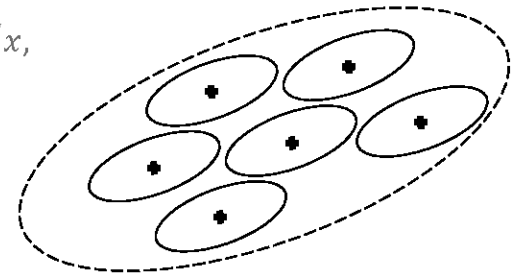
For the multivariate case, the KDE is

$$p_{KDE}(x) = \frac{1}{Nh^D}\sum_{n=1}^{N} K\left(\frac{x-x^{(n}}{h}\right)$$

- Notice that the bandwidth h is the same for all the axes, so this density estimate will be weight all the axis equally
- If one or several of the features has larger spread than the others, we should use a vector of smoothing parameters or even a full covariance matrix, which complicates the procedure

There are two basic alternatives to solve the scaling problem without having to use a more general KDE

- Pre-scaling each axis (normalize to unit variance, for instance)
- Pre-whitening the data (linearly transform so $\Sigma = I$), estimate the density, and then transform back [Fukunaga]
  - The whitening transform is $y = \Lambda^{-1/2}M^T x$, where Λ and M are the eigenvalue and eigenvector matrices of Σ
  - Fukunaga's method is equivalent to using a hyper-ellipsoidal kernel



Product kernels

A good alternative for multivariate KDE is the product kernel

$$p_{PKDE}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(x, x^{(n}, h_1, \ldots h_D\right)$$

$$where\ \ K\left(x, x^{(n}, h_1, \ldots h_D\right) = \frac{1}{h_1 \ldots h_D}\prod_{d=1}^{D} K_d\left(\frac{x_d - x_d^{(n}}{h_d}\right)$$

- The product kernel consists of the product of one-dimensional kernels
  - Typically the same kernel function is used in each dimension ($K_d(x) = K(x)$), and only the bandwidths are allowed to differ

Product kernels

A good alternative for multivariate KDE is the product kernel

$$p_{PKDE}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(x, x^{(n}, h_1, \ldots h_D\right)$$

$$where\ \ K\left(x, x^{(n}, h_1, \ldots h_D\right) = \frac{1}{h_1 \ldots h_D}\prod_{d=1}^{D} K_d\left(\frac{x_d - x_d^{(n}}{h_d}\right)$$

- The product kernel consists of the product of one-dimensional kernels
  - Typically the same kernel function is used in each dimension ($K_d(x) = K(x)$), and only the bandwidths are allowed to differ

$K\left(x, x^{(n}, h_1, \ldots h_D\right)$ uses kernel independence
features are independent

Product kernels

A good alternative for multivariate KDE is the product kernel

$$p_{PKDE}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(x, x^{(n}, h_1, \ldots h_D\right)$$

$$where\ \ K\left(x, x^{(n}, h_1, \ldots h_D\right) = \frac{1}{h_1 \ldots h_D}\prod_{d=1}^{D} K_d\left(\frac{x_d - x_d^{(n}}{h_d}\right)$$

- The product kernel consists of the product of one-dimensional kernels
  - Typically the same kernel function is used in each dimension ($K_d(x) = K(x)$), and only the bandwidths are allowed to differ

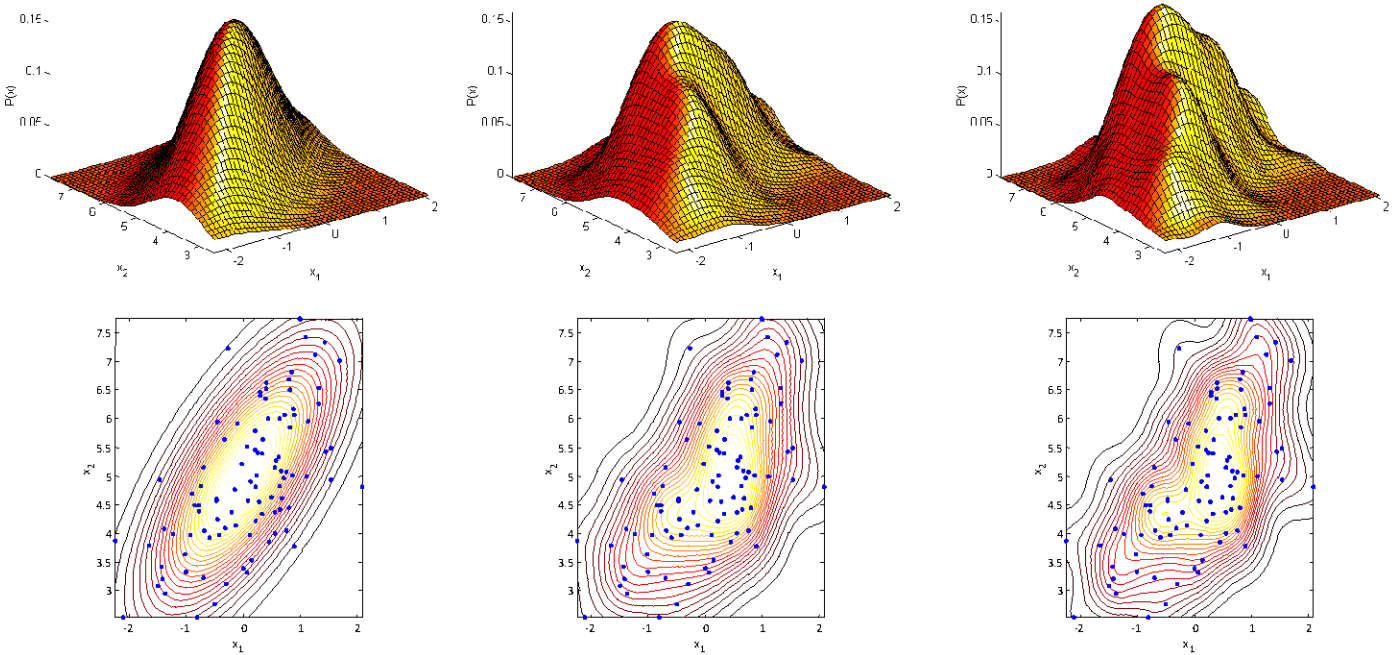
- Note that although $K\left(x, x^{(n}, h_1, \ldots h_D\right)$ uses kernel independence does not imply we assume the features are independent
  - If we assumed feature independence, the DE would have the expression

$$p_{FEAT-IND}(x) = \prod_{d=1}^{D} \frac{1}{Nh^D}\sum_{i=1}^{N} K_d\left(\frac{x_d - x_d^{(n}}{h_d}\right)$$

  - Notice how the order of the summation and product are reversed compared to the product kernel

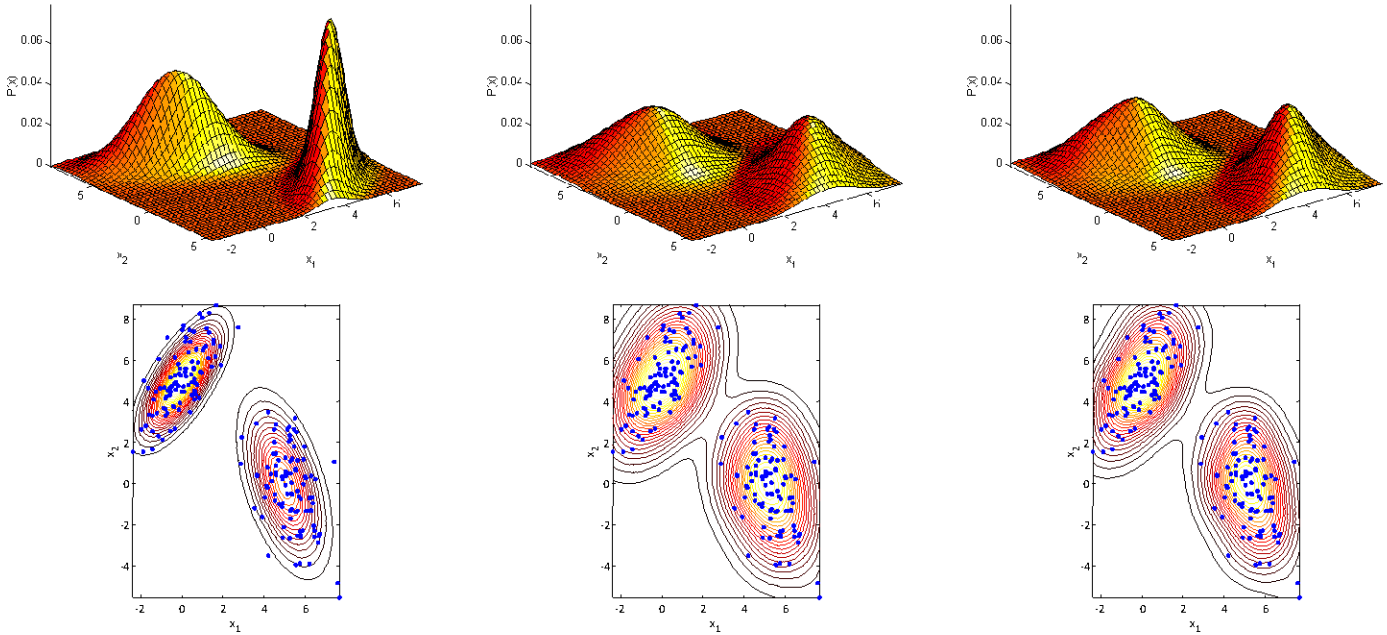
Example I

- This example shows the product KDE of a bivariate unimodal Gaussian
  - 100 data points were drawn from the distribution
  - The figures show the true density (left) and the estimates using $h = 1.06\sigma N^{-1/5}$ (middle) and $h = 0.9AN^{-1/5}$ (right)



Example II

- This example shows the product KDE of a bivariate <u>bimodal</u> Gaussian
  - 100 data points were drawn from the distribution
  - The figures show the true density (left) and the estimates using $h = 1.06\sigma N^{-1/5}$ (middle) and $h = 0.9AN^{-1/5}$ (right)



KDE

- https://scikit-learn.org/stable/modules/density.html

```
>>> from sklearn.neighbors.kde import KernelDensity
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> kde = KernelDensity(kernel='gaussian', bandwidth=0.2).fit(X)
>>> kde.score_samples(X)
array([-0.41075698, -0.41075698, -0.41076071, -0.41075698, -0.41075698,
       -0.41076071])
```

$$p(x) \cong \frac{k}{NV} \text{ where } \begin{cases} V & volume\ surrounding\ x \\ N & total\ \#examples \\ k & \#examples\ inside\ V \end{cases}$$

- We can fix V and determine k from the data. This leads to **kernel density estimation** (KDE), the subject of this lecture
- We can fix k and determine V from the data. This gives rise to the **k-nearest-neighbor** (kNN) approach

– The probability that a vector x, drawn from a distribution $p(x)$, will fall in a given region $\Re$ of the sample space is

$$P = \int_{\Re} p(x')dx'$$

- The probability that a vector x, drawn from a distribution $p(x)$, will fall in a given region $\Re$ of the sample space is

$$P = \int_{\Re} p(x')dx'$$

- Suppose now that N vectors $\{x^{(1}, x^{(2}, \ldots x^{(N}\}$ are drawn from the distribution; the probability that k of these N vectors fall in $\Re$ is given by

- The probability that a vector x, drawn from a distribution $p(x)$, will fall in a given region $\Re$ of the sample space is

$$P = \int_{\Re} p(x')dx'$$

- Suppose now that N vectors $\{x^{(1}, x^{(2}, \ldots x^{(N}\}$ are drawn from the distribution; the probability that k of these N vectors fall in $\Re$ is given by the binomial distribution

$$P(k) = \binom{N}{k} P^k (1-P)^{N-k}$$

- The probability that a vector x, drawn from a distribution $p(x)$, will fall in a given region $\Re$ of the sample space is

$$P = \int_{\Re} p(x')dx'$$

- Suppose now that N vectors $\{x^{(1}, x^{(2}, \ldots x^{(N}\}$ are drawn from the distribution; the probability that k of these N vectors fall in $\Re$ is given by the binomial distribution

$$P(k) = \binom{N}{k} P^k (1-P)^{N-k}$$

- It can be shown (from the properties of the binomial p.m.f.) that the mean and variance of the ratio k/N are

$$E\left[\frac{k}{N}\right] = P \quad \text{and} \quad var\left[\frac{k}{N}\right] = E\left[\left(\frac{k}{N} - P\right)^2\right] = \frac{P(1-P)}{N}$$

- The probability that a vector x, drawn from a distribution $p(x)$, will fall in a given region $\Re$ of the sample space is

$$P = \int_{\Re} p(x')dx'$$

- Suppose now that N vectors $\{x^{(1}, x^{(2}, \dots x^{(N}\}$ are drawn from the distribution; the probability that k of these N vectors fall in $\Re$ is given by the binomial distribution

$$P(k) = \binom{N}{k} P^k (1-P)^{N-k}$$

- It can be shown (from the properties of the binomial p.m.f.) that the mean and variance of the ratio k/N are

$$E\left[\frac{k}{N}\right] = P \quad \text{and} \quad var\left[\frac{k}{N}\right] = E\left[\left(\frac{k}{N} - P\right)^2\right] = \frac{P(1-P)}{N}$$

- Therefore, as $N \to \infty$ the distribution becomes sharper (the variance gets smaller), so we can expect that a good estimate of the probability P can be obtained from the mean fraction of the points that fall within $\Re$

$$P \cong \frac{k}{N}$$

[Bishop, 1995]

– On the other hand, if we assume that $\mathfrak{R}$ is so small that $p(x)$ does not vary appreciably within it, then

$$\int_{\mathfrak{R}} p(x')dx' \cong p(x)V$$

  - where V is the volume enclosed by region $\mathfrak{R}$

– On the other hand, if we assume that $\Re$ is so small that $p(x)$ does not vary appreciably within it, then

$$\int_{\Re} p(x')dx' \cong p(x)V$$

  - where V is the volume enclosed by region $\Re$

– Merging with the previous result we obtain

$$\left.\begin{array}{l} P = \int_{\Re} p(x')dx' \cong p(x)V \\ P \cong \frac{k}{N} \end{array}\right\} \Rightarrow p(x) \cong \frac{k}{NV}$$

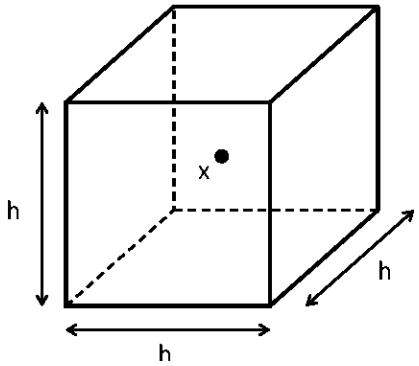
– In conclusion, the general expression for non-parametric density estimation becomes

$$p(x) \cong \frac{k}{NV} \text{ where } \begin{cases} V & volume\ surrounding\ x \\ N & total\ \#examples \\ k & \#examples\ inside\ V \end{cases}$$

Parzen windows

Problem formulation

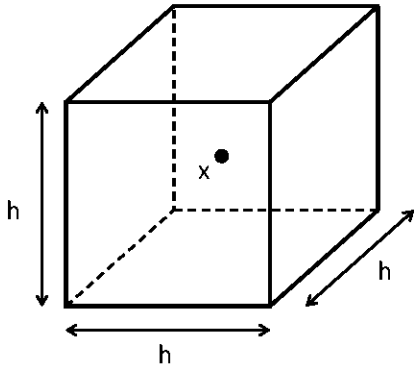
- Assume that the region $\Re$ that encloses the k examples is a hypercube with sides of length h centered at x
  - Then its volume is given by $V = h^D$, where D is the number of dimensions



Parzen windows

Problem formulation

- Assume that the region $\Re$ that encloses the k examples is a hypercube with sides of length h centered at x
  - Then its volume is given by $V = h^D$, where D is the number of dimensions



- To find the number of examples that fall within this region we define a kernel function $K(u)$

$$K(u) = \begin{cases} 1 & |u_j| < 1/2 \quad \forall j = 1 \ldots D \\ 0 & otherwise \end{cases}$$

  - This kernel, which corresponds to a unit hypercube centered at the origin, is known as a Parzen window or the naïve estimator
  - The quantity $K((x - x^{(n}/h)$ is then equal to unity if $x^{(n}$ is inside a hypercube of side h centered on x, and zero otherwise

[Bishop, 1995]

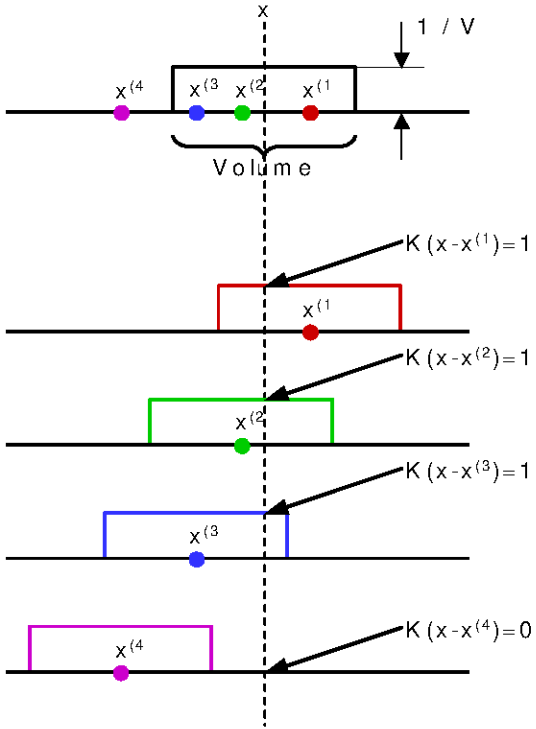
– The total number of points inside the hypercube is then

$$k = \sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

Substituting back into the expression for the density estimate

$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(n}}{h}\right)$$

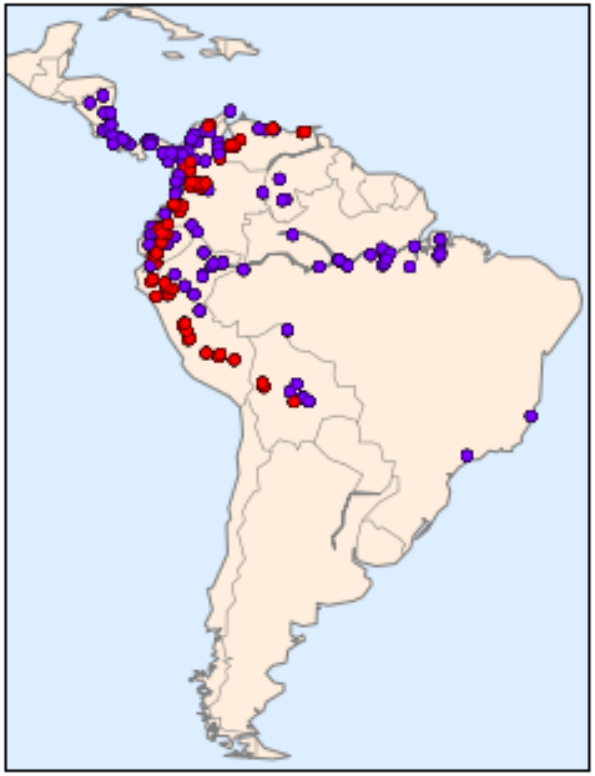
– Notice how the Parzen window estimate resembles the histogram, with the exception that the bin locations are determined by the data



https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html

Using KDE for visualization

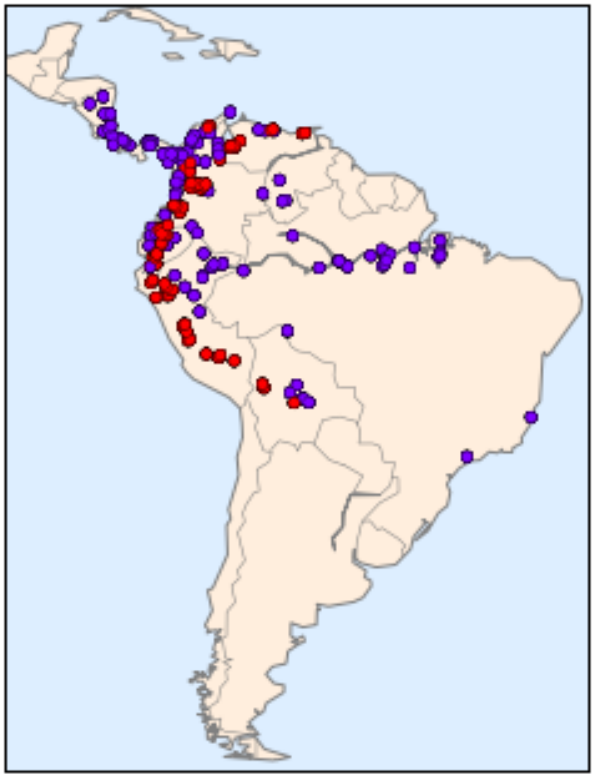
Geographic distributions of recorded observations of two South American mammals, *Bradypus variegatus* (the Brown-throated Sloth) and *Microryzomys minutus* (the Forest Small Rice Rat)



https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html

Using KDE for visualization

Geographic distributions of recorded observations of two South American mammals, *Bradypus variegatus* (the Brown-throated Sloth) and *Microryzomys minutus* (the Forest Small Rice Rat)

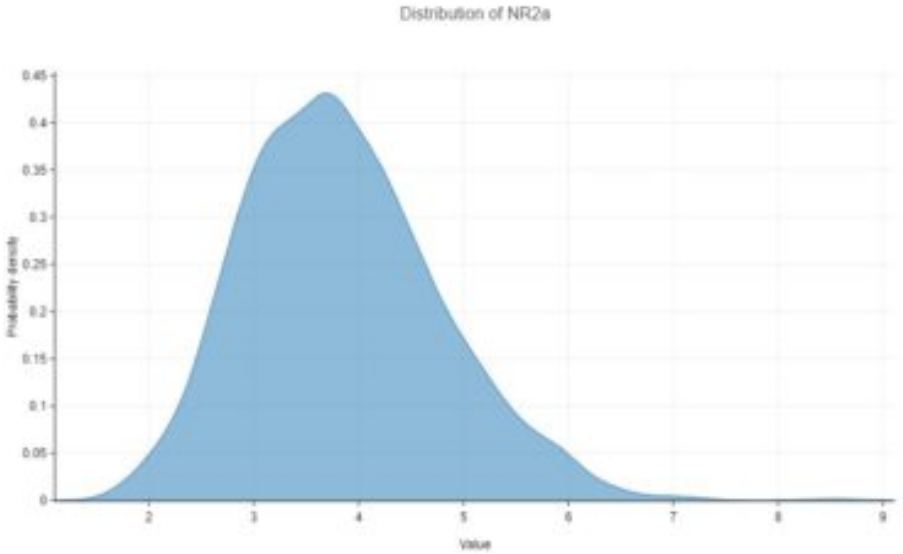


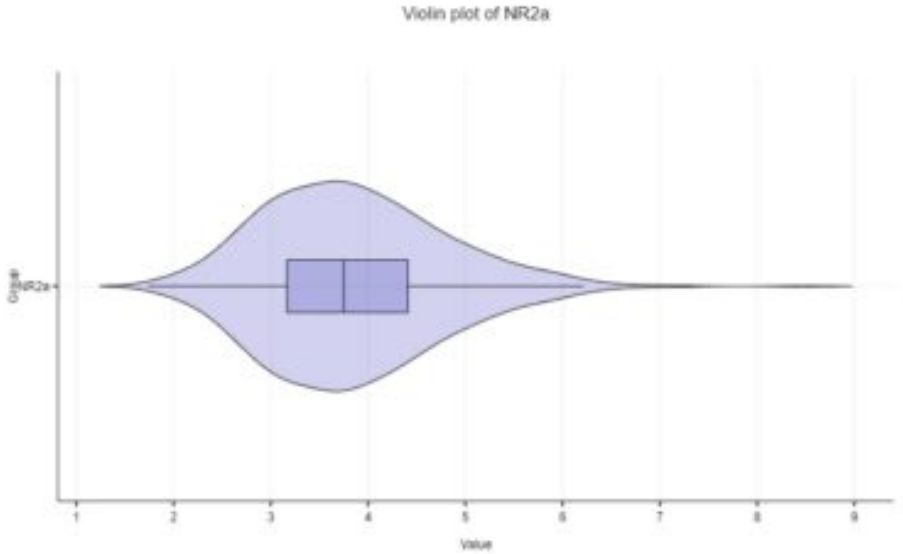
Bradypus Variegatus

Microryzomys Minutus

https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html

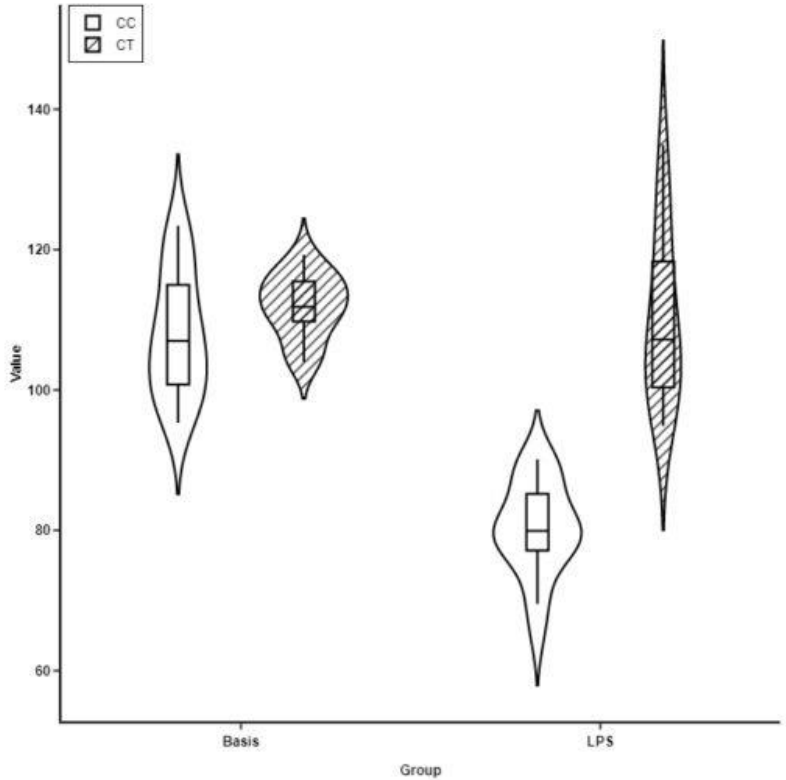
Using KDE for visualization – violin plot

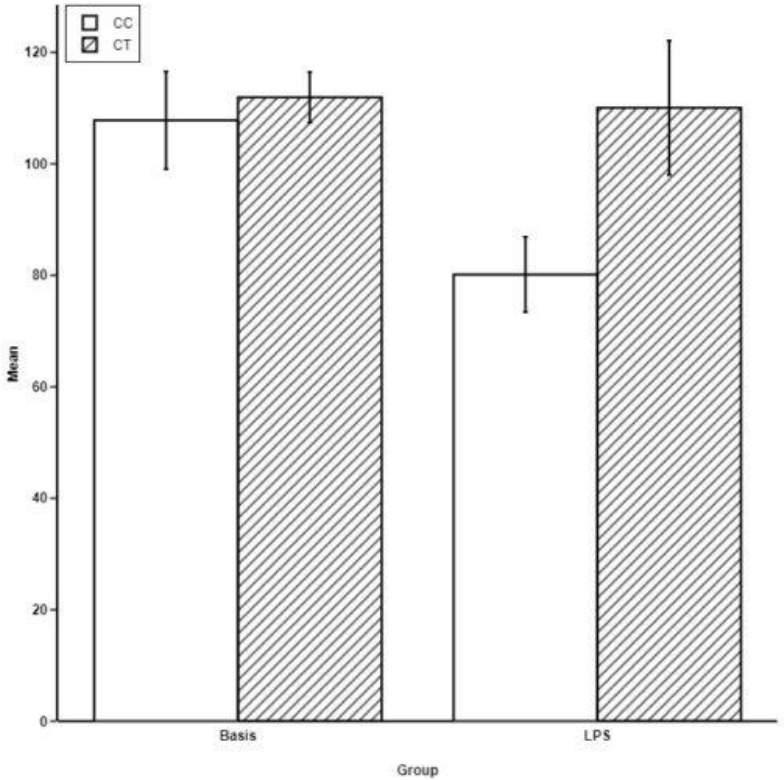




https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html

Using KDE for visualization – violin plot





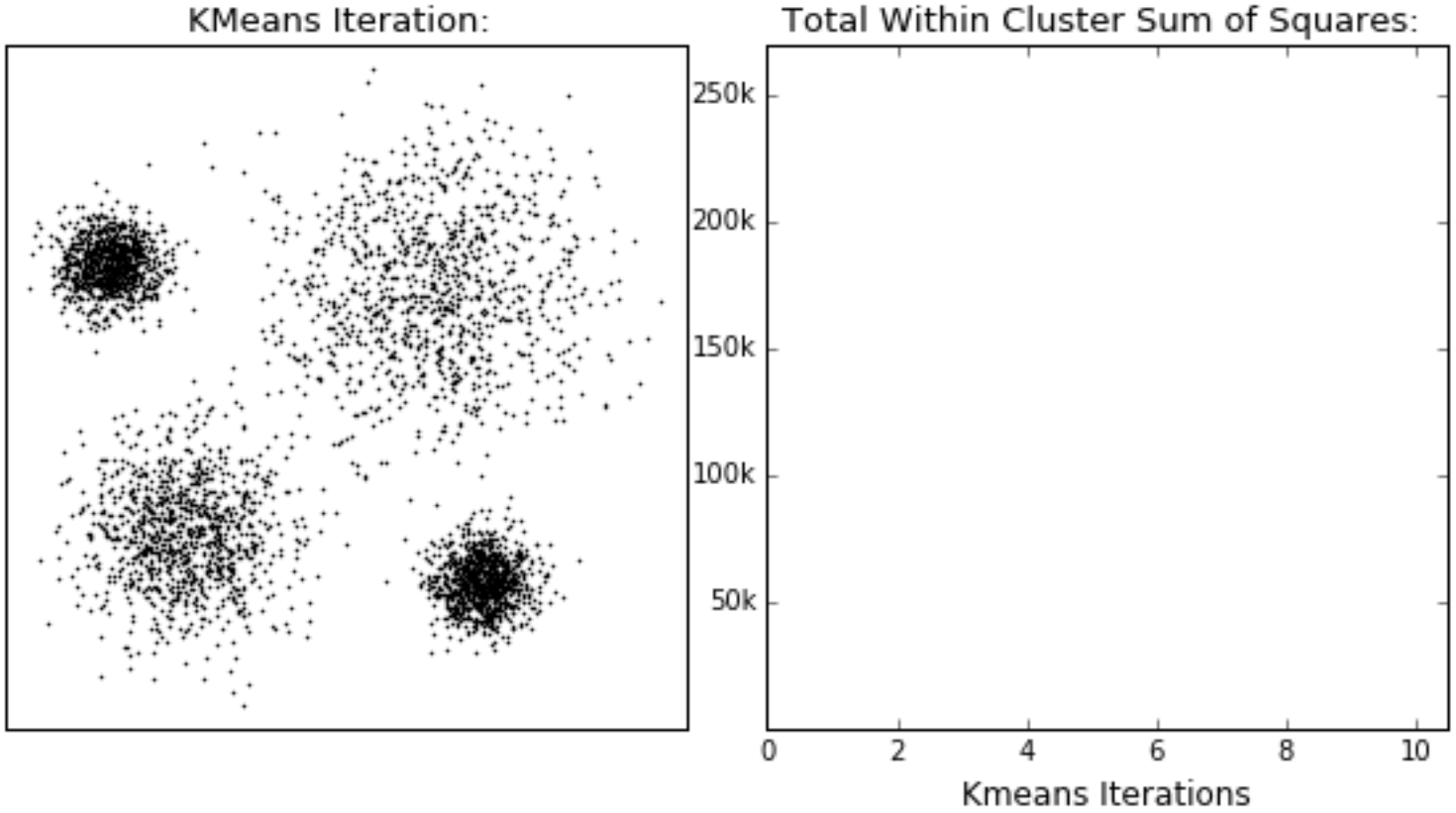
KDE

- Non-parametric density estimation → generative
- Advantage: Data-driven, Data-adaptive
- Disadvantage: Need to keep around all data samples to estimate the density, sensitive to bandwidth ‘h’ and choice of kernel

References

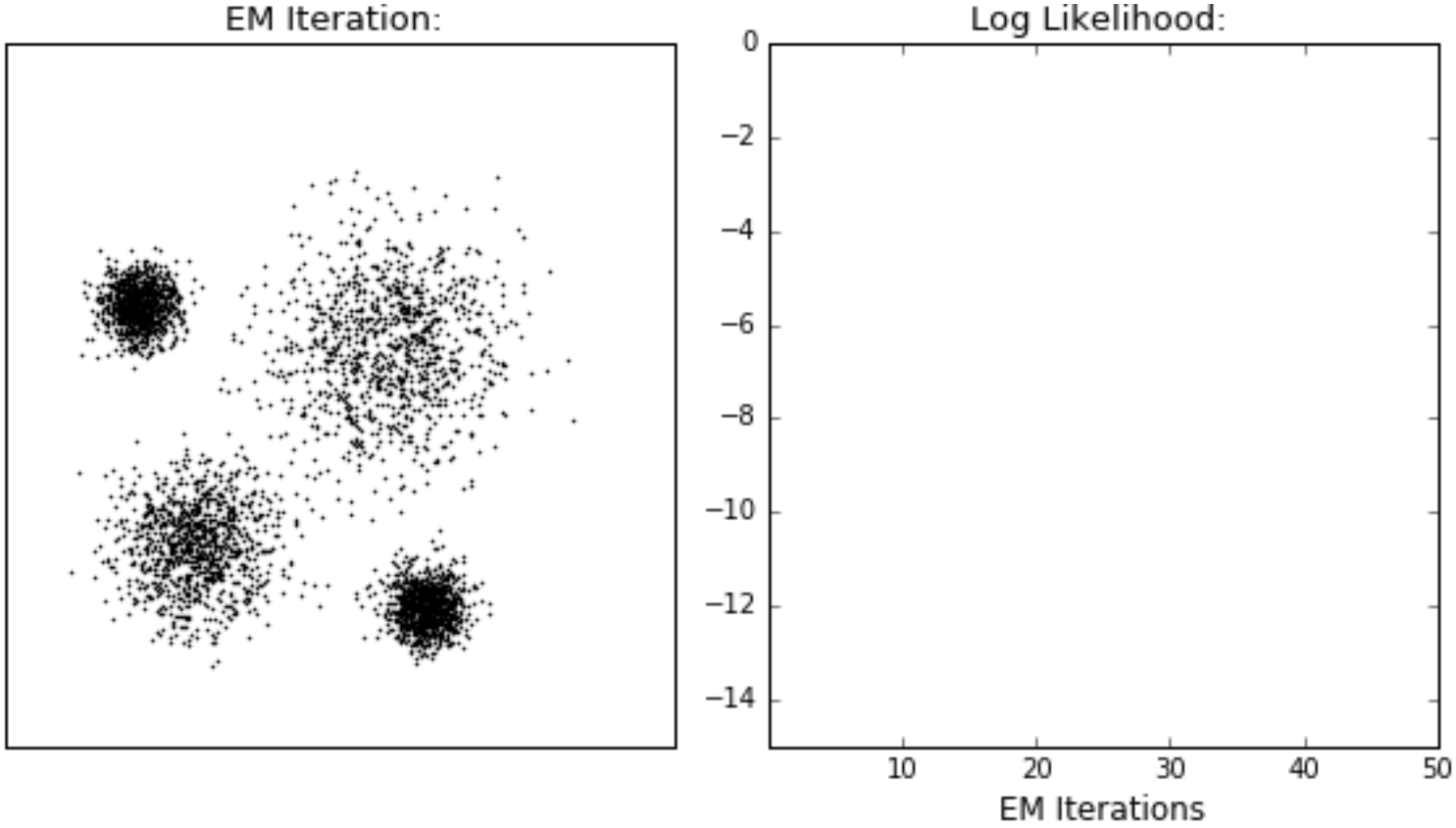
- Duda, Hart : 4.1, 4.2, 4.3.1 – 4.3.3, 4.4.1
- PRML (Bishop) : 2.5

K-means



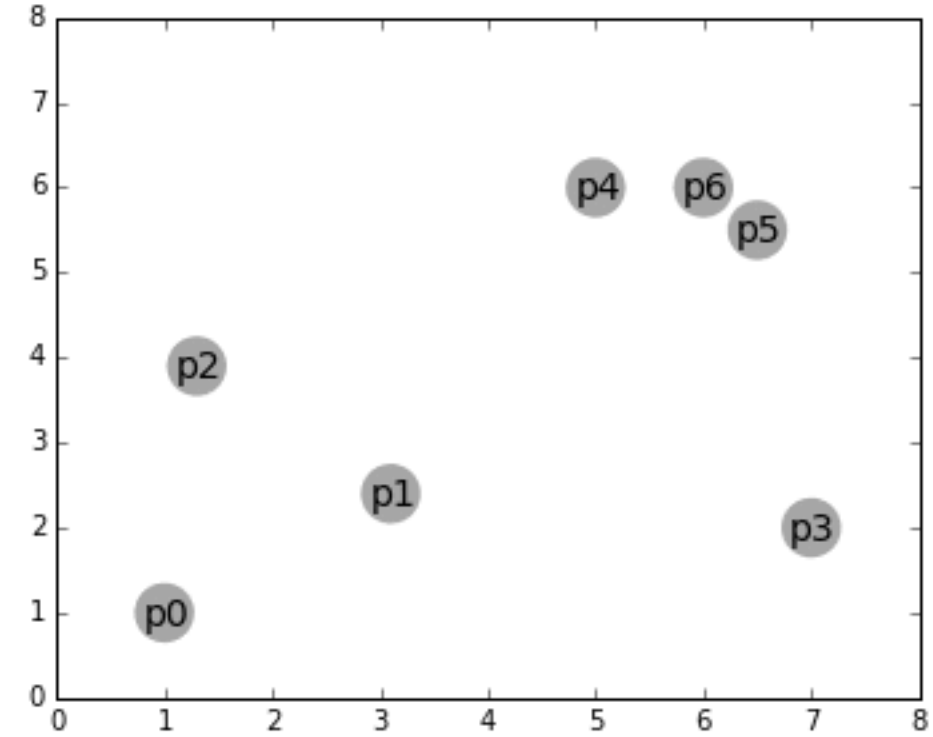
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

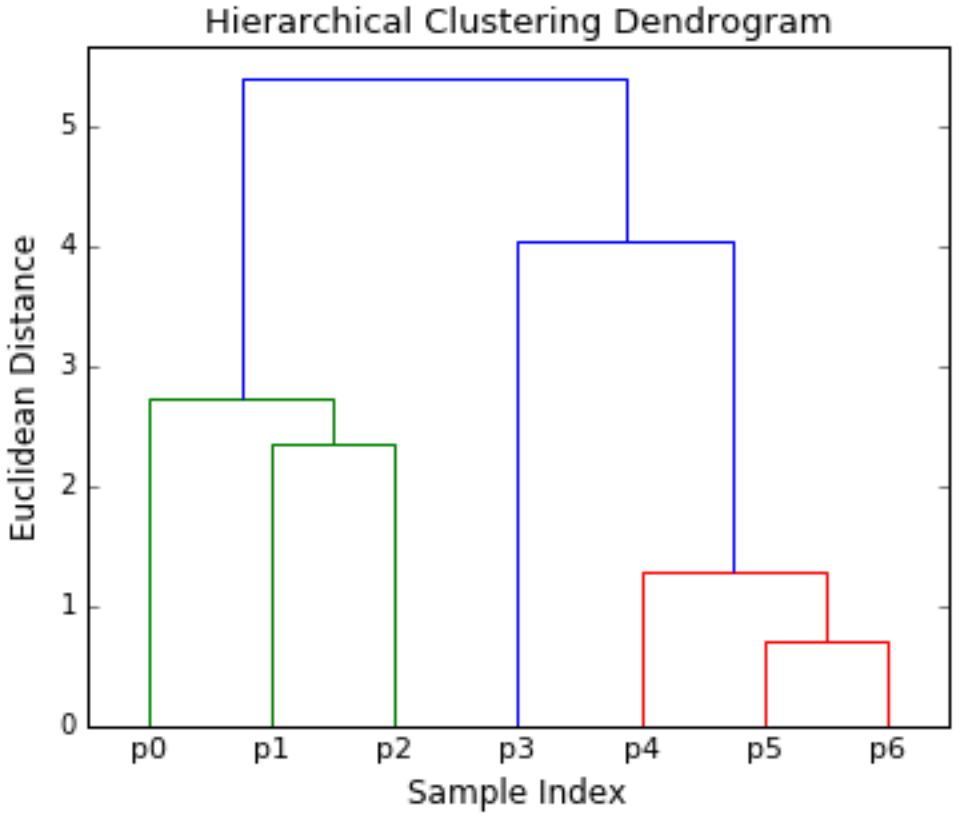
GMM



https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

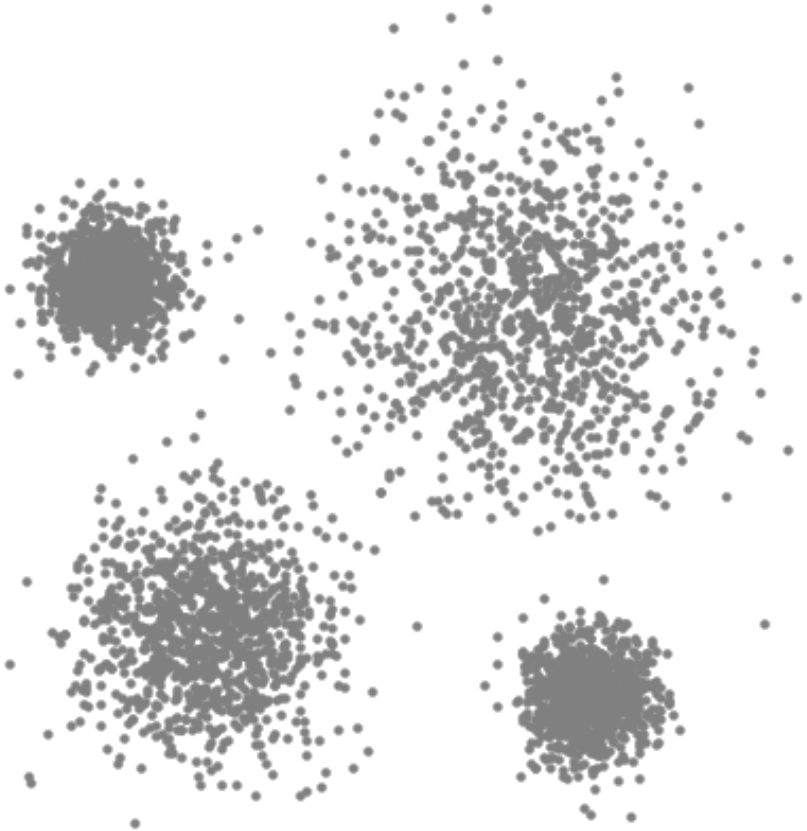
Hierarchical Clustering





https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Application of KDE – mean shift clustering



https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/