22.10.2024

Statistical Methods in AI (CS7.403)

Lecture-23: Non-parametric density estimation (KDE)

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

https://ravika.github.io

@vikataravi

Center for Visual Information Technology (CVIT)

IIIT Hyderabad

Unsupervised Learning → Density Estimation

Aka "learning without a teacher"

Feature Space $\mathcal{X}$

Words in a document ⇨ Word distribution (Probability of a word)

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

Parametric Density Estimation (GMM)

- Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k
- Iterate until convergence:
  - E-step: Evaluate the responsibilities given current parameters

$$\gamma_k^{(n)} = p(z^{(n)}|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\mu_j, \Sigma_j)}$$

  - M-step: Re-estimate the parameters given current responsibilities

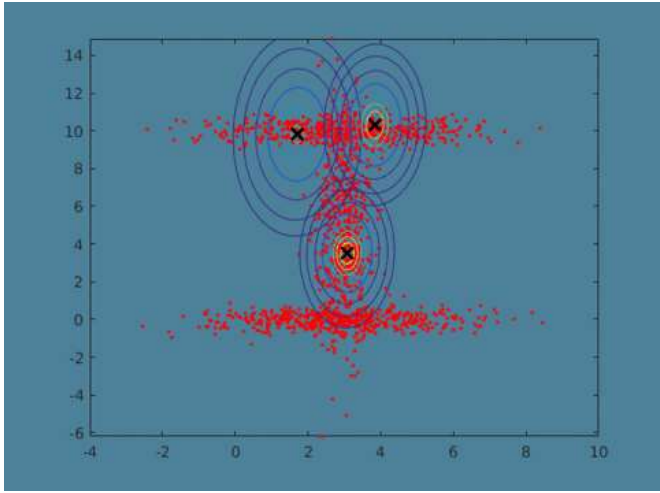
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$$

  - Evaluate log likelihood and check for convergence

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k) \right)$$



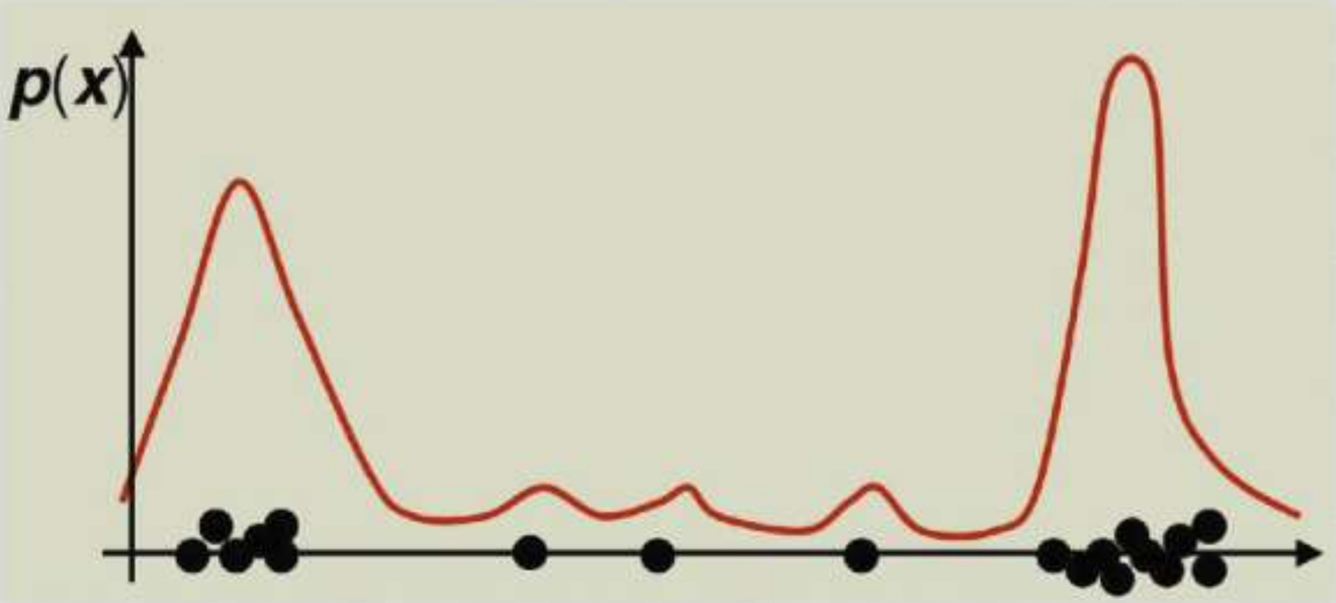
- In previous lectures we assumed that
  - The exact distribution of the data is known
  - Or at least the shape of the distribution is known

- In previous lectures we assumed that
  - The exact distribution of the data is known
  - Or at least the shape of the distribution is known
- But in real-life tasks
  - The exact distribution is never known
  - Assuming that its shape takes a special form (e.g. Gaussian) is also mostly irrealistic – this results in a mismatch between the real and the assumed distribution, which we shortly called as the "modelling error"

- In previous lectures we assumed that
  - The exact distribution of the data is known
  - Or at least the shape of the distribution is known
- But in real-life tasks
  - The exact distribution is never known
  - Assuming that its shape takes a special form (e.g. Gaussian) is also mostly irrealistic – this results in a mismatch between the real and the assumed distribution, which we shortly called as the "modelling error"
- Today: methods that do not use a parametric curve (such as a Gaussian), to describe the shape of the distribution
  - This is why they are called "non-parametric methods"

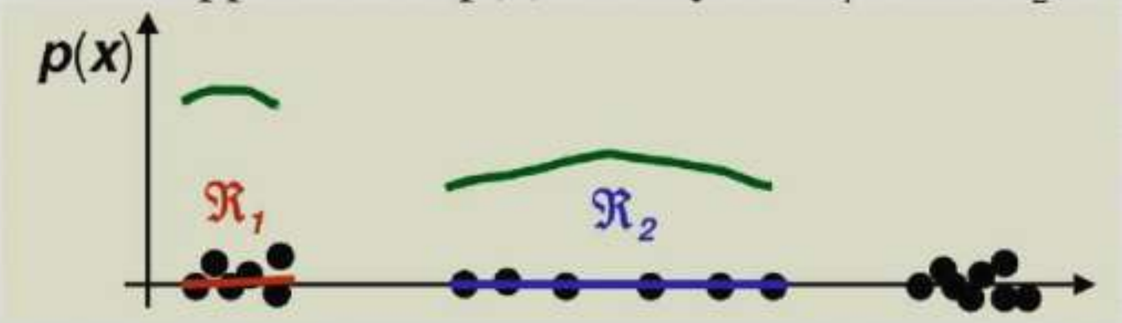
Basic idea

- In a given point we will estimate p(x) from the density of the data points falling into a small region R around x
  - More samples → larger probability



Example

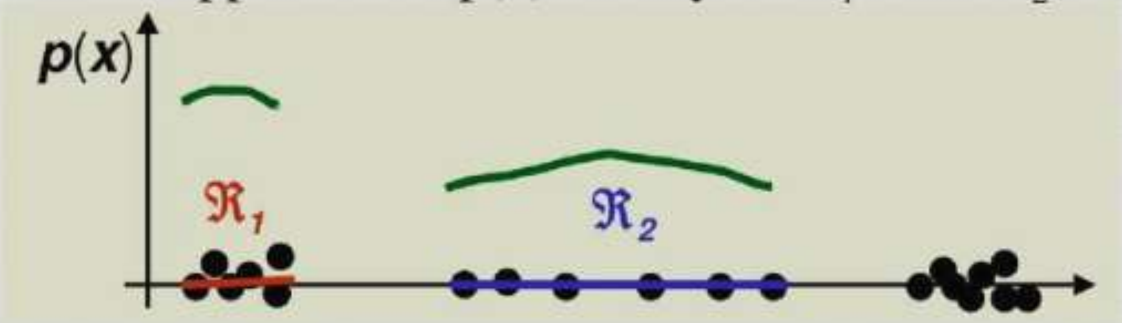
- How can we approximate $p(x)$ for any $x \in R_1$ or $x \in R_2$?



- To estimate $P[x \in R_1]$ and $P[x \in R_2]$, we divide the number k of the samples that fall in the given region R with the total number n of all samples

Example

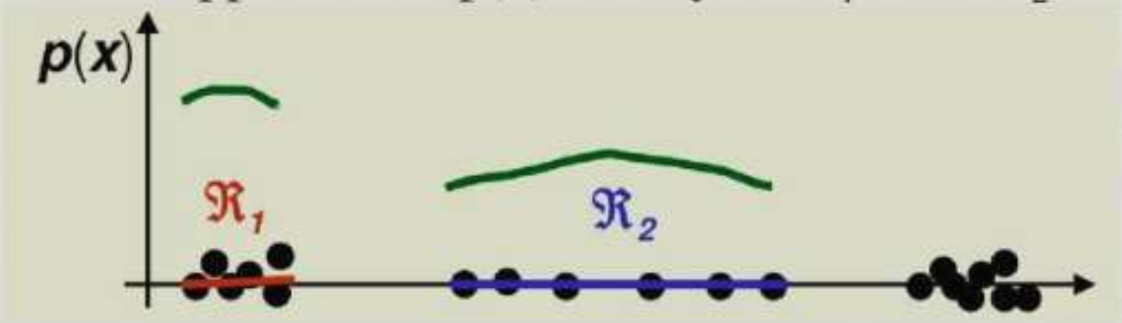
- How can we approximate p(x) for any $x \in R_1$ or $x \in R_2$?



- To estimate $P[x \in R_1]$ and $P[x \in R_2]$, we divide the number k of the samples that fall in the given region R with the total number n of all samples
  - $P[x \in R_1] = 6/20$ $P[x \in R_2] = 6/20$
- Should our estimates for p(x) be equal?

Example

- How can we approximate p(x) for any $x \in R_1$ or $x \in R_2$?



- To estimate $P[x \in R_1]$ and $P[x \in R_2]$, we divide the number k of the samples that fall in the given region R with the total number n of all samples
  - $P[x \in R_1]=6/20$ $P[x \in R_2]=6/20$
- Should our estimates for p(x) be equal?
  - No, because R_2 is wider than R_1
- So the estimate will be inversely proportional to the region size V
  - Altogether, our estimate will be $p(x) \approx k/nV$

Non-parametric density estimation : preliminaries

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

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

Non-parametric density estimation : preliminaries

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

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

- 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 $\mathfrak{R}$ is given by

Non-parametric density estimation : preliminaries

- 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

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

Non-parametric density estimation : preliminaries

- 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

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

Non-parametric density estimation : preliminaries

- 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

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

- 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]

Non-parametric density estimation : preliminaries

- 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 $\Re$

Non-parametric density estimation : preliminaries

- 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

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

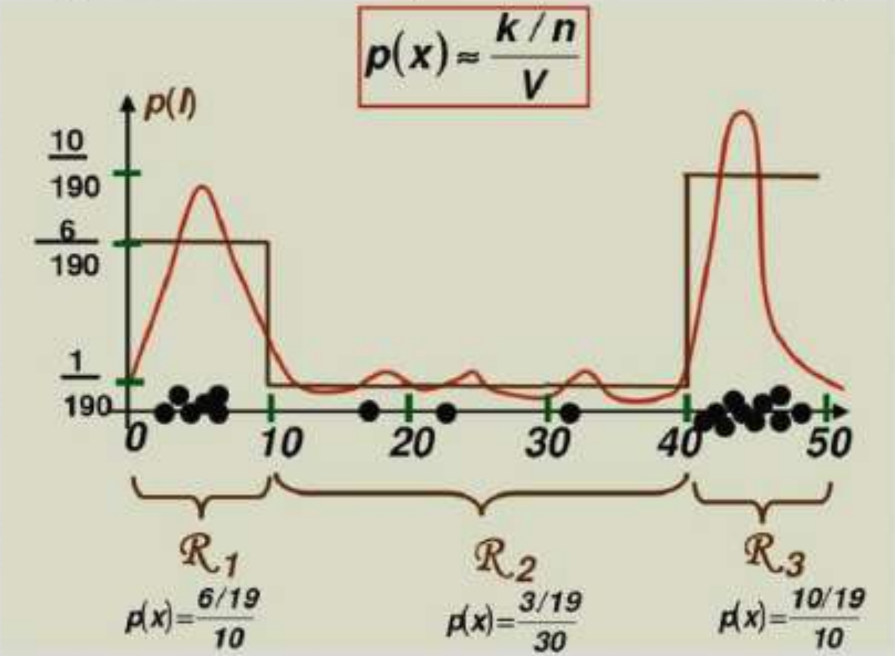
Non-parametric density estimation : preliminaries

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

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

Interpretation as a histogram

- Our probability estimate is very similar to a histogram:



- If the regions do not overlap and cover the whole range of the values then our estimate is basically a (normalized) histogram

The histogram

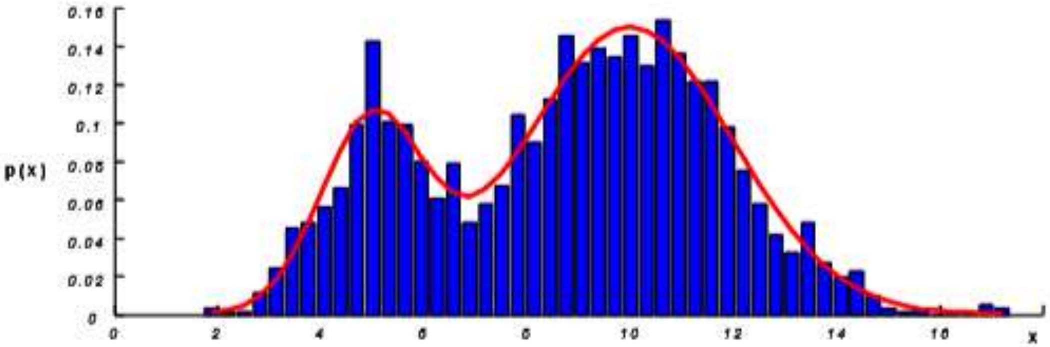
The simplest form of non-parametric DE is the histogram

$p(x) \cong \frac{k}{NV}$

- Divide the sample space into a number of bins and approximate the density at the center of each bin by the fraction of points in the training data that fall into the corresponding bin

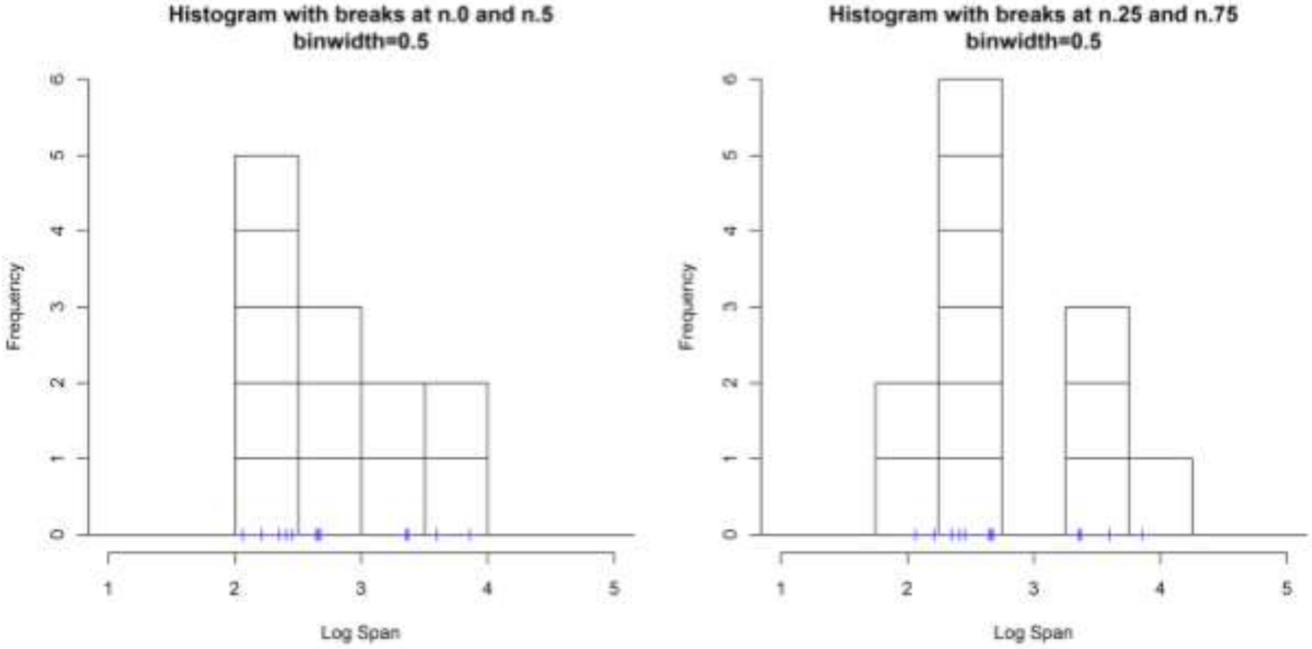
$$p_H(x) = \frac{1}{N} \frac{[\# \; of \; x^{(k} \; in \; same \; bin \; as \; x]}{[width \; of \; bin]}$$

- The histogram requires two "parameters" to be defined: bin width and starting position of the first bin



A toy example

- (the log of) wing spans of aircraft built from 1956 – 1984
- Wing-spans: 2, 22, 42, 62, 82, 102, 122, 142, 162, 182, 202, 222



Issues with Histograms

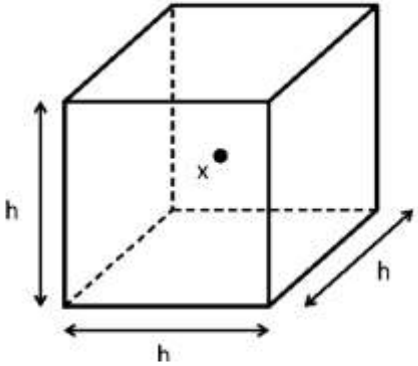
- Not smooth
- Depend on end points of bins
- Depend on width of bins

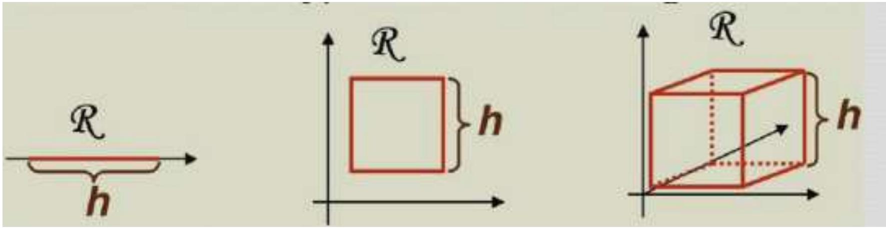
Characterizing density – a data-driven perspective

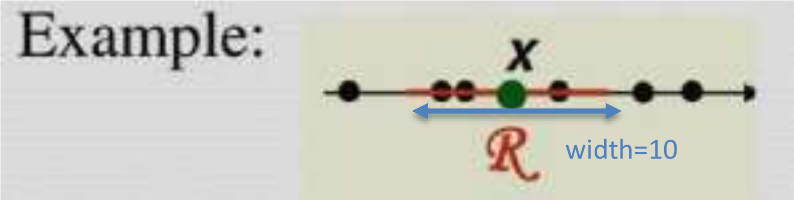
Parzen windows

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





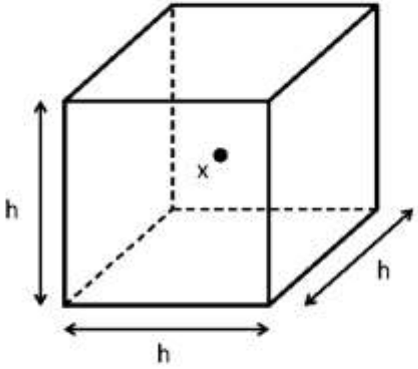


$$p(x) \cong \frac{k}{NV} \text{ where } \begin{cases} V & \textit{volume surrounding } x \\ N & \textit{total \#examples} \\ k & \textit{\#examples inside } V \end{cases}$$

Parzen windows

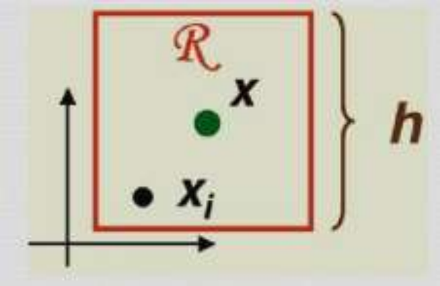
Problem formulation

- 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



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

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



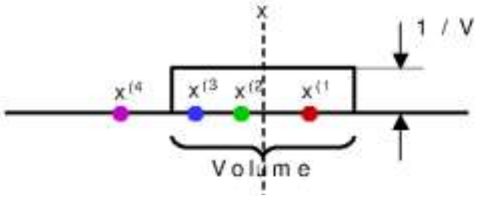
  - This kernel, which corresponds to a unit hypercube centered at the origin, is known as a Parzen window or the naïve estimator
  - The quantity $K((x - x^{(n)})/h)$ is then equal to unity if $x^{(n}$ is inside a hypercube of side h centered on x, and zero otherwise

[Bishop, 1995]

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

- 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

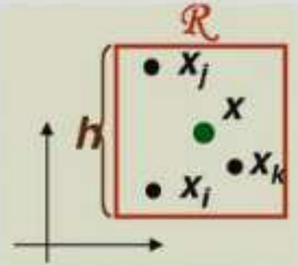
$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(n)}}{h}\right)$$

Still has to be a valid distribution

Parzen window – another intepretation

- So far, we fixed x and varied i to see which of the x_i samples fall within the hypercube centered on x, so that

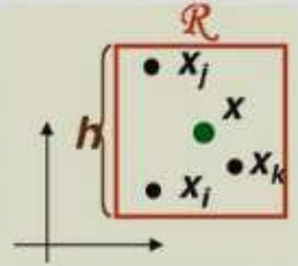
$$\varphi\left(\frac{x - x_i}{h}\right) = 1$$



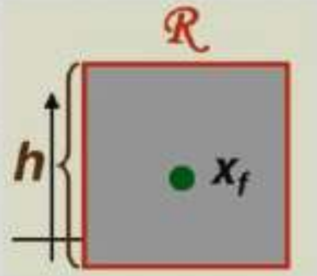
Parzen window – another intepretation

- So far, we fixed x and varied i to see which of the x_i samples fall within the hypercube centered on x, so that

$$\varphi\left(\frac{x - x_i}{h}\right) = 1$$



- Let's turn it around and analyze how a given x_i contributes to the estimate of p(x)
- We see that $\varphi\left(\frac{x - x_f}{h}\right) = 1$ is simply a function that gives 1 for all x values that are close enough to x_f , and 0 otherwise



Parzen window as a sum of functions

- Now, if we look at our estimate again

$$p_\varphi(x) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{h^d}\varphi\left(\frac{x-x_i}{h}\right) = \sum_{i=1}^{i=n}\frac{1}{nh^d}\underbrace{\varphi\left(\frac{x-x_i}{h}\right)}_{\boxed{\begin{array}{l}1 \text{ inside square centered at } x_i \\ 0 \text{ otherwise}\end{array}}}$$

Parzen window as a sum of functions

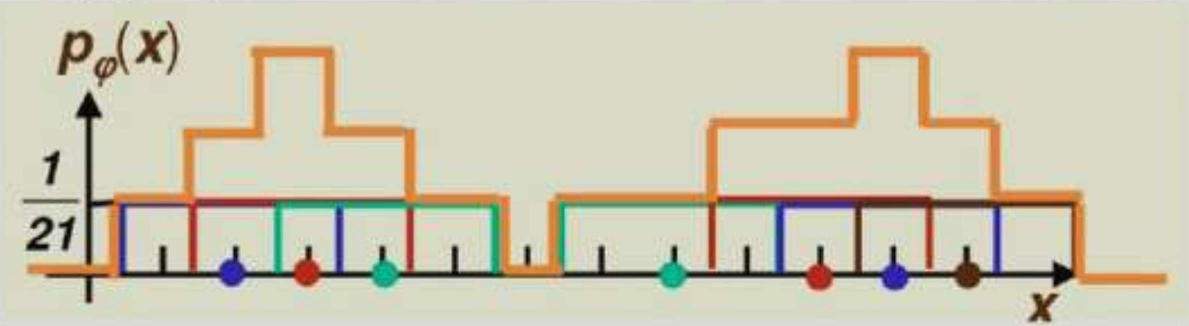
- Now, if we look at our estimate again

$$p_\varphi(x) = \frac{1}{n}\sum_{i=1}^{i=n}\frac{1}{h^d}\varphi\left(\frac{x - x_i}{h}\right) = \sum_{i=1}^{i=n}\frac{1}{nh^d}\underbrace{\varphi\left(\frac{x - x_i}{h}\right)}_{\begin{array}{l}\text{1 inside square centered at } x_i \\ \text{0 otherwise}\end{array}}$$

- We see that we can easily calculate it by fitting hypercubes on all training instances $x_1, \ldots, x_n$
- So p(x) is just a sum of n box-like functions with height $\frac{1}{nh^d}$
- Let's see an example!

Parzen window - example

- We have seven samples D={2,3,4,8,10,11,12}
- n = 7, h = 3, d = 1

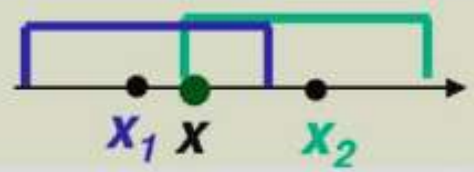


- To obtain our estimate we simply have to sum 7 boxes positioned on the seven points
- The height of the boxes is

$$\frac{1}{nh^d} = \frac{1}{21}$$

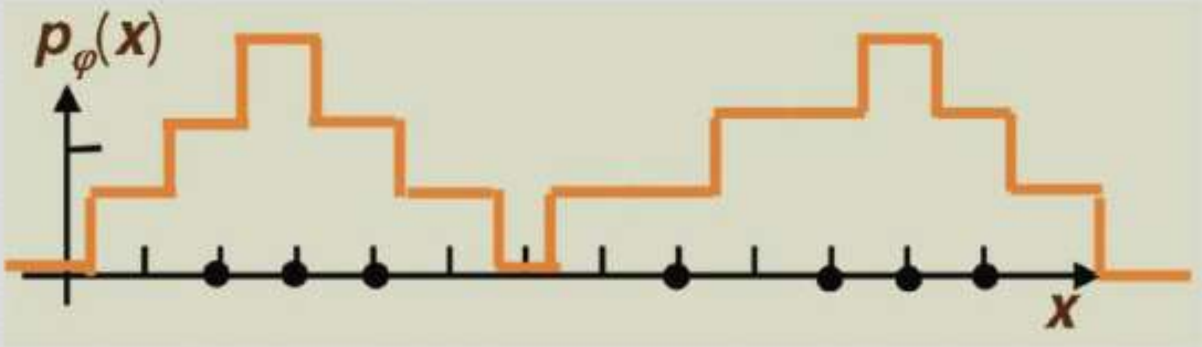
Drawbacks of the hypercube Parzen window

- As long as x_i is within the hypercube around x, its contribution to p(x) will be the same, independent of its distance from x
  - The same is true for the samples outside the hypercube – they give a contribution of 0, no matter how far or close they are to x



$$\varphi\left(\frac{x-x_1}{h}\right)=\varphi\left(\frac{x-x_2}{h}\right)=1$$

- The estimate of p(x) is not smooth



Smooth kernels

The Parzen window has several drawbacks

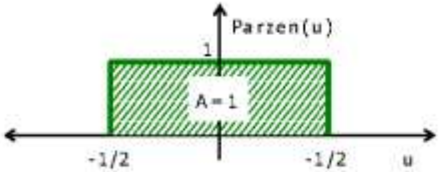
- It yields density estimates that have discontinuities
- It weights equally all points x_i, regardless of their distance to the estimation point x

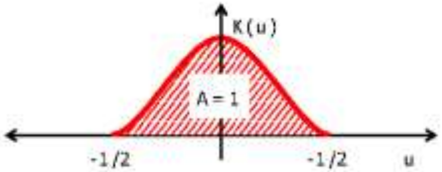
For these reasons, the Parzen window is commonly replaced with a smooth kernel function $K(u)$

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

- 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

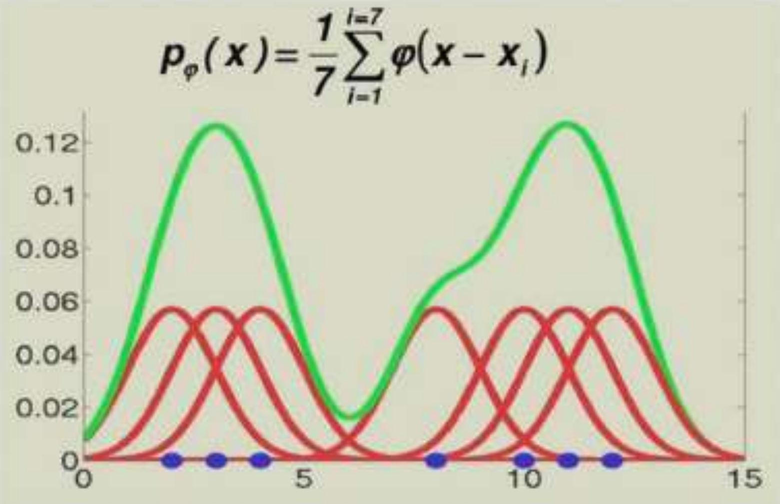
$$p_{KDE}(x) = \frac{1}{Nh^D} \sum_{n=1}^{N} K\left(\frac{x - x^{(k}}{h}\right)$$

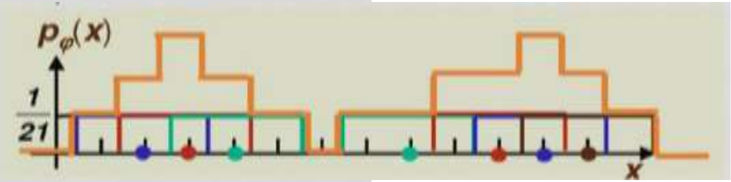




Gaussian window function - example

- Let's return to our previous example
- D={2,3,4,8,10,11,12}, n = 7, h = 1, d = 1

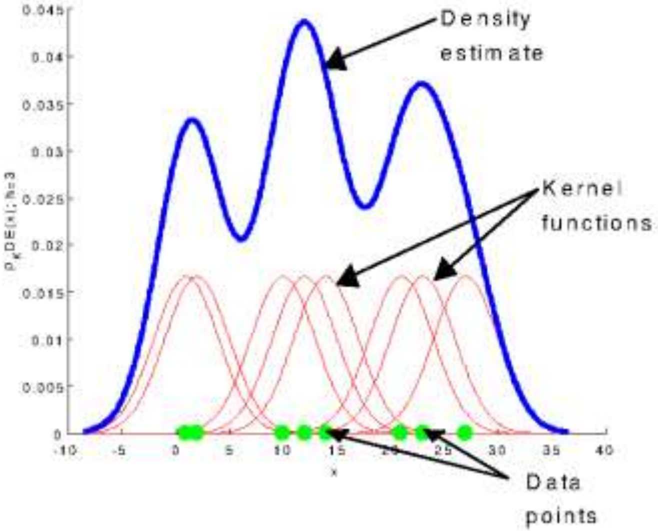




- The estimate for p(x) will be sum of 7 Gaussians, each centerd on one of the sample points, each scaled by 1/7

Interpretation

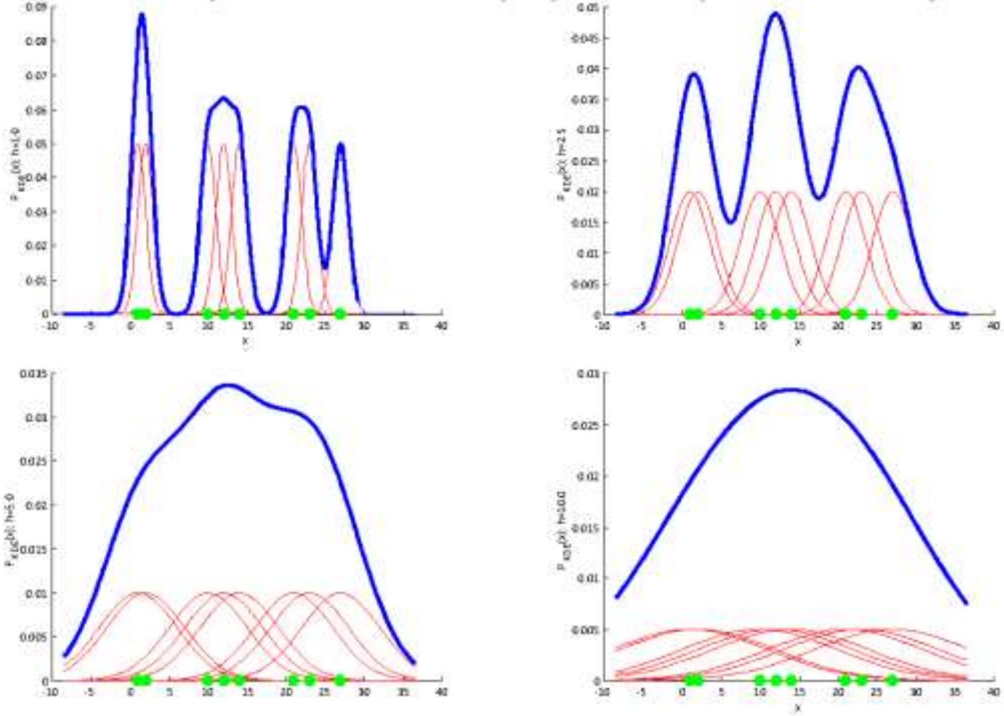
- Just as the Parzen window estimate can be seen as a sum of boxes centered at the data, the smooth kernel estimate is a sum of "bumps"
- The kernel function determines the shape of the bumps
- The parameter h, also called the smoothing parameter or bandwidth, determines their width



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

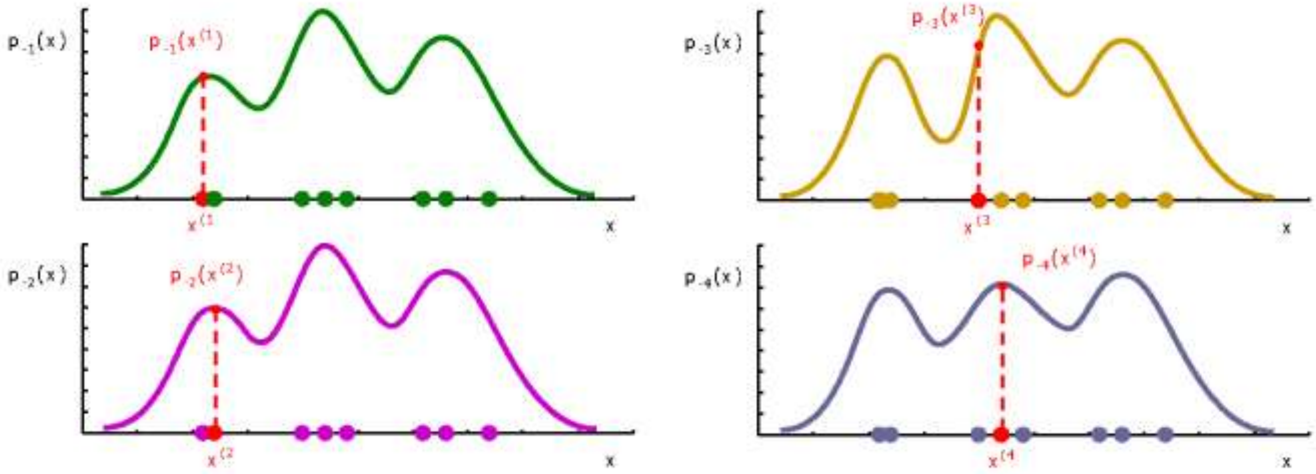
- The ML estimate of h is degenerate since it yields $h_{ML} = 0$, a density estimate with Dirac delta functions at each training data point

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

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

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


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

Product kernels

A good alternative for multivariate KDE is the product kernel

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

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

- 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
  - Bandwidth selection can then be performed with any of the methods presented for univariate density estimation

Product kernels

A good alternative for multivariate KDE is the product kernel

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

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

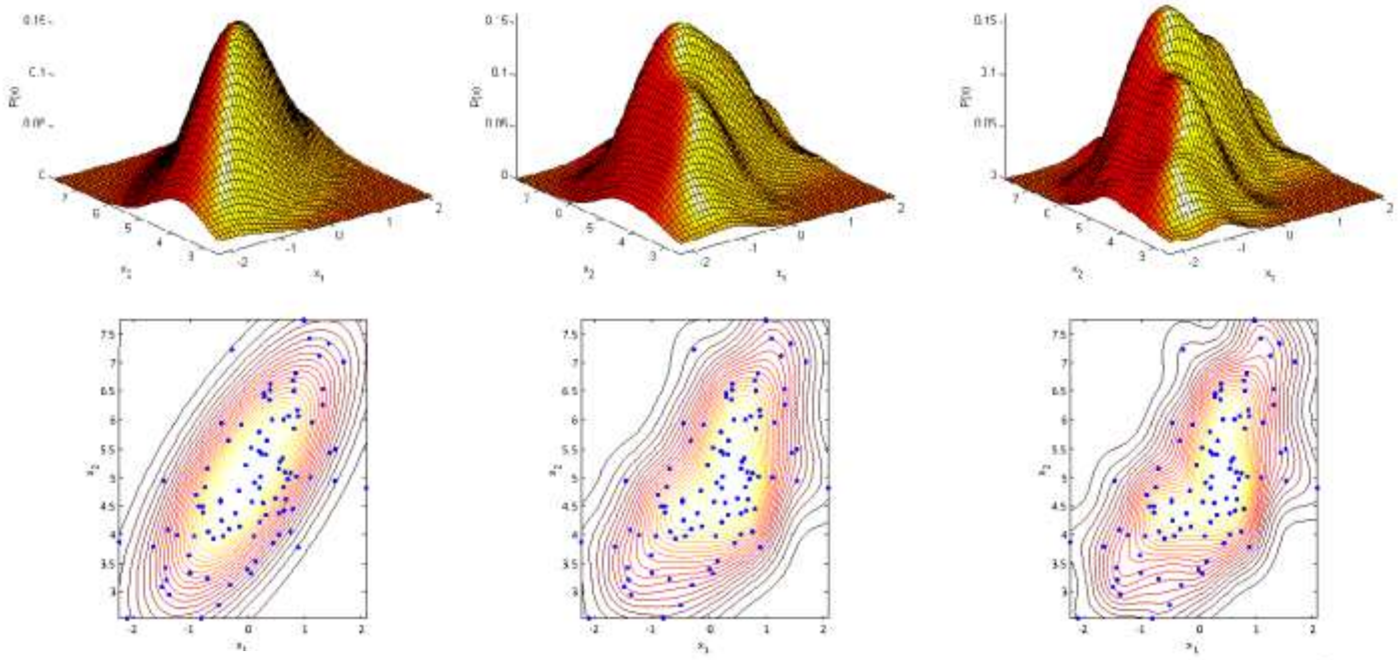
- The product kernel consists of the product of one-dimensional kernels
  - Typically the same kernel function is used in each dimension ($K_d(x) = K(x)$), and only the bandwidths are allowed to differ
  - Bandwidth selection can then be performed with any of the methods presented for univariate density estimation
- Note that although $K\left(x, x^{(n}, h_1, \dots h_D\right)$ 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

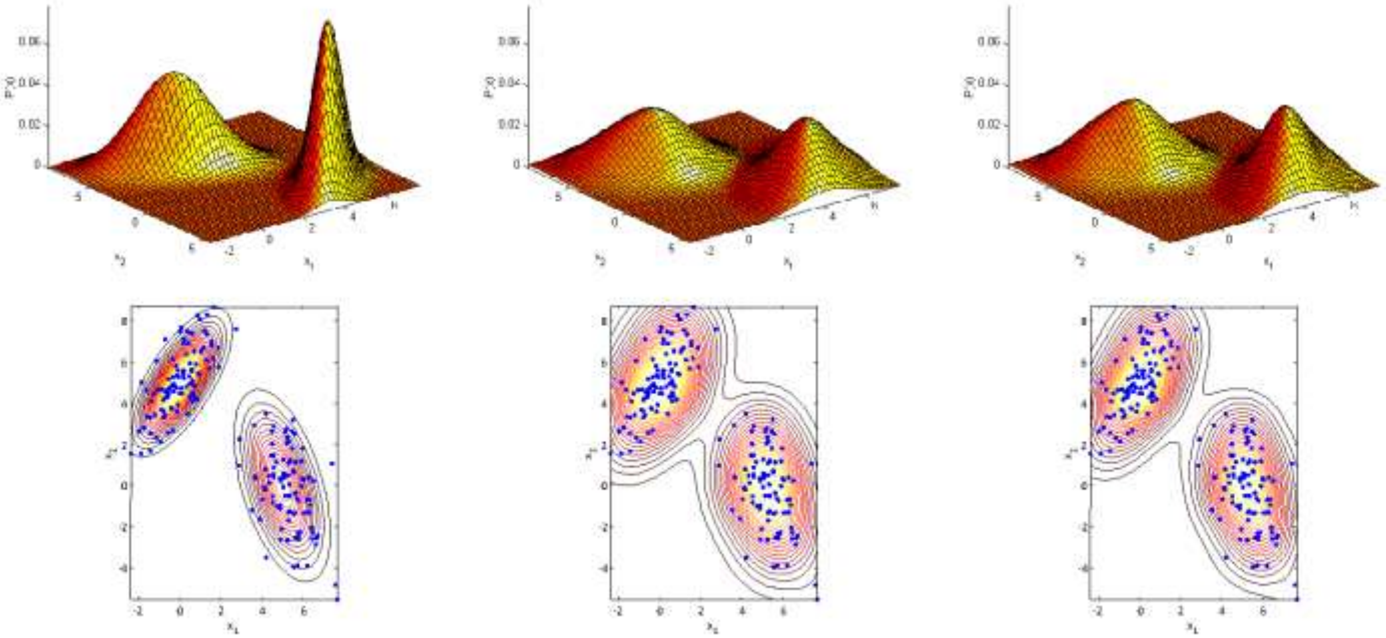
Example I

- This example shows the product KDE of a bivariate <u>unimodal</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)



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])
```

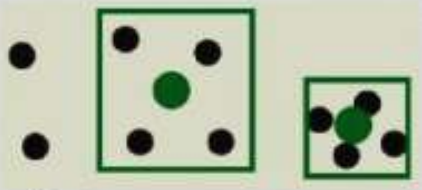
Connection between KDE and k-NN

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

- 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

Using k-NN for density estimation

- The k-NN approach seems to be a good solution for the "optimal window size" problem
  - Center a cell on x and let it grow until it captures k samples
  - These k samples will be the k nearest neighbors of x

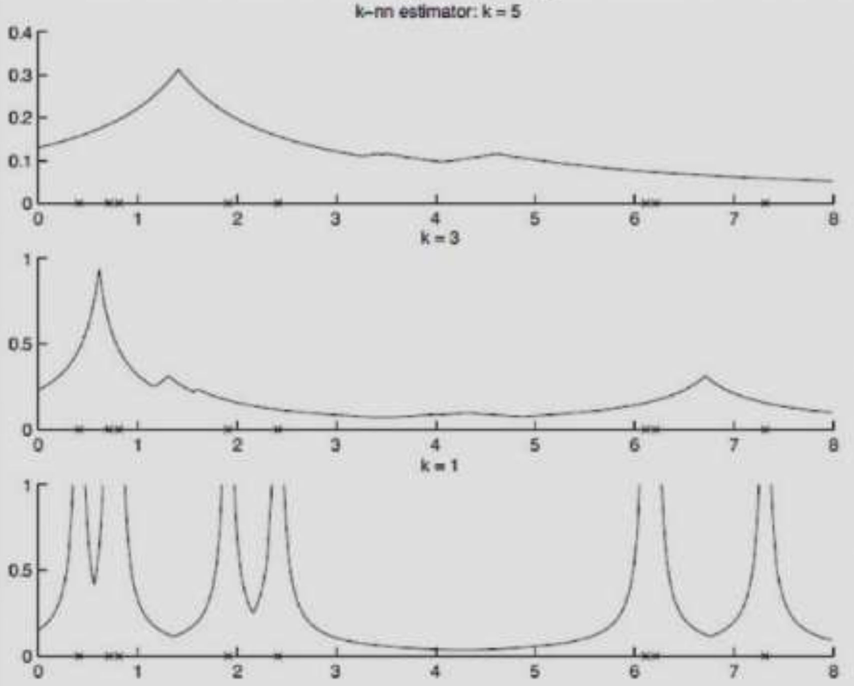


- The window size will change dynamically
  - If the samples are locally dense, then V will be small, and we obtain a more precise estimate
  - If the samples are sparse, then V is larger and the estimate is smoother

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

Using k-NN for density estimation

- For a larger k the estimate is better, but still not a valid distribution
- For small k the estimates are very "spiky"



References

- Duda-Hart: 4.1 – 4.6
- Bishop (PRML) : 2.5