Machine learning

Non-parametric Methods I Parzen

Mohammad-Reza A. Dehaqani

dehaqani@ut.ac.ir

Introduction

- Density estimation with parametric models assumes that **the forms of the underlying density** functions are known.
- However, **common parametric** forms do not always fit the densities actually **encountered in practice**.
  - In addition, most of the classical parametric densities are **unimodal**, whereas many practical problems involve **multimodal** densities.
- Non-parametric methods can be used with **arbitrary distributions** and **without the assumption** that the forms of the underlying densities are known.

Non-parametric Density Estimation

- Suppose that n samples $x_1, \ldots, x_n$ are drawn **i.i.d.** according to the distribution p(x).
- The probability P that a vector x will fall in a region R is given by

$$P = \int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}'.$$

- The probability that k of the n will **fall in R** is given by the binomial law

$$P_k = \binom{n}{k} P^k (1-P)^{n-k}.$$

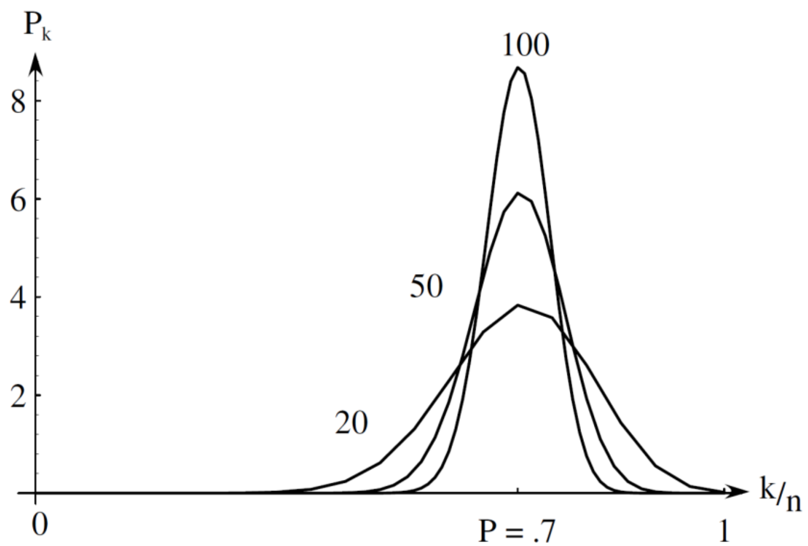
- The expected value of k is $\mathbf{E[k] = nP}$ and the MLE for P is

$$\hat{P} = \frac{k}{n}.$$

Can use k/n as an estimate of P

$$E\left\{\left(\frac{k}{n}-P\right)^2\right\}=\frac{1}{n^2}E\left\{(k-nP)^2\right\}=\frac{P(1-P)}{n}$$

- Estimate peaks sharply around the mean as the number of samples increases ($n \to \infty$)



Non-parametric Density Estimation

- If we assume that p(x) is **continuous** and R is **small** enough so that p(x) **does not vary** significantly in it, we can get the approximation

$$\int_{\mathcal{R}} p(\mathbf{x}')d\mathbf{x}' \simeq p(\mathbf{x})V$$

where x is a point in R and V is the **volume** of R.

- Then, the density estimate becomes

$$\boxed{p(\mathbf{x}) \simeq \frac{k/n}{V}.}$$

Several problems that remain

- If **we fix the volume *V*** and take more and more training samples, the ratio $\frac{k}{n}$ will converge (in probability) as desired, but we have only obtained an estimate of the **space-averaged value of** $p(x)$
- We must be prepared to **let *V* approach zero**.
- If we **fix the number *n*** of samples and let ***V* approach zero**, the region will eventually become **so small** that it will **enclose no samples**, and our estimate $p(\mathbf{x}) \approx 0$ will be useless
  - Or if by chance one or more of the training samples **coincide at x**, the **estimate diverges to infinity**, which is equally useless

Conditions for convergence

- Let n be the **number of samples** used, R_n be the region used with n samples, $\mathbf{V_n}$ be the volume of R_n, k_n be the number of samples falling in R_n, and the estimate for p(x) be

$$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$

- If $p_n(x)$ is to **converge** to p(x), three **conditions** are required:

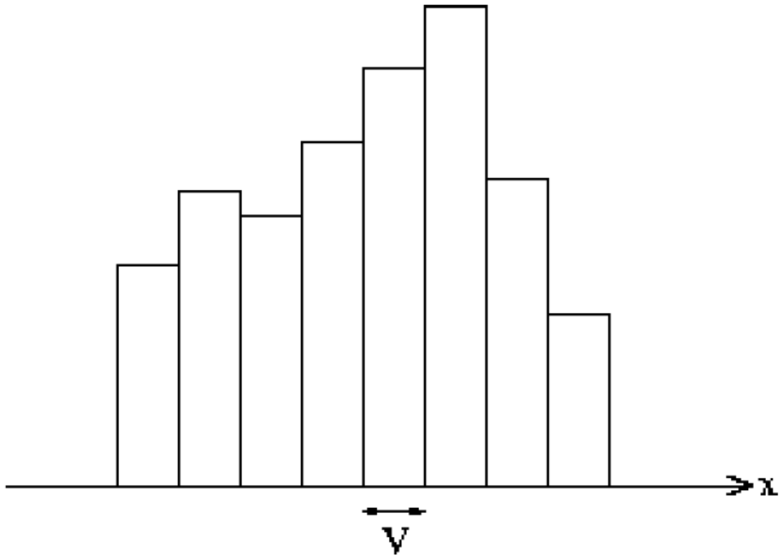
$$\lim_{n\to\infty} V_n = 0$$

$$\lim_{n\to\infty} k_n = \infty$$

$$\lim_{n\to\infty} \frac{k_n}{n} = 0.$$

Histogram Method

A very simple method is to **partition** the space into a number of **equally-sized cells** (**bins**) and compute a histogram.



- The estimate of the density at a point x becomes

$$p(\mathbf{x}) = \frac{k}{nV}$$

where n is the total **number of samples**, k is the number of samples in the cell that includes x, and V is the **volume of that cell**.

Histogram Method

- The number of bins M (or bin size) is acting as a **smoothing parameter**.
  - If bin width is **small** (big M), then the estimated density is very **spiky** (i.e., noisy).
  - If bin width is **large** (small M), then the true structure of the density is **smoothed** out.
- Although the histogram method is **very easy** to implement, it is usually not practical in high-dimensional spaces **due to the number of cells**.
- **Many observations are required** to prevent the estimate being zero over a large region.

Methods for obtaining the regions for estimation

- Two common ways of obtaining sequences of regions that satisfy three conditions

$$\lim_{n\to\infty} V_n = 0$$

$$\lim_{n\to\infty} k_n = \infty$$

$$\lim_{n\to\infty} \frac{k_n}{n} = 0.$$

- **Parzen window** estimation

  Shrink regions as some function of n, **such as**

$$V_n = 1/\sqrt{n}.$$

- **k-nearest neighbor** estimation

$$\boxed{p(\mathbf{x}) \simeq \frac{k/n}{V}.}$$

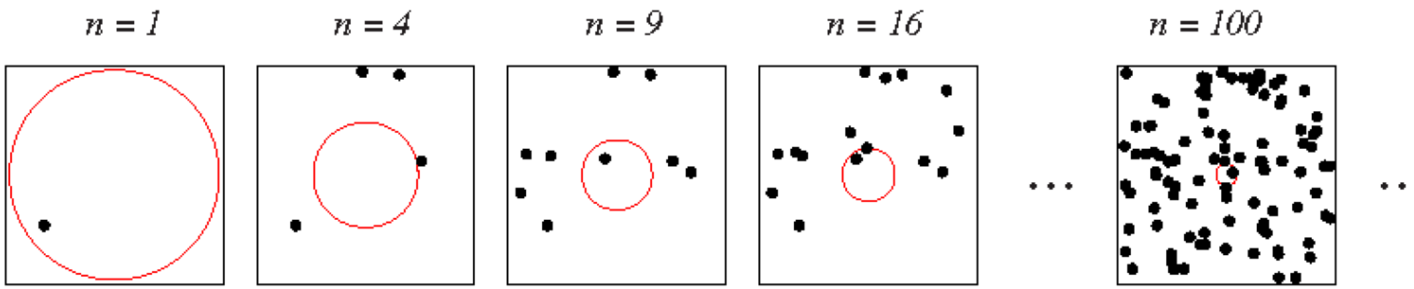
  Specify k_n as some function of n, **such as**

$$k_n = \sqrt{n}.$$

Two methods for estimating the density at a point $\mathbf{x}$ (at the center of each square)

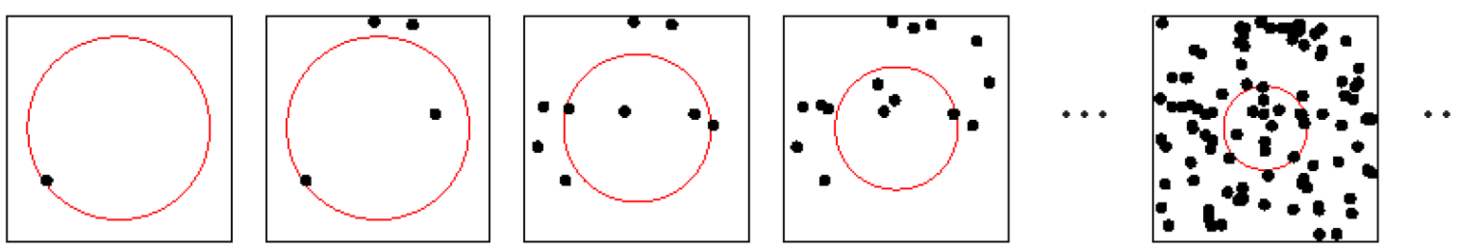
- **Parzen window**

$V_n = 1/\sqrt{n}$



- **k-nearest neighbor**

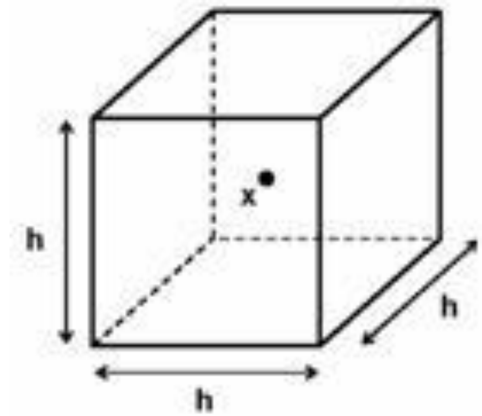
$k_n = \sqrt{n}$



Unit hypercube centered at the origin kernel function

- The region R_n is a d-dimensional hypercube which encloses k samples.

$$V_n = h_n^d.$$

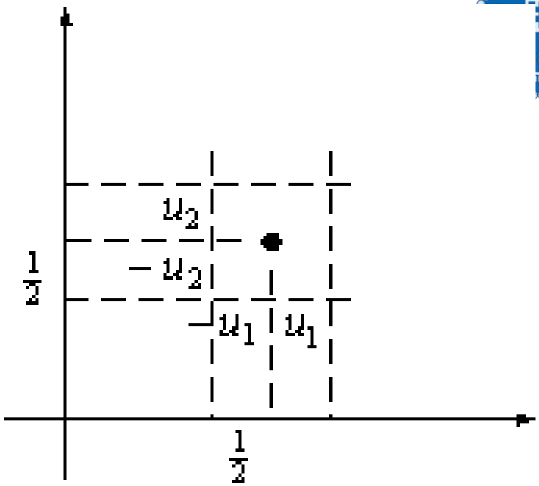


- We can obtain an **analytic expression for** k_n. The number of samples falling in the window hypercube, by defining the following *window function* (***kernel function***)

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq 1/2 \qquad j = 1, \ldots, d \\ 0 & \text{otherwise.} \end{cases}$$

Parzen Windows

$\varphi((\mathbf{x} - \mathbf{x}_i)/ h_n)$ is equal to unity if $\mathbf{x}_i$ falls within the hypercube of volume V_n centered at $\mathbf{x}$, and is zero otherwise.



The kernel function in two dimensions

- The **number of samples** in this hypercube is therefore given by

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right),$$

- Substitute this into last equation we obtain the **estimate**

$$p_n(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right).$$

Parzen Windows; more general class of window functions.

- Rather than limiting ourselves to the hypercube window function
- *p*(**x**) as an **average** of functions of **x** and the samples **x***i*
- The window function is being used for ***interpolation*** — each sample **contributing** to the estimate in accordance **with its distance from x**.
- Suppose that φ is a d-dimensional window function that satisfies the **properties of a density function**, i.e.,

$$\varphi(\mathbf{u}) \geq 0 \quad \text{and} \quad \int \varphi(\mathbf{u})d\mathbf{u} = 1.$$

- A **density estimate** can be obtained as

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

- where h_n is the **window width** and $V_n = h_n^d$. (*d*-dimensional ***hypercube***.)

Parzen Windows

- The density estimate can also be written as

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \delta_n(\mathbf{x} - \mathbf{x}_{\boldsymbol{i}})$$

where $\delta_n(\mathbf{x}) = \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}}{h_n}\right)$

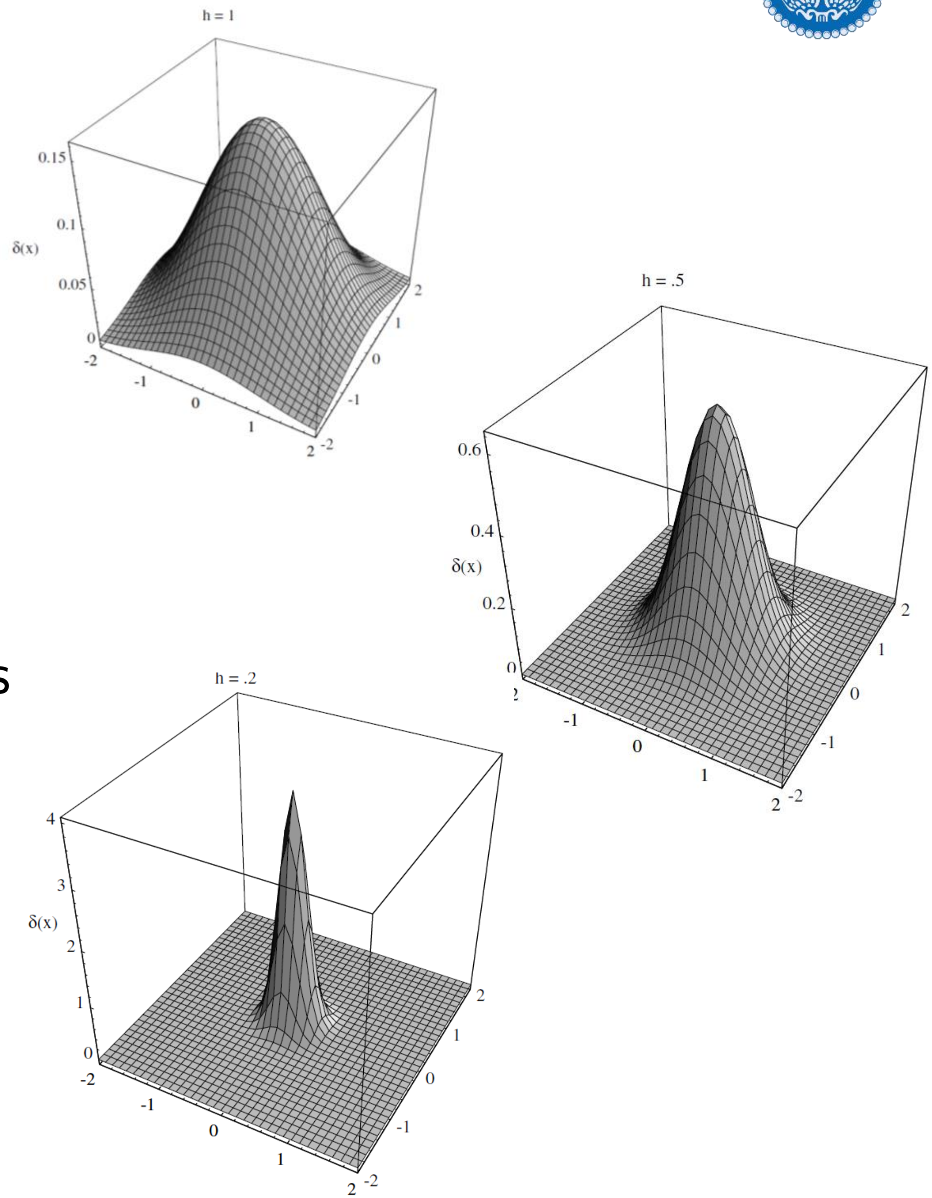
- For any value of h_n, the distribution is **normalized**

$$\int \delta_n(\mathbf{x} - \mathbf{x}_i)\, d\mathbf{x} = \int \frac{1}{V_n}\varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right)\, d\mathbf{x} = \int \varphi(\mathbf{u})\, d\mathbf{u} = 1.$$

- The parameter h_n acts as a ***smoothing*** parameter that needs to be optimized

The role of h_n

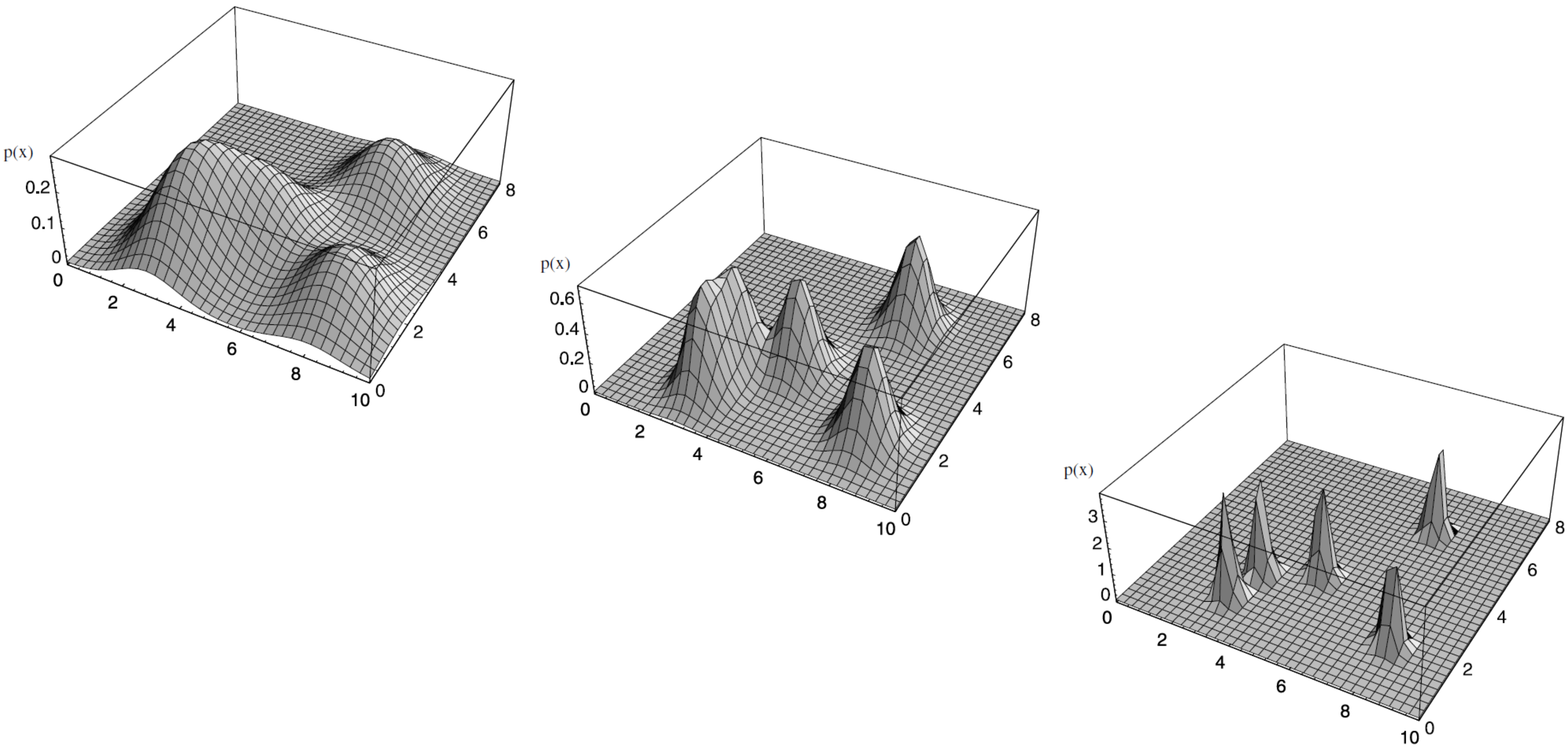
- If h_n is very **large**, $p_n(x)$ is the superposition of n **broad functions**, and is a smooth **"out-of-focus"** estimate of $p(x)$.
- If h_n is very **small**, $p_n(x)$ is the superposition of n **sharp pulses centered at the samples**, and is a "**noisy**" estimate of $p(x)$.
- As h_n approaches zero, $\delta_n(x - x_i)$ approaches a **Dirac delta** function centered at x_i, and $p_n(x)$ is a superposition of delta functions.



$\varphi(u)$ as a **function** of h_n

$p_n(x)$ as a function of h_n

Parzen window density estimates based on the same set of five samples using the window functions in the previous figure



Discussing convergence

- We are talking about the **convergence of a sequence of random variables**, since for any fixed $\mathbf{x}$ the value of $p_n(\mathbf{x})$ depends on the random samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$.
- The estimate $p_n(\mathbf{x})$ converges to $p(\mathbf{x})$ if:

$$\lim_{n \to \infty} \bar{p}_n(\mathbf{x}) = p(\mathbf{x}) \quad \text{and} \quad \lim_{n \to \infty} \sigma_n^2(\mathbf{x}) = 0.$$

- Following additional conditions assure convergence

$$\sup_{\mathbf{u}} \varphi(\mathbf{u}) < \infty$$

$$\lim_{\|\mathbf{u}\| \to \infty} \varphi(\mathbf{u}) \prod_{i=1}^{d} u_i = 0$$

$$\lim_{n \to \infty} V_n = 0$$

$$\lim_{n \to \infty} nV_n = \infty.$$

V_n must approach zero, but at a rate slower than $1/\sqrt{n}$

Convergence of the Mean

- The samples $\mathbf{x}_i$ are **i.i.d**. according to the (unknown) density $p(\mathbf{x})$, we have

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)$$

$$\begin{aligned}\bar{p}_n(\mathbf{x}) &= E\left[p_n(\mathbf{x})\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} E\Big[\frac{1}{V_n}\varphi\Big(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\Big)\Big] \\ \mathbf{x}_i \text{ are } \textbf{i.i.d} \Rightarrow \quad &= \int \frac{1}{V_n}\varphi\Big(\frac{\mathbf{x}-y}{h_n}\Big)\, p(y)\, dy \\ &= \int \delta_n(\mathbf{x}-y)p(y)\, dy.\end{aligned}$$

- A ***convolution*** of the unknown **density** and the **window function** ($\bar{p}_n(\mathbf{x})$ is a blurred version of $p(\mathbf{x})$)
- *As* V_n approaches zero, $\delta_n(\mathbf{x}-\mathbf{y})$ approaches a **delta function** centered at **x. So** $\bar{p}_n(\mathbf{x})$ will approach $p(\mathbf{x})$ as n approaches infinity

Convergence of the Variance

- $p_n(\mathbf{x})$ is the **sum of** functions of statistically **independent random variables**, its variance is the sum of the variances of the separate terms

$$\boxed{p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)}$$

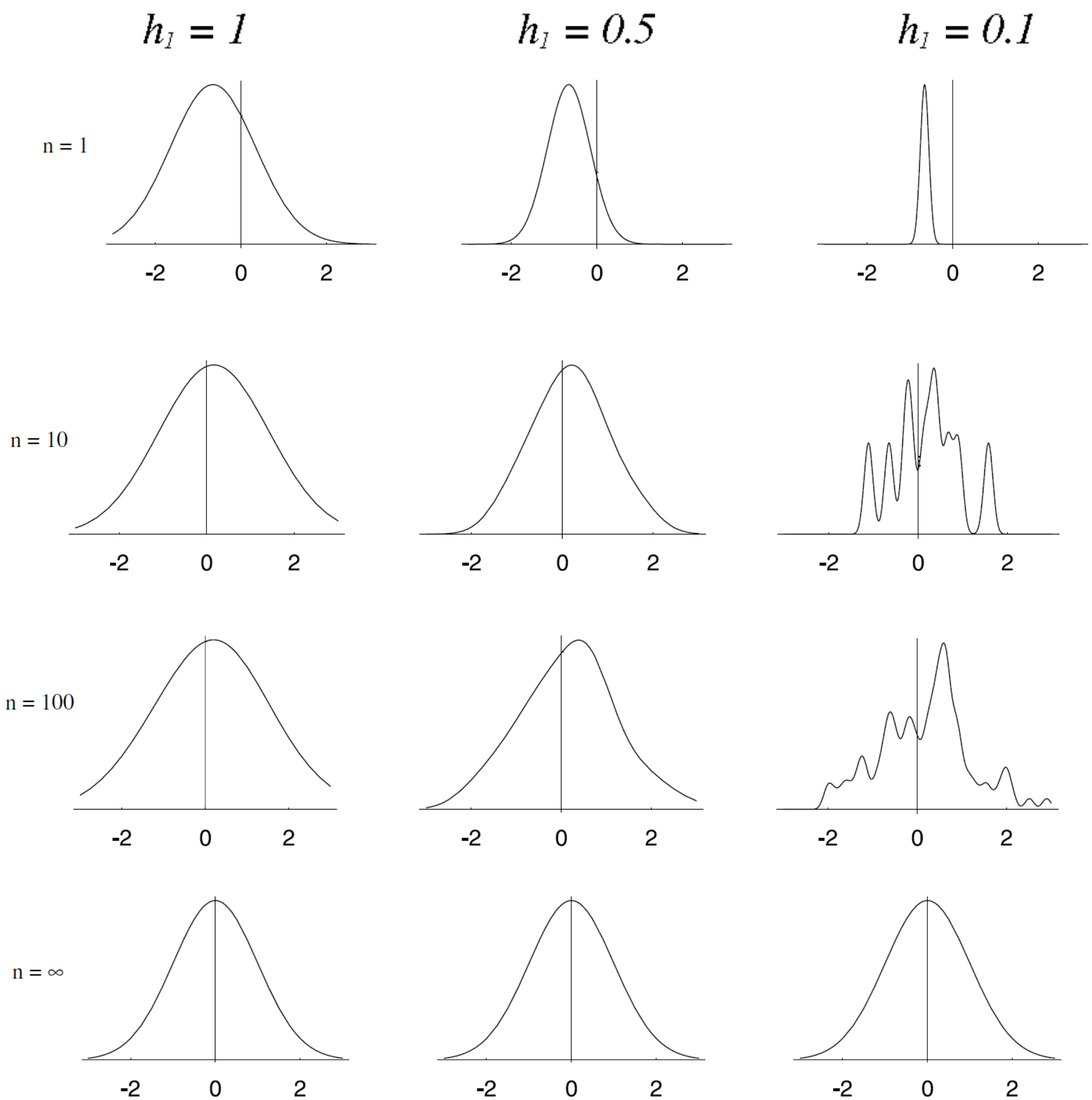
$var(Pn(x))$

$$\begin{aligned}
\sigma_n^2(\mathbf{x}) &= \sum_{i=1}^{n}\left[E\left[\left(\frac{1}{nV_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right)^2\right]-\left(\bar{p}_n(\mathbf{x})\right)^2\right] \\
&= n\,E\left[\frac{1}{n^2V_n^2}\varphi^2\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right] - n\,\bar{p}_n^2(\mathbf{x}) \\
&= \frac{1}{nV_n}\int\frac{1}{V_n}\varphi^2\left(\frac{\mathbf{x}-y}{h_n}\right)p(y)\,dy - n\,\bar{p}_n^2(\mathbf{x}).
\end{aligned}$$

$$\boxed{\sigma_n^2(\mathbf{x}) \le \frac{\sup(\varphi(\cdot))\;\bar{p}_n(\mathbf{x})}{nV_n}}.$$

- *So* nV_n approaches infinity. We can let $V_n = V_1/\sqrt{n}$ or $V_1/\ln n$ or any other equation that satisfy our conditions

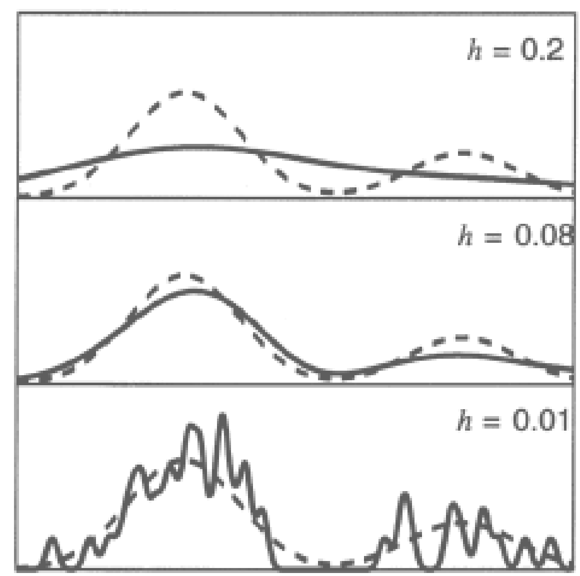
both p(x) and φ(u) are Gaussian



$p_n(x)$ is **an average of normal** densities centered at the samples

$$\varphi(u) = \frac{1}{\sqrt{2\pi}}\, e^{-u^2/2}.$$

$$h_n = h_1/\sqrt{n},$$

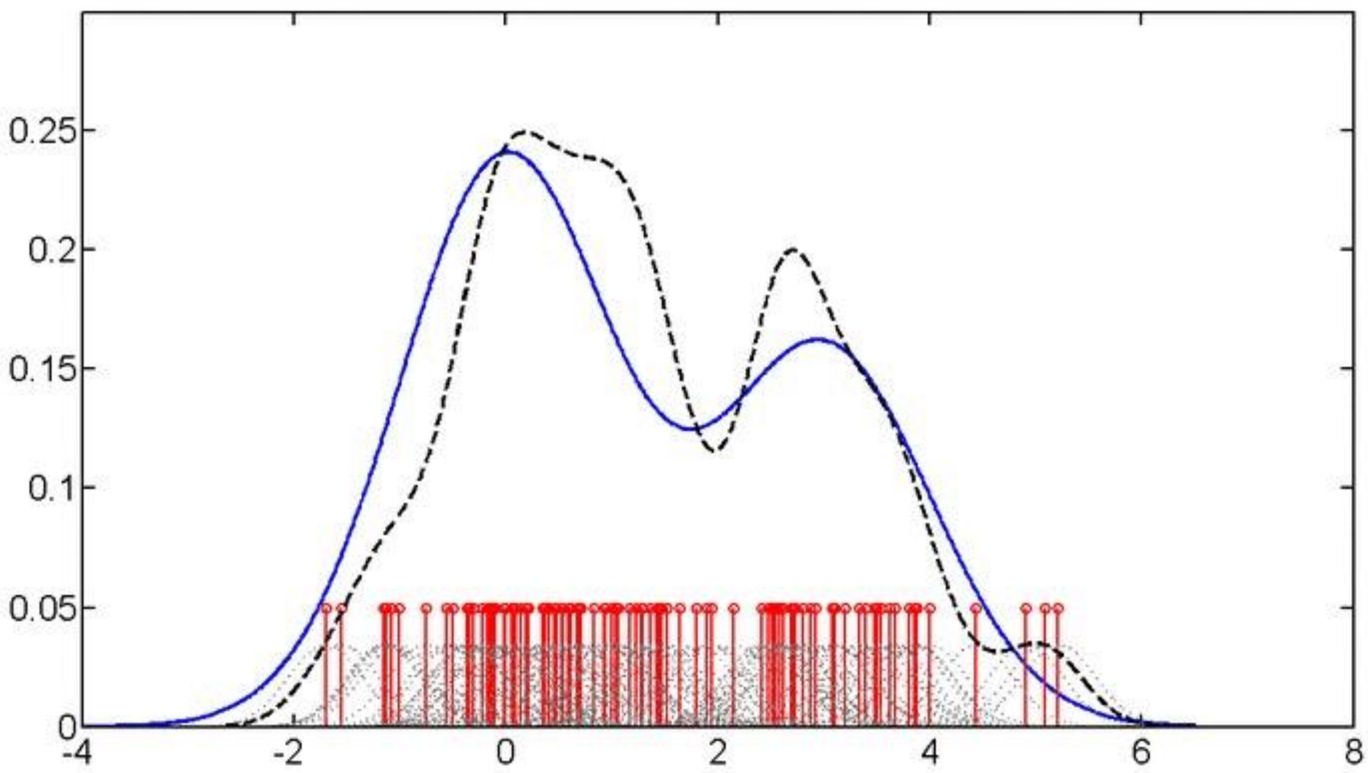


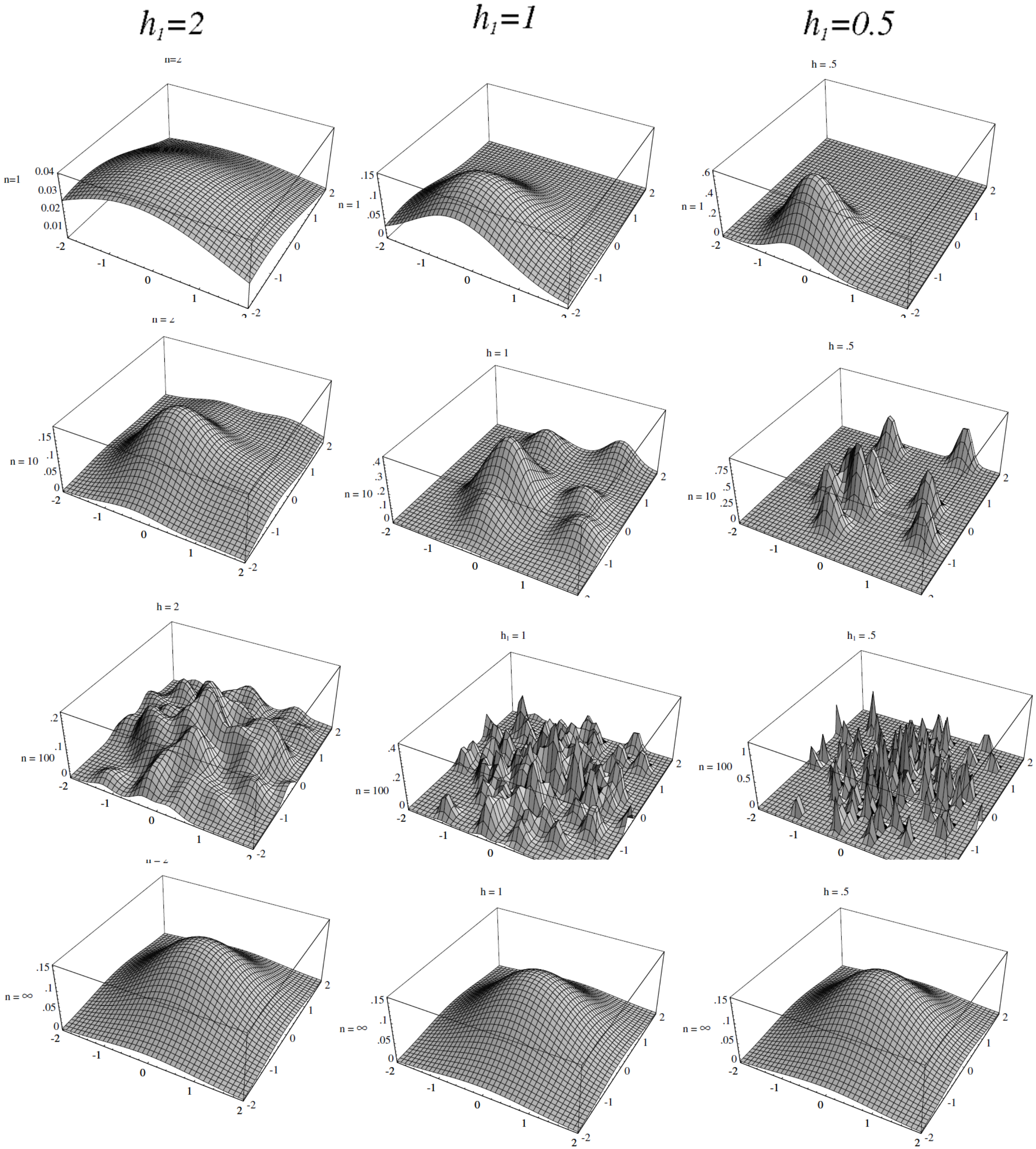
Parzen window

Blue curve: true density is mixture of two Gaussians centered around 0 and 3

In each frame, 100 samples are generated from the distribution, shown in **red**

Dashed black curve: averaging the Gaussians yields the density estimate



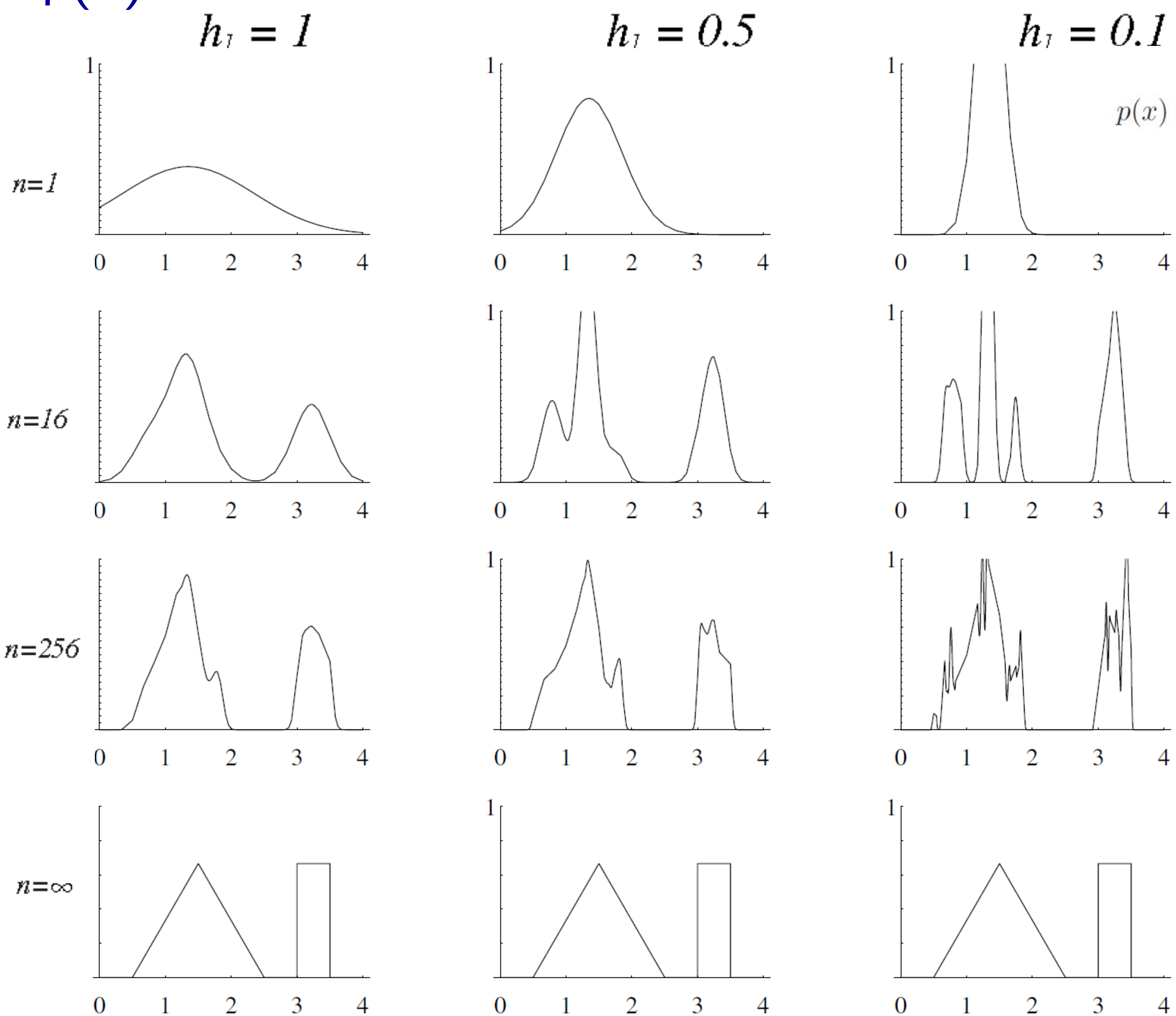


Parzen-window estimates of a **bivariate** normal

$$\varphi(\mathbf{u}) = N(\mathbf{0}, \mathbf{I})$$

$$h_n = h_1/\sqrt{n}.$$

$p(\mathbf{x})$ consists of a uniform and triangular density and φ(x) is Gaussian



$$p(x) = \begin{cases} 1 & -2.5 < x < -2 \\ 1/4 & 0 < x < 2 \\ 0 & \text{otherwise.} \end{cases}$$

$$\varphi(u) = \frac{1}{\sqrt{2\pi}}\, e^{-u^2/2}$$

$$h_n = h_1/\sqrt{n},$$

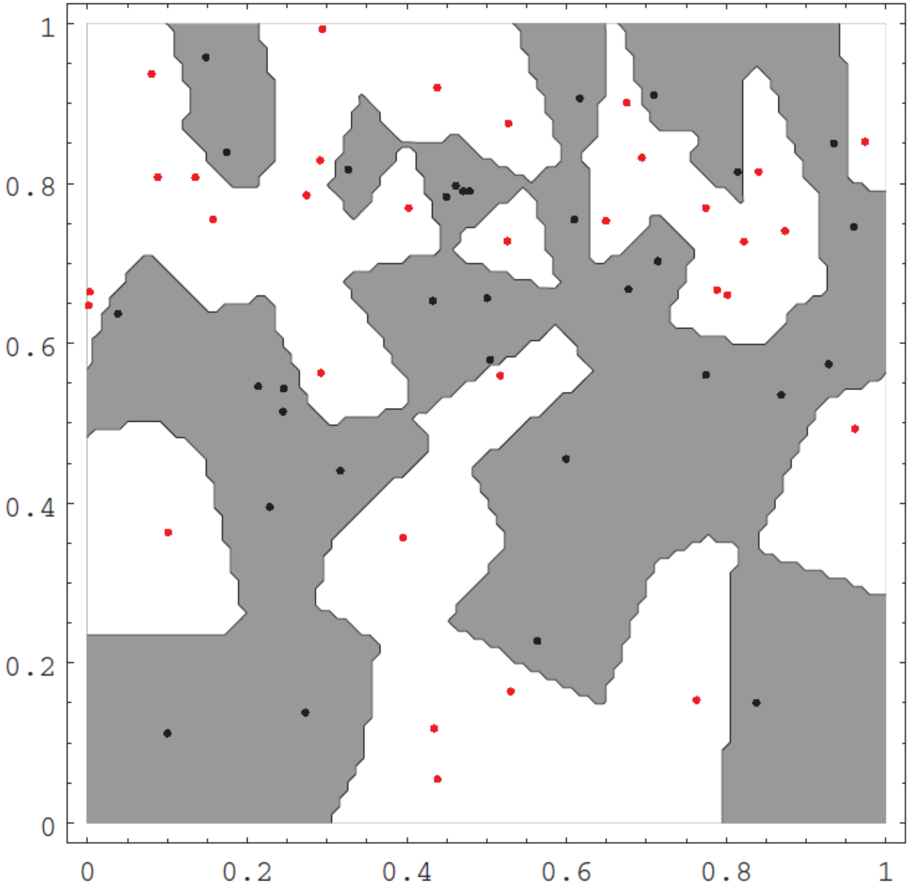
Classification using kernel-based density estimation (Bayesian decision rule)

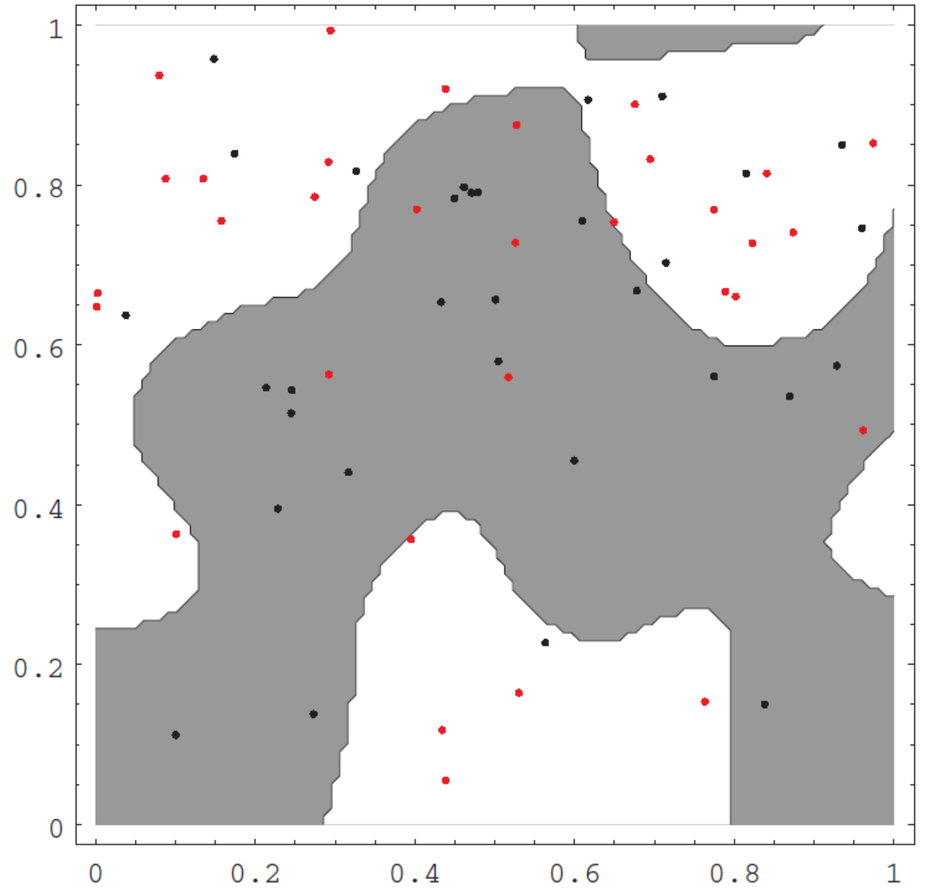
- Estimate density for **each class**.
- **Classify** a test point by computing the **posterior probabilities** and **picking the max**.
- The **decision regions** depend on the choice of the **kernel function** and $\boldsymbol{h_n}$.
- The training error can be made **arbitrarily low** by making the window width **sufficiently small**.
- However, the **goal is to classify novel** patterns so the window width **cannot be made** too small.

dimensional Parzen-window dichotomizer

- small h_n
- Boundaries are more complicated

- large h_n
- No single window width is ideal overall





Drawbacks of kernel-based methods

- Require a large number of samples.
- Require all the samples to **be stored**.
- Evaluation of the density could be **very slow** if the number of data points is large.
- Possible **solution**:
  - use **fewer kernels** and **adapt the positions** and widths in response to the data (e.g., mixtures of Gaussians!)