Non parametric Techniques

- Parzen Windows
- Nearest Neighbor classifier

Non parametric generative classifiers

- Generative models assume data to come from a probability density function.

- Parametric learning assumes we know the form of the underlying density function, which is often not true in real applications.

- All parametric densities are either unimodal (have a single local maximum), such as a Gaussian distribution, or multi-modal (example: GMMs)

- Nonparametric procedures can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known.

- They are data-driven (or are estimated from the data).

- There are two types of nonparametric methods:
  - Estimating $p(\mathbf{x}|\omega_j)$ -→ Parzen Window
  - Bypass class conditional probability estimation and go directly to *a-posteriori* probability estimation, $P(\omega_j|\mathbf{x})$ -→ Nearest neighbor

Basic Idea

- The basic idea in density estimation is that a vector, **x**, will fall in a region R with probability: .

$$P = \int_R p(\mathbf{x}')d\mathbf{x}'$$

- P is a smoothed or averaged version of the density function p(**x**).

- Suppose *n* samples are drawn independently and identically distributed (i.i.d.) according to p(**x**). The probability that *k* of these *n* fall in *R* is given by:

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

The expected value for k is: E[k] = nP.

- The ML estimate, $\max_{\theta}(P_k \mid \theta)$, is $\hat{\theta} = \frac{k}{n} \cong P$
- Therefore, with large number of samples, the ratio k/n is a good estimate for the probability P and hence for the density function p(**x**).

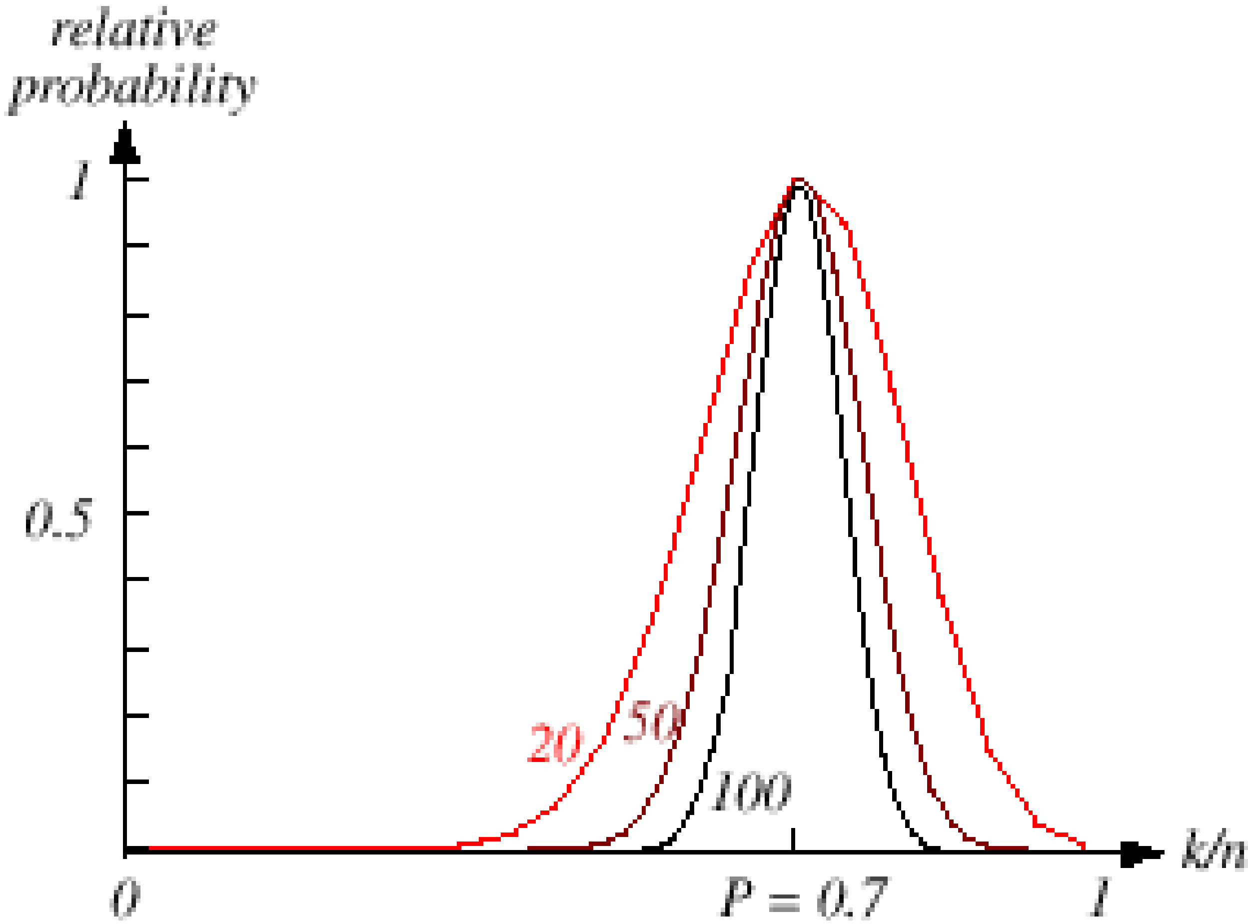
- Assume p(**x**) is continuous and that the region *R* is so small that p(**x**) does not vary significantly within it. We can write:

$$\int_R p(\mathbf{x}')d\mathbf{x}' \cong p(\mathbf{x})V$$

where **x** is a point within *R* and V the volume enclosed by *R*, and .

$$p(\mathbf{x}) \cong \frac{k/n}{V}$$

–



- **A demonstration of nonparametric density estimation. The true probability was chosen to be 0.7. The curves vary as a function of the number of samples,** n**. We see the binomial distribution peaks strongly at the true probability.**

- However, *V* cannot become arbitrarily small because we reach a point where no samples are contained in *V*, so we cannot get convergence this way.

- Alternate approach:
  - *V* cannot be allowed to become small since the number of samples is always limited.
  - One will have to accept a certain amount of variance in the ratio k/n.

$$p(\mathbf{x}) \cong \frac{k/n}{V}$$

- Fix the volume of region *V* and count the number of samples *k* (out of *n*) falling in *V* → Parzen Window

- Vary *V* in a way so that to enclose k samples around **x** and make a decision for the label of **x** **-**→ k- Nearest neighbor

- To estimate the density of **x**, we form a sequence of regions $R_1, R_2, \ldots$ containing **x**: the first region contains one sample, the second two samples and so on.

Let V_n be the volume of R_n,

k_n the number of samples falling in R_n and

$p_n(\mathbf{x})$ be the n^{th} estimate for p(x): $p_n(\mathbf{x}) = (k_n/n)/V_n$.

Theoretically, if an unlimited number of samples is available, we can show $p_n(\mathbf{x}) \rightarrow p(\mathbf{x})$.

- Three necessary conditions should apply if we want $p_n(\mathbf{x})$ to converge to $p(\mathbf{x})$:

$$1)\ \lim_{n\to\infty} V_n = 0$$

$$2)\ \lim_{n\to\infty} k_n = \infty$$

$$3)\ \lim_{n\to\infty} k_n / n = 0$$

Parzen Window

- Parzen-window approach to estimate densities assume that the region R_n is a d-dimensional hypercube:

$$V_n = h_n^d \quad (\mathrm{h_n : length\ of\ the\ edge\ of\ } \Re_n)$$

$$Let\ \varphi(\mathbf{u}) \text{ be the following window function :}$$

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \le \frac{1}{2} \quad j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases}$$

$\varphi(\frac{\mathbf{x - x_i}}{h})$ is equal to unity when $\mathbf{x}_i$ falls within hypercube of volume V_n centred at $\mathbf{x}$

Parzen Window

- The number of samples in this hypercube is:

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

- The estimate for $p_n(x)$ is:

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

- $p_n(\mathbf{x})$ estimates $p(\mathbf{x})$ as an average of functions of $\mathbf{x}$ and the samples $\{\mathbf{x}_i\}$ for i = 1,… ,n.
- These basis functions, φ, can be general!

Parzen Window

- We must choose functions that φ satisfy:

$$\int \varphi(\mathbf{x})\ \ d\mathbf{x} = 1$$

$$\varphi(\mathbf{x}) \geq 0$$

Illustration

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

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

where h_1 is a known parameter.

- Thus: $$p_n(x) = \frac{1}{n}\sum_{i=1}^{i=n} \frac{1}{h_n}\,\varphi\left(\frac{x - x_i}{h_n}\right)$$

is an average of normal densities centered at the samples x_i.

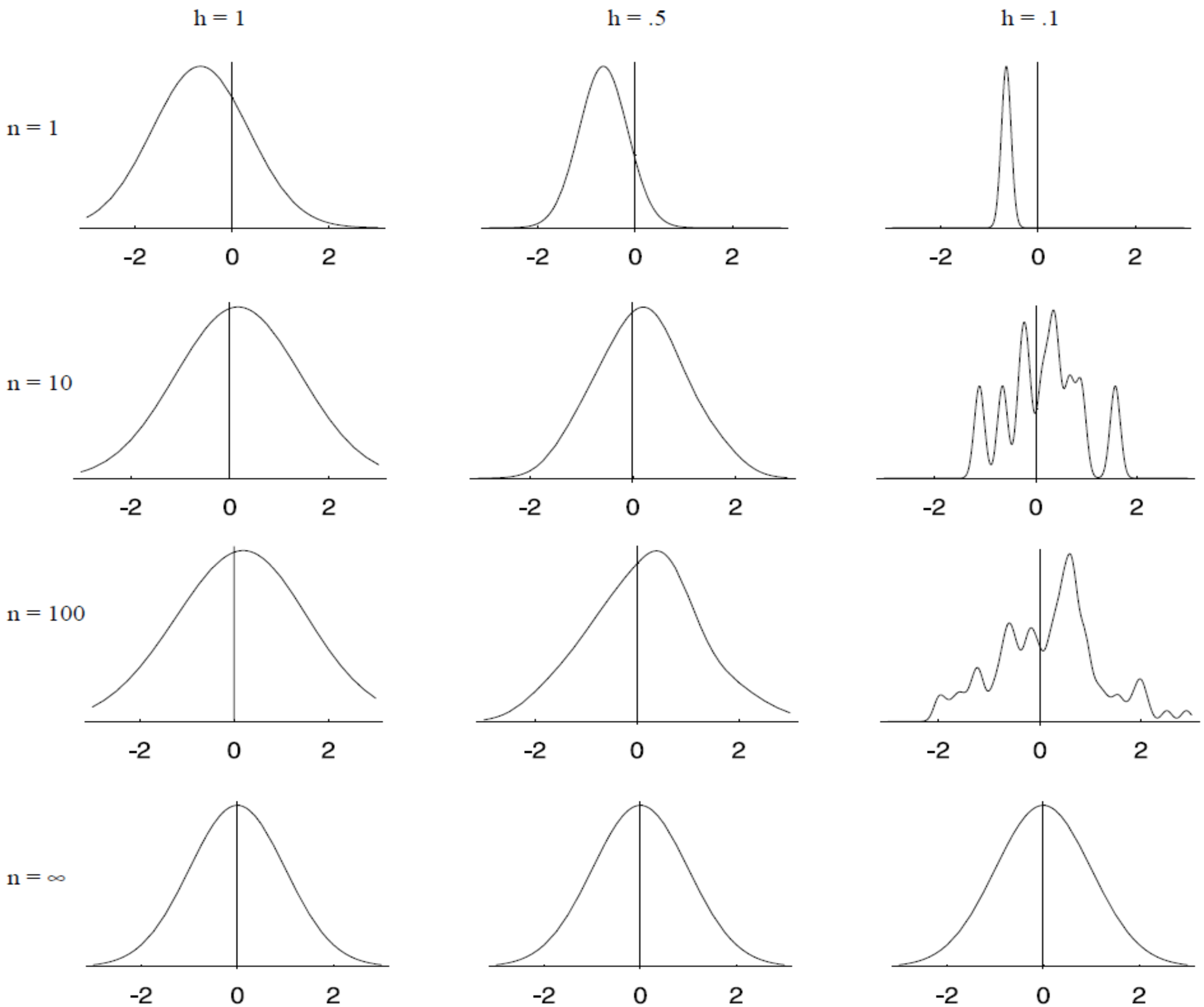
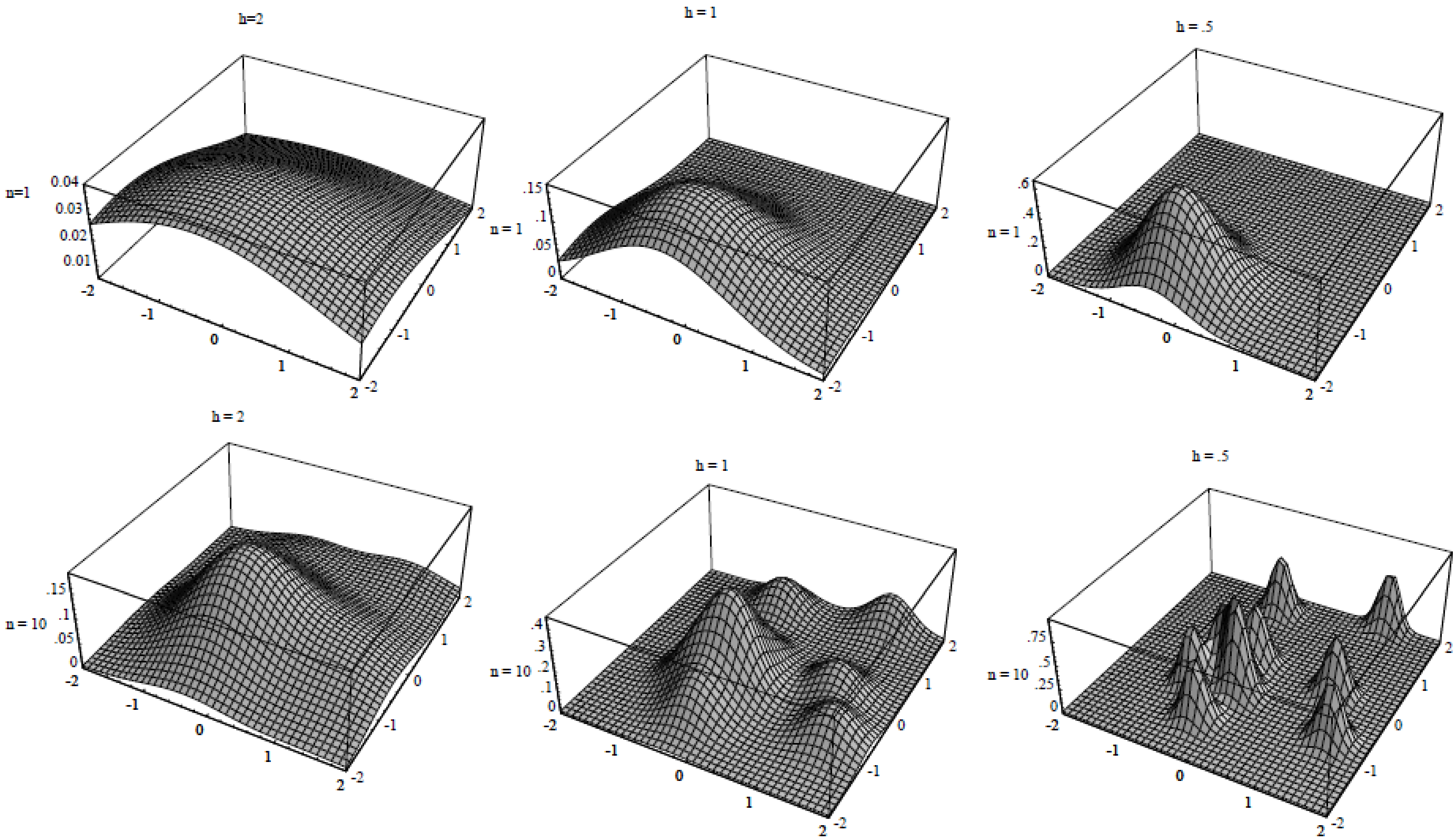
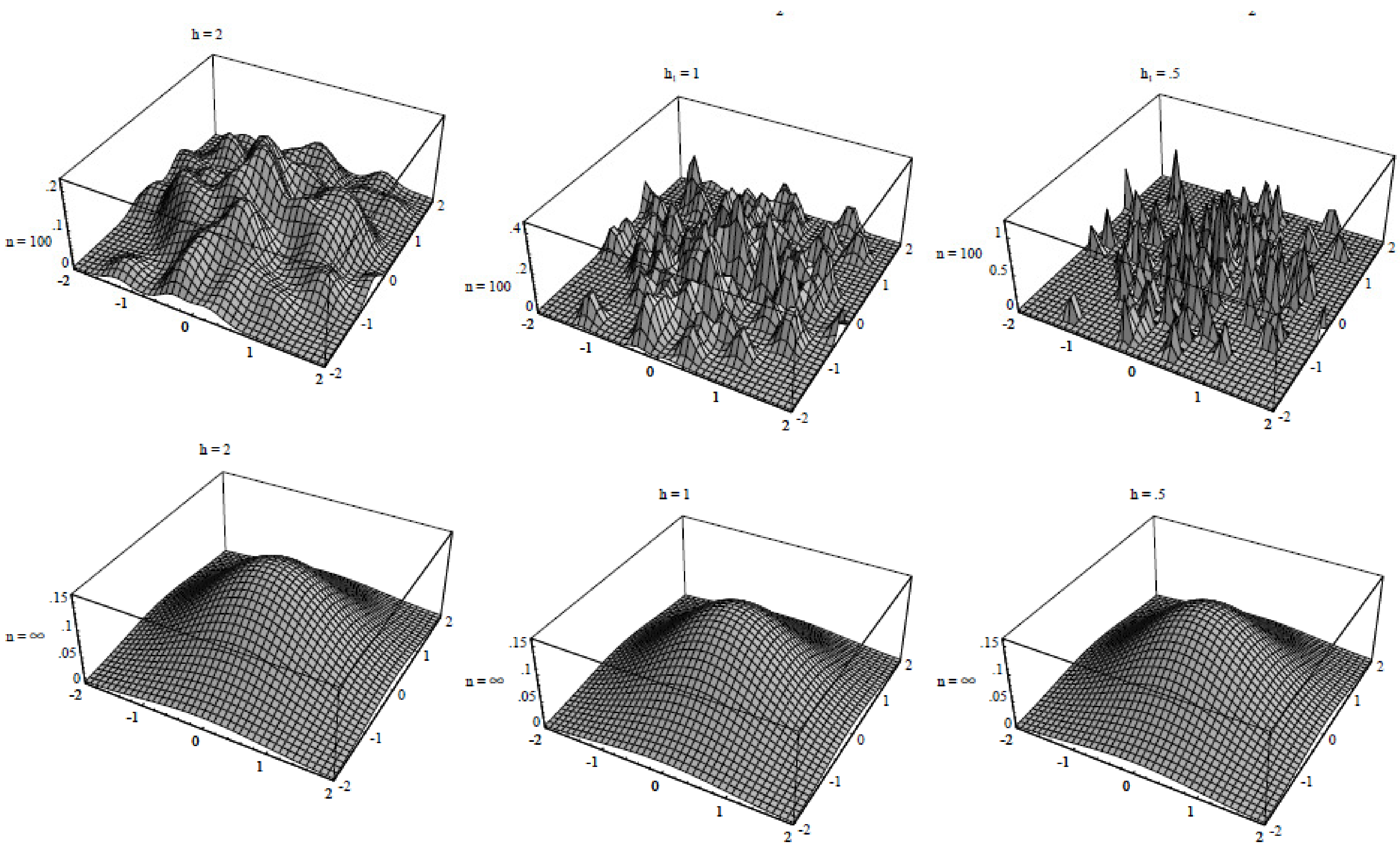


Figure 4.5: Parzen-window estimates of a univariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true generating function), regardless of window width h.


h=2
h = 1
h = .5
n=1
0.04
0.03
0.02
0.01
n = 1
.15
.1
.05
0
n = 1
.6
.4
.2
0
h = 2
h = 1
h = .5
n = 10
.15
.1
.05
0
n = 10
.4
.3
.2
.1
0
n = 10
.75
.5
.25
0

Parzen Window



Window size issue

If h_n is very large,

$p_n(\boldsymbol{x})$ *is the superposition of n* broad, slowly changing functions and is a very smooth “out-of-focus” estimate of $p(\boldsymbol{x})$

If h_n is very small,

$p(\boldsymbol{x})$ is the superposition of n sharp pulses centered at the samples — an erratic, “noisy” estimate

If V_n *is* too large, the estimate will suffer from too little resolution;

if V_n is too small, the estimate will suffer from too much statistical variability.

K Nearest neighbor

- **Goal**: a solution for the problem of the unknown “best” window function.
- Approach: Estimate density using data points.
- Let the cell volume be a function of the training data.
- Center a cell about **x** and let it grow until it captures k_n samples:

$$k_n = f(n)$$

- k_n are called the k_n nearest-neighbors of **x**.

K Nearest neighbor

- Two possibilities can occur:
  - Density is high near **x**; therefore the cell will be small which provides good resolution.
  - Density is low; therefore the cell will grow large and stop until higher density regions are reached.

K Nearest neighbor

- **Goal**: estimate $P_n(\omega_i \mid \mathbf{x})$ from a set of *n* labeled samples.
- Let's place a cell of volume *V* around **x** and capture *k* samples.
- k_i samples amongst *k* turned out to be labeled ω_i then:

$$p_n(\mathbf{x}, \omega_i) = \frac{k_i / n}{V}$$

- A reasonable estimate for $P_n(\omega_i \mid \mathbf{x})$ is:

$$P_n(\omega_i \mid \mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^{c} p_n(\mathbf{x}, \omega_j)} = \frac{k_i / nV}{\sum_{j=1}^{c} k_j / nV} = \frac{k_i}{k}$$

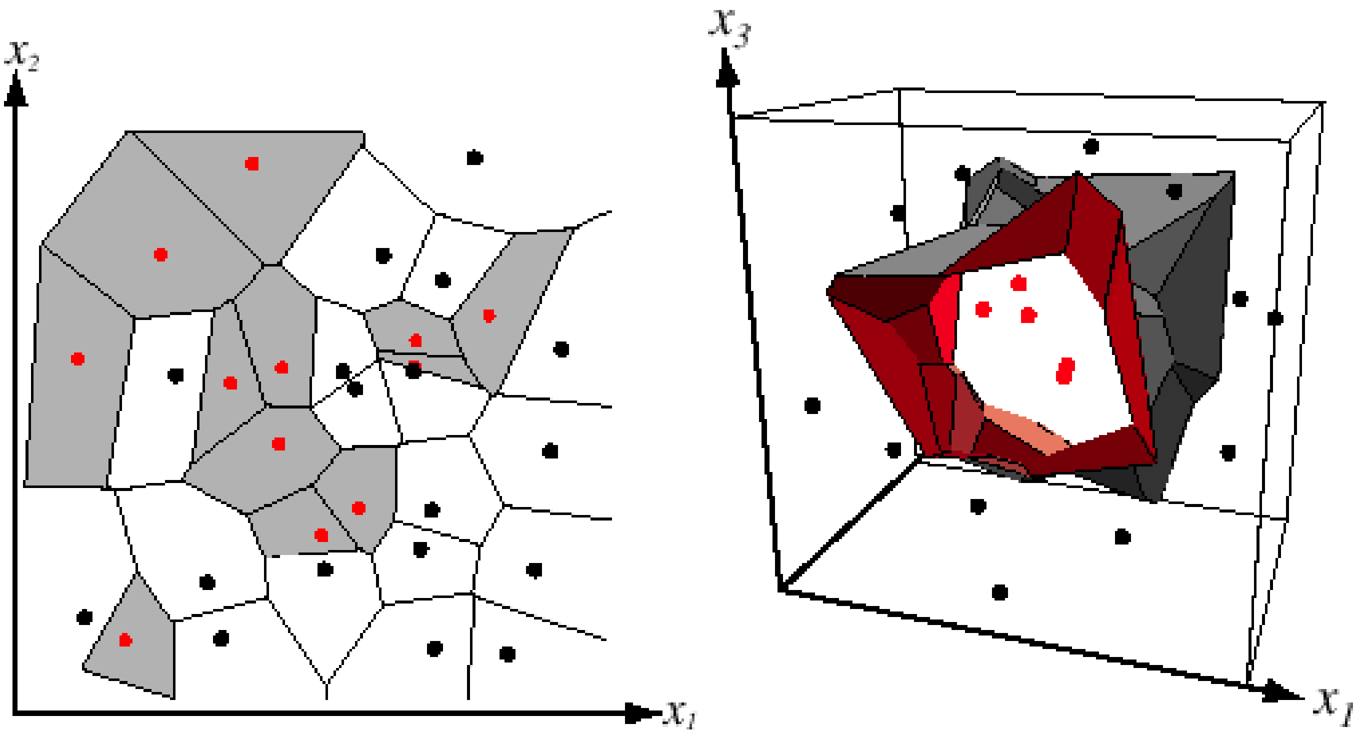
$\frac{k_i}{k}$ is the fraction of the samples within the cell that are labeled ω_i .

- For minimum error rate, the most frequently represented category within the cell is selected.
- If *k* is large and the cell sufficiently small, the performance will approach the best possible.

- Let $D_n = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of n labeled prototypes.
- Let $\mathbf{x}' \in D_n$ be the closest prototype to a test point $\mathbf{x}$.
- The nearest-neighbor rule for classifying $\mathbf{x}$ is to assign it the label associated with $\mathbf{x}'$
- The nearest-neighbor rule leads to an error rate greater than the minimum possible: the Bayes rate.
- If the number of prototypes is large (unlimited), the error rate of the nearest-neighbor classifier is never worse than twice the Bayes rate.
- If $n \to \infty$, it is always possible to find $\mathbf{x}'$ sufficiently close so that:

$$P(\omega_i \mid \mathbf{x}') \cong P(\omega_i \mid \mathbf{x})$$

K Nearest neighbor



- **This produces a Voronoi tesselation of the space, and the individual decision regions are called Voronoi cells.**
- **For large data sets, this approach can be very effective but not computationally efficient.**