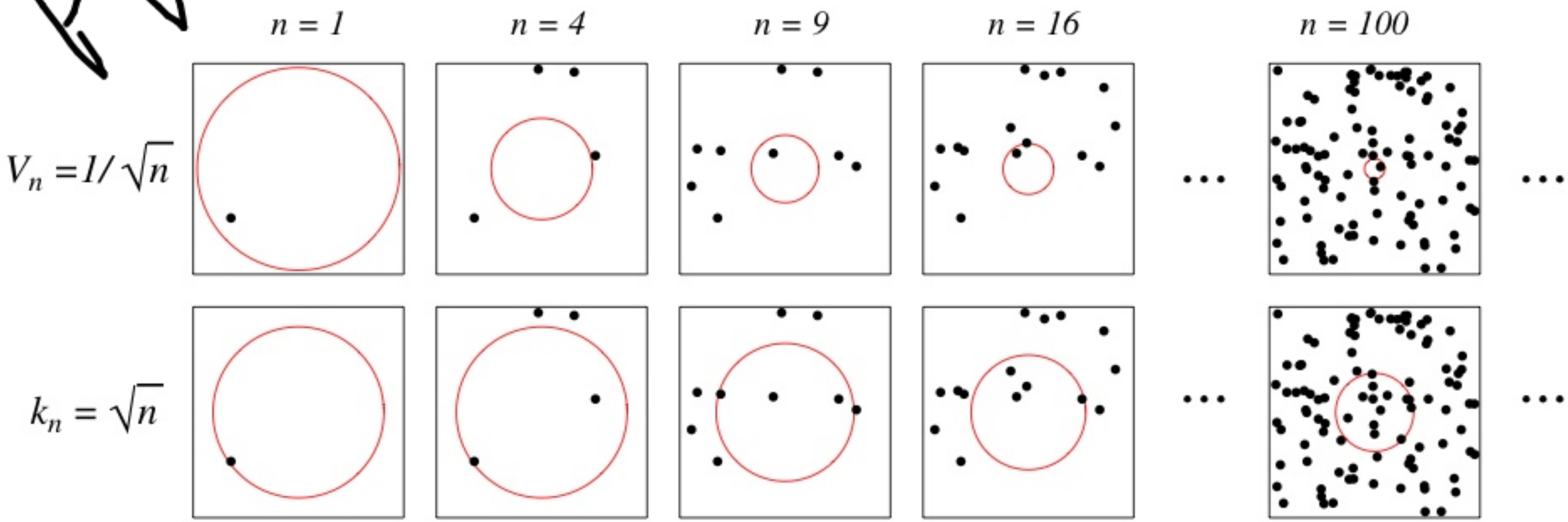
PW




KNN

# Parzen Windows

- Let's temporarily assume the region $\mathcal{R}$ is a $d$-dimensional hypercube with $h_n$ being the length of an edge.

# Parzen Windows

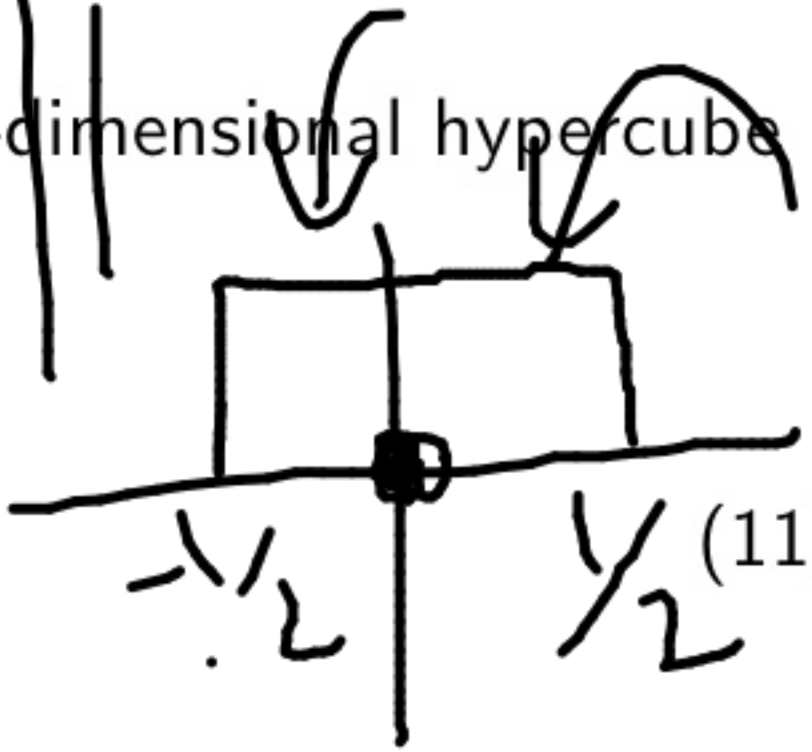
- Let's temporarily assume the region $\mathcal{R}$ is a $d$-dimensional hypercube with $h_n$ being the length of an edge.
- The volume of the hypercube is given by

$$V_n = h_n^d \quad . \tag{11}$$

# Parzen Windows

- Let's temporarily assume the region $\mathcal{R}$ is a $d$-dimensional hypercube with $h_n$ being the length of an edge.
- The volume of the hypercube is given by

$$V_n = h_n^d \quad . \tag{11}$$




- We can derive an analytic expression for $k_n$:
  - Define a **windowing function**:

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq 1/2 \qquad j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

  - This windowing function $\varphi$ defines a unit hypercube centered at the origin.
  - Hence, $\varphi((\mathbf{x} - \mathbf{x}_i)/h_n)$ 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 number of samples in this hypercube is therefore given by

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right) \quad . \tag{13}$$

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

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right). \tag{13}$$

- Substituting in equation (7), $p_n(\mathbf{x}) = k_n/(nV_n)$ yields the estimate

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right). \tag{14}$$

$\frac{1}{nV_n}$

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

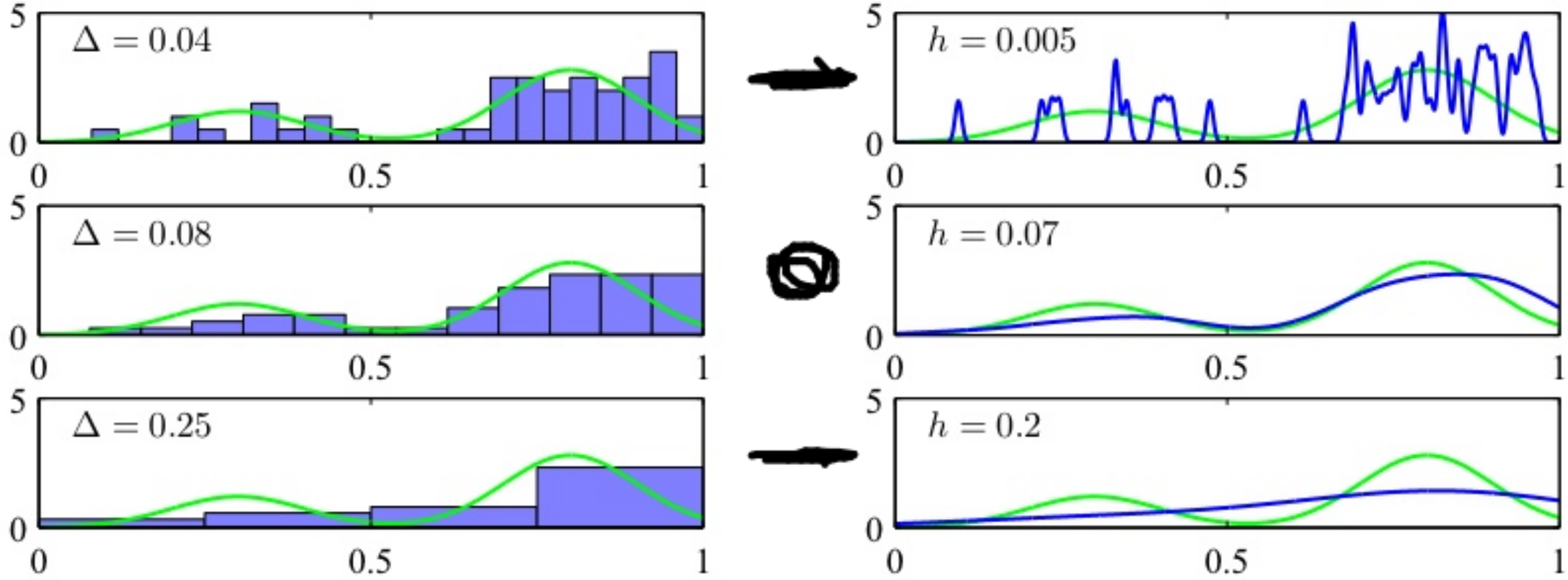
$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right) \quad . \tag{13}$$

- Substituting in equation (7), $p_n(\mathbf{x}) = k_n/(nV_n)$ yields the estimate

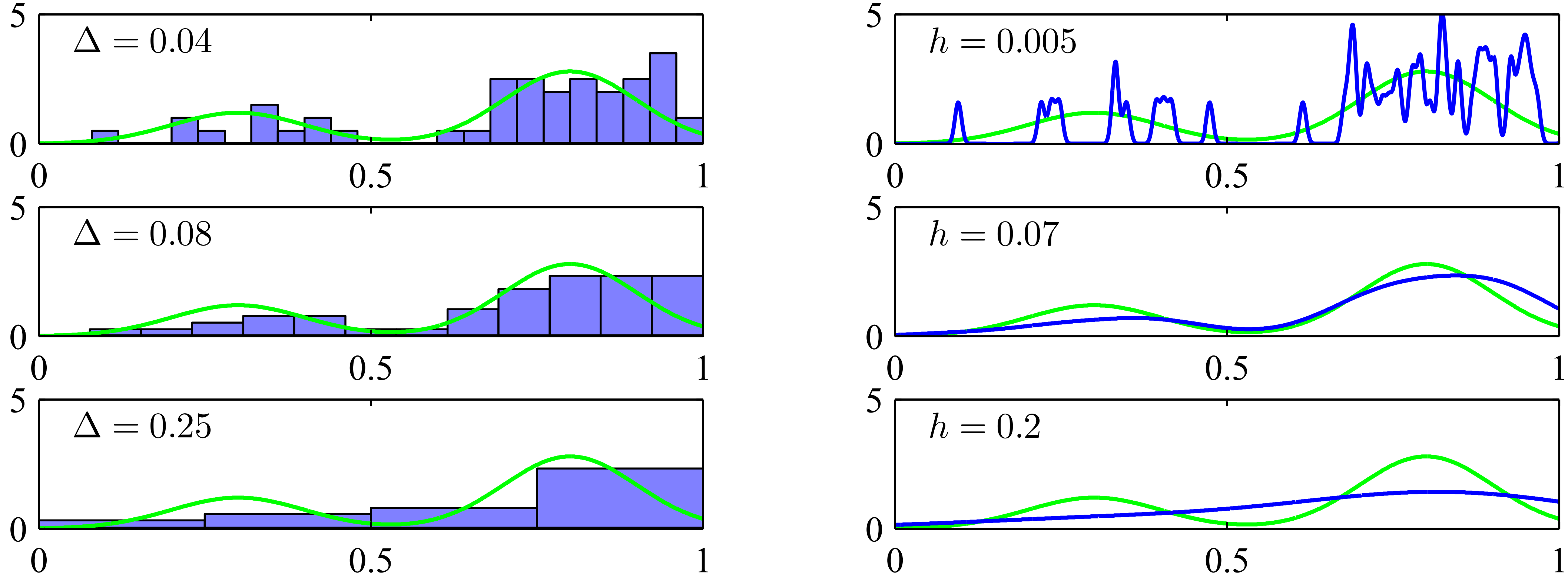
$$p_n(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right) \quad . \tag{14}$$

- Hence, the windowing function $\varphi$, in this context called a **Parzen window**, tells us how to **weight** all of the samples in $\mathcal{D}$ to determine $p(\mathbf{x})$ at a particular $\mathbf{x}$.

# Example

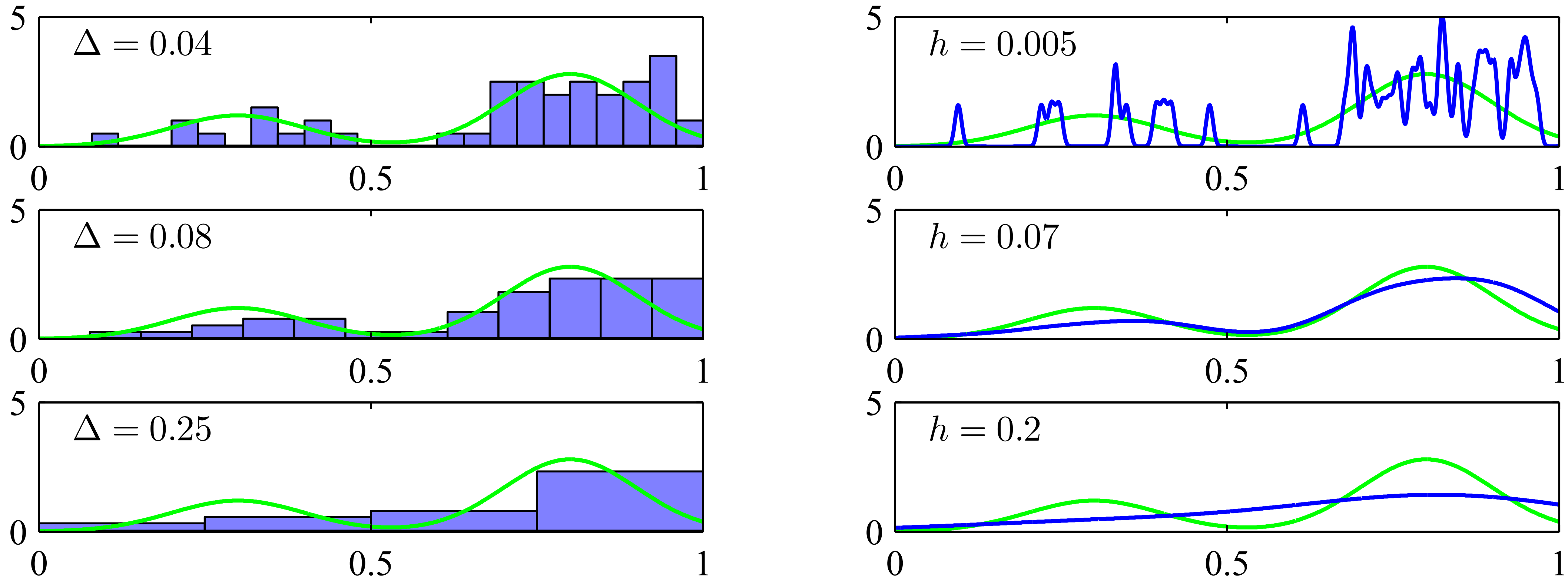


# Example




- But, what undesirable traits from histograms are inherited by Parzen window density estimates of the form we've just defined?

# Example




- But, what undesirable traits from histograms are inherited by Parzen window density estimates of the form we've just defined?
- Discontinuities...
- Dependence on the bandwidth.

# Generalizing the Kernel Function

- What if we allow a more general class of windowing functions rather than the hypercube?
- If we think of the windowing function as an interpolator, rather than considering the window function about $\mathbf{x}$ only, we can visualize it as a kernel sitting on each data sample $\mathbf{x}_i$ in $\mathcal{D}$.

# Generalizing the Kernel Function

- What if we allow a more general class of windowing functions rather than the hypercube?
- If we think of the windowing function as an interpolator, rather than considering the window function about $\mathbf{x}$ only, we can visualize it as a kernel sitting on each data sample $\mathbf{x}_i$ in $\mathcal{D}$.
- And, if we require the following two conditions on the kernel function $\varphi$, then we can be assured that the resulting density $p_n(\mathbf{x})$ will be proper: non-negative and integrate to 1.

$$\varphi(\mathbf{x}) \geq 0 \tag{15}$$

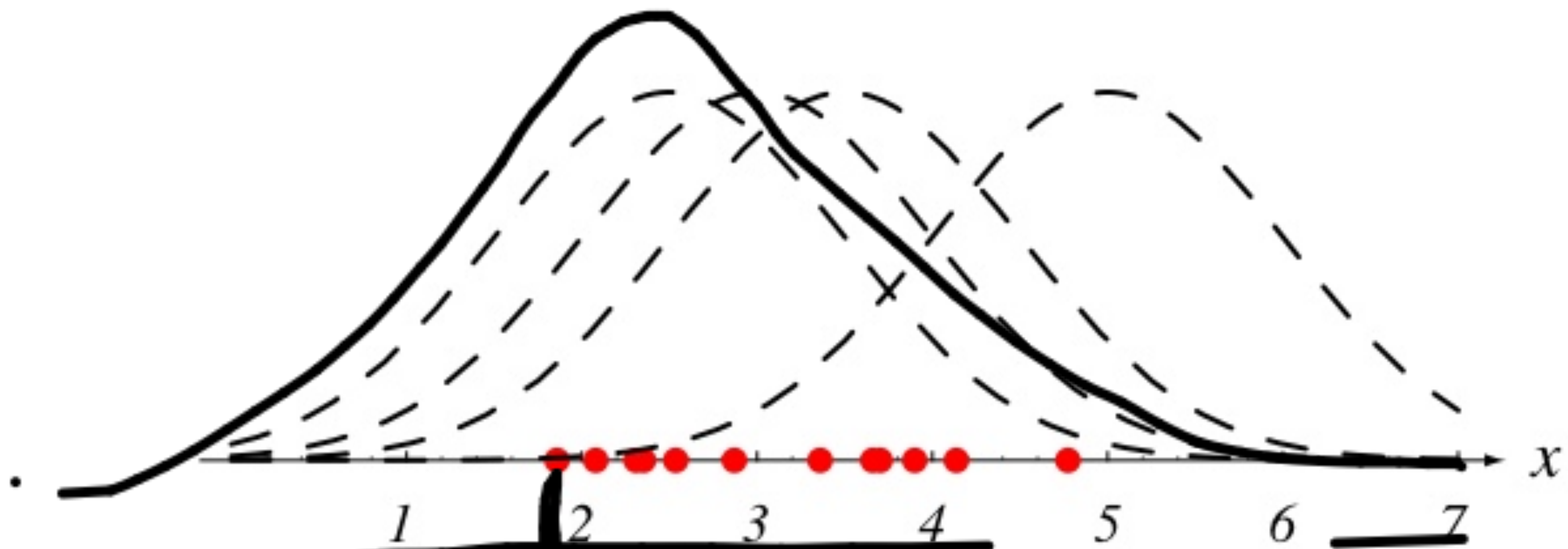
$$\int \varphi(\mathbf{u}) d\mathbf{u} = 1 \tag{16}$$

- For our previous case of $V_n = h_n^d$, then it follows $p_n(\mathbf{x})$ will also satisfy these conditions.

# Example: A Univariate Guassian Kernel

- A popular choice of the kernel is the Gaussian kernel:

$$\varphi_h(u) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}u^2\right] \tag{17}$$




- The resulting density is given by:

$$p(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_n\sqrt{2\pi}} \exp\left[-\frac{1}{2h_n^2}(\mathbf{x}-\mathbf{x}_i)^2\right] \tag{18}$$

- It will give us smoother estimates without the discontinuites from the hypercube kernel.

# Effect of the Window Width

## Slide I

- An important question is what effect does the window width $h_n$ have on $p_n(\mathbf{x})$?
- Define $\delta_n(\mathbf{x})$ as

$$\delta_n(\mathbf{x}) = \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}}{h_n}\right) \tag{19}$$

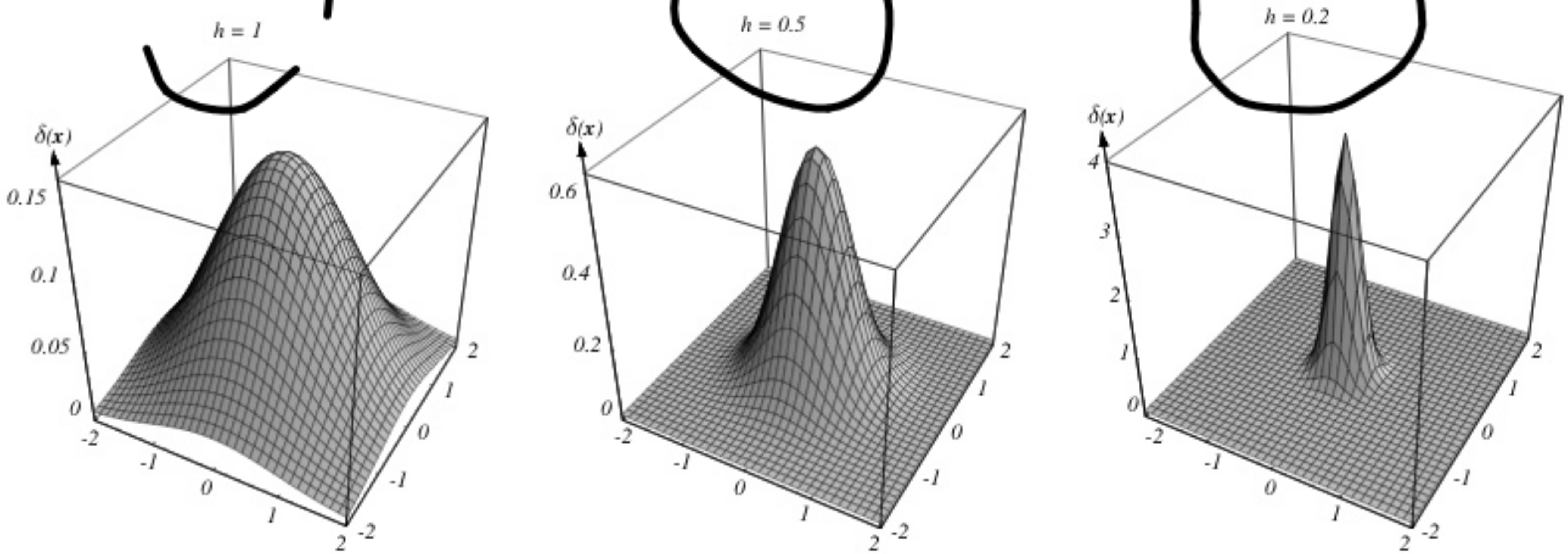
and rewrite $p_n(\mathbf{x})$ as the average

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(\mathbf{x}-\mathbf{x}_i) \tag{20}$$

# Effect of the Window Width

## Slide II

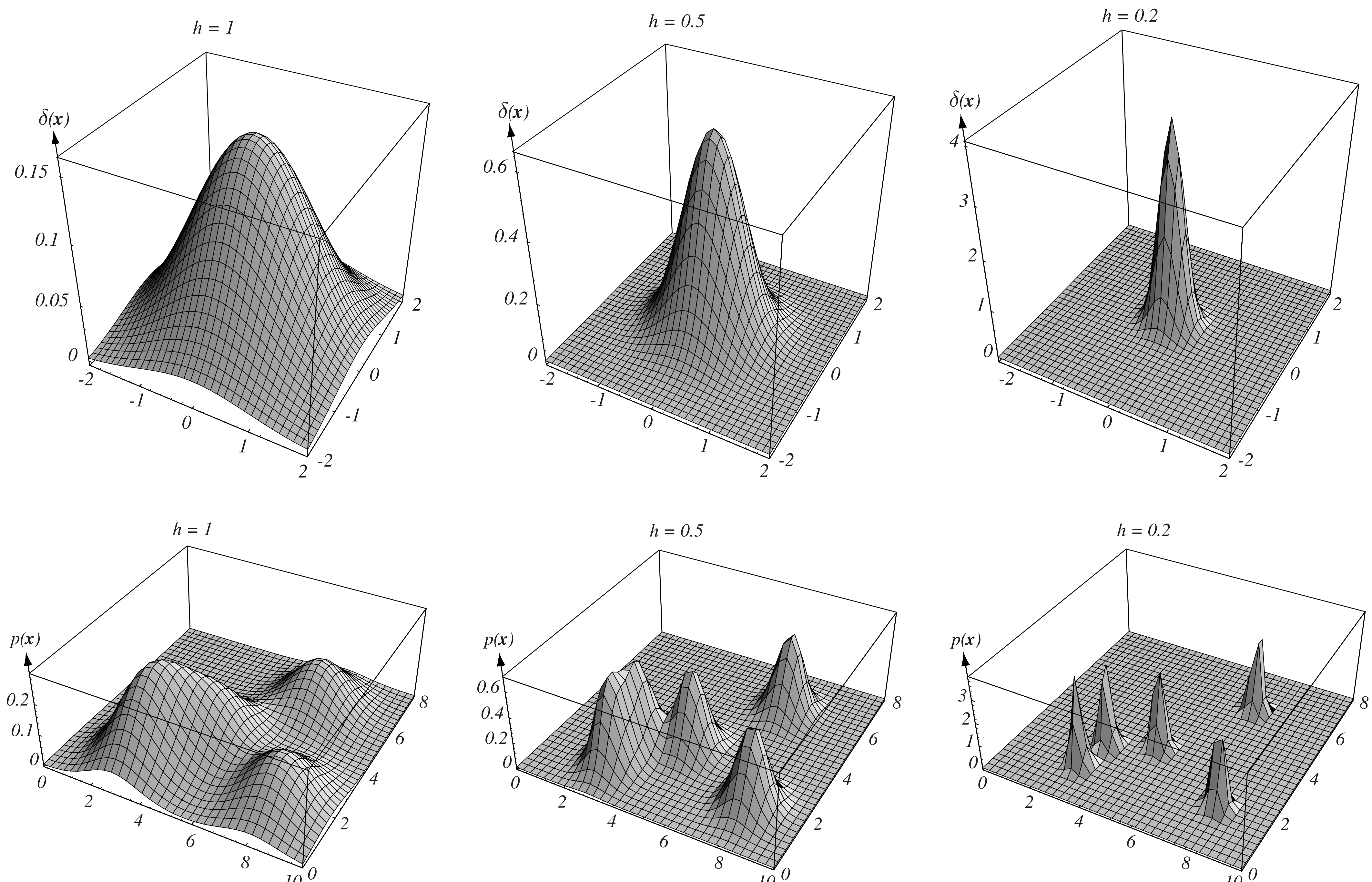
- $h_n$ clearly affects both the amplitude and the width of $\delta_n(\mathbf{x})$.



# Effect of the Window Width

## Slide II

- $h_n$ clearly affects both the amplitude and the width of $\delta_n(\mathbf{x})$.



# Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:

$$\int \delta(\mathbf{x} - \mathbf{x}_i) d\mathbf{x} = \int \frac{1}{V_n} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right) d\mathbf{x} = \int \varphi(\mathbf{u}) d\mathbf{u} = 1 \tag{21}$$

# Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:

$$\int \delta(\mathbf{x} - \mathbf{x}_i) d\mathbf{x} = \int \frac{1}{V_n} \varphi \left( \frac{\mathbf{x} - \mathbf{x}_i}{h_n} \right) d\mathbf{x} = \int \varphi(\mathbf{u}) d\mathbf{u} = 1 \qquad (21)$$

- If $V_n$ is too large, the estimate will suffer from too little resolution.

# Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:

$$\int \delta(\mathbf{x}-\mathbf{x}_i)d\mathbf{x} = \int \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)d\mathbf{x} = \int \varphi(\mathbf{u})d\mathbf{u} = 1 \qquad (21)$$

- 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 variability.

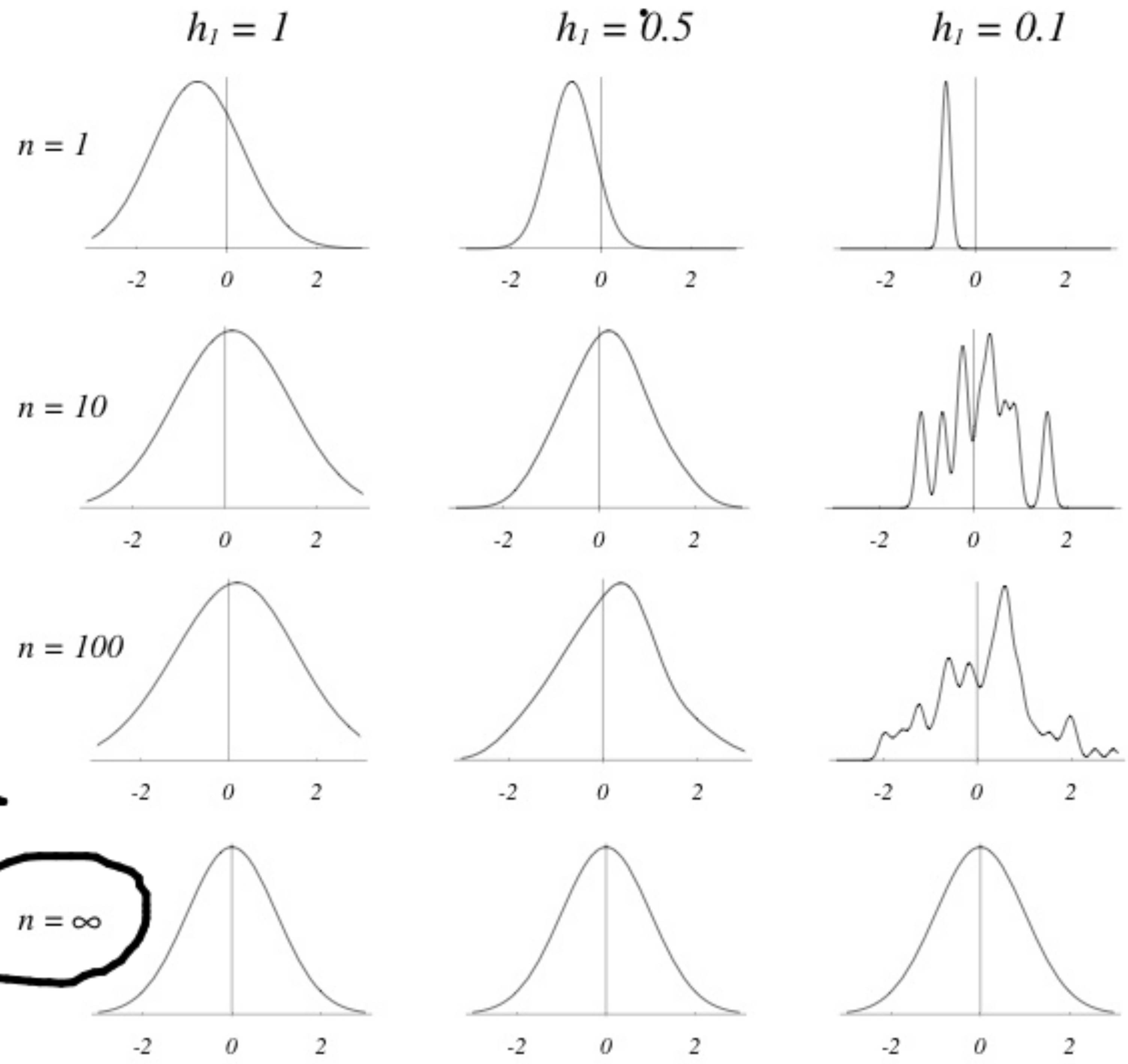
# Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:

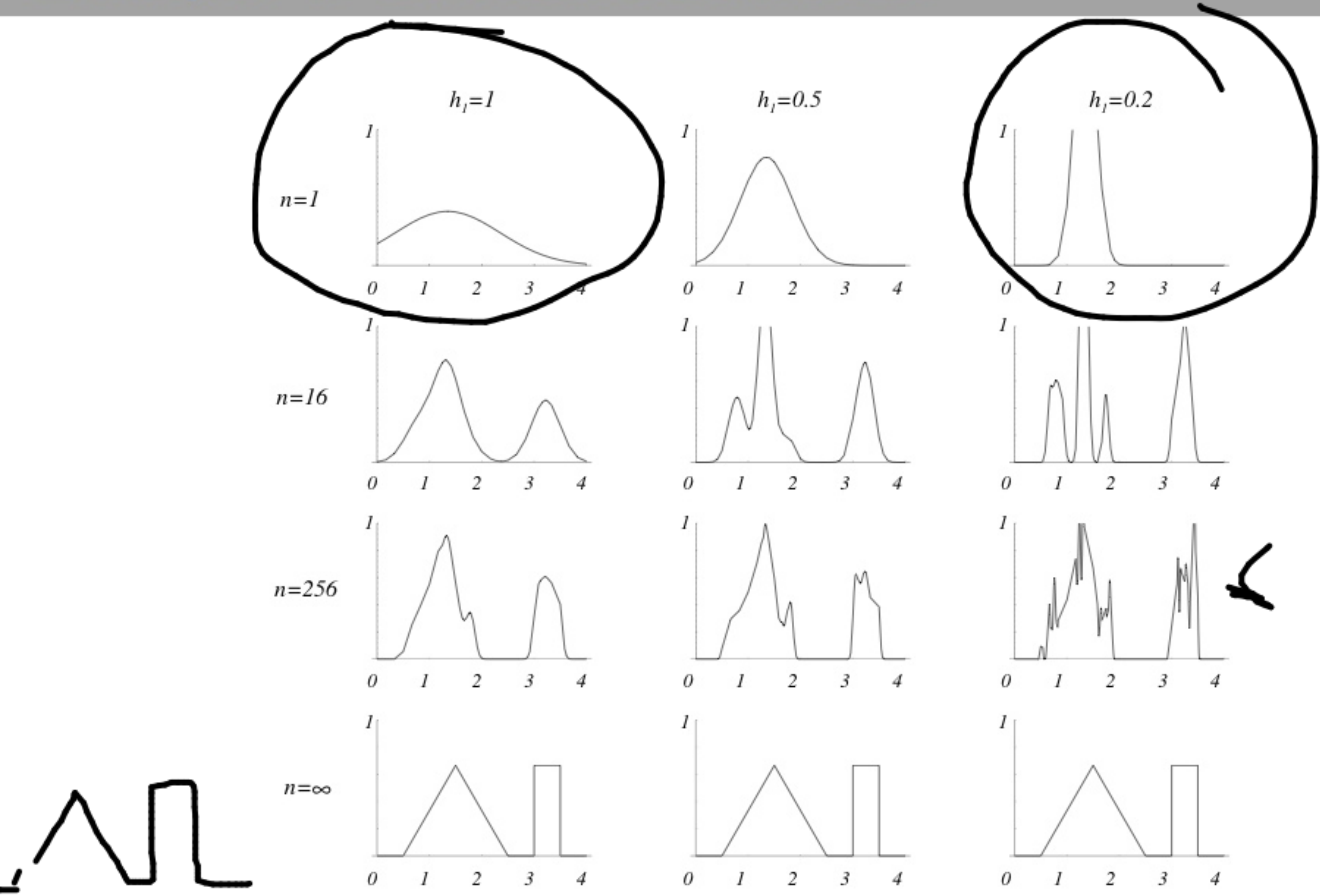
$$\int \delta(\mathbf{x}-\mathbf{x}_i)d\mathbf{x} = \int \frac{1}{V_n}\varphi\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)d\mathbf{x} = \int \varphi(\mathbf{u})d\mathbf{u} = 1 \tag{21}$$

- 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 variability.
- In theory (with an unlimited number of samples), we can let $V_n$ slowly approach zero as $n$ increases and then $p_n(\mathbf{x})$ will converge to the unknown $p(\mathbf{x})$. But, in practice, we can, at best, seek some compromise.

# Example: Revisiting the Univariate Guassian Kernel



# Example: A Bimodal Distribution

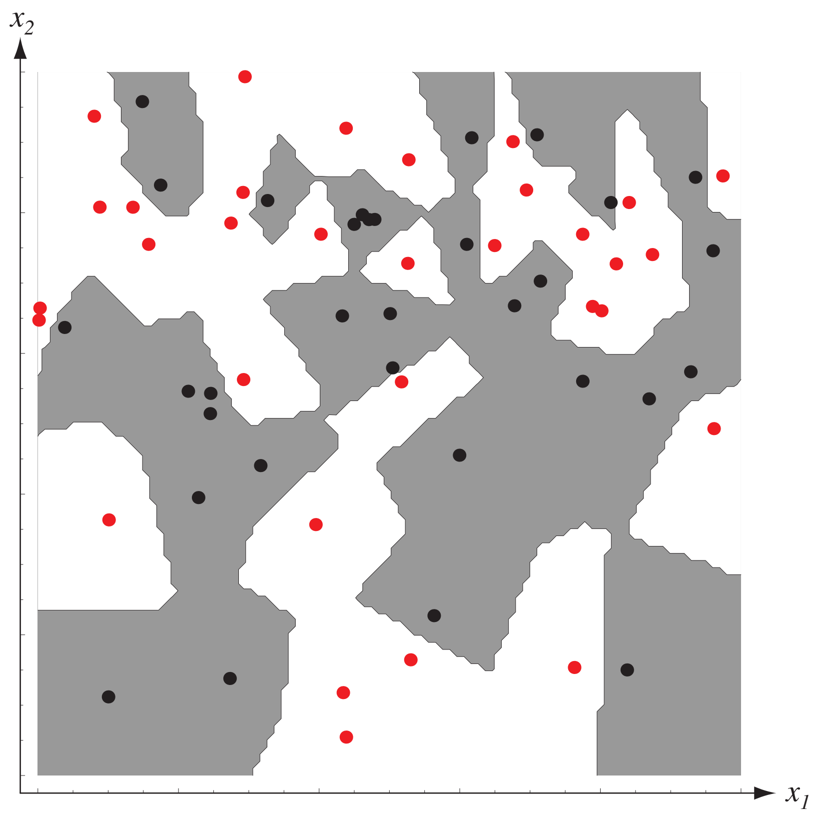


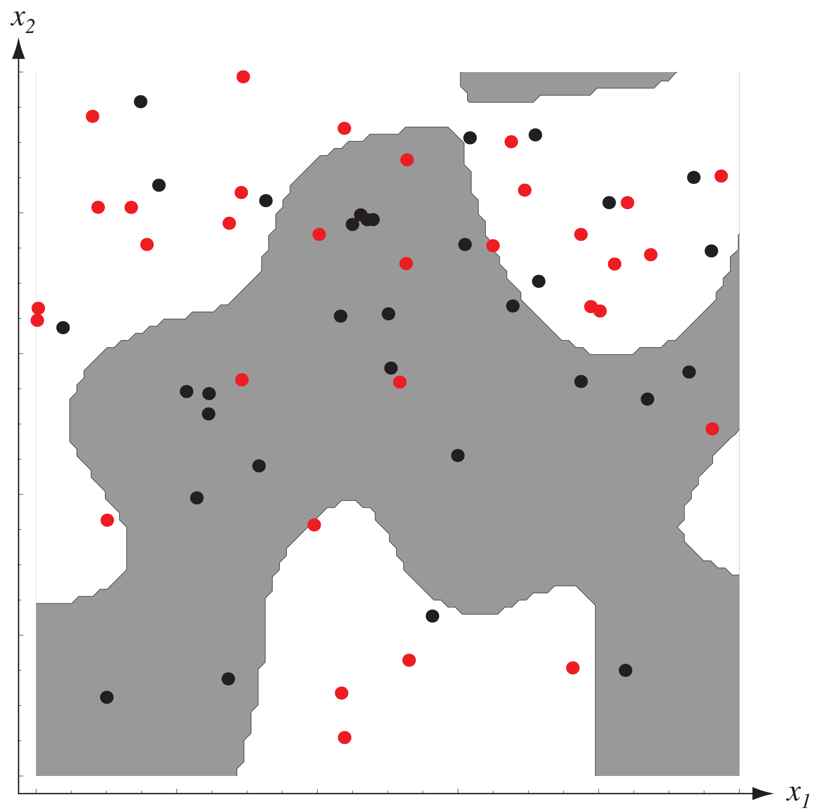
# Parzen Window-Based Classifiers

- Estimate the densities for each category.
- Classify a query point by the label corresponding to the maximum posterior (i.e., one can include priors).

# Parzen Window-Based Classifiers

- Estimate the densities for each category.
- Classify a query point by the label corresponding to the maximum posterior (i.e., one can include priors).
- As you guessed it, the **decision regions for a Parzen window-based classifier depend upon the kernel function**.



# Parzen Window-Based Classifiers

- During training, we can make the error arbitrarily low by making the window sufficiently small, but this will have an ill-effect during testing (which is our ultimate need).
- Think of any possibilities for system rules of choosing the kernel?

# Parzen Window-Based Classifiers

- During training, we can make the error arbitrarily low by making the window sufficiently small, but this will have an ill-effect during testing (which is our ultimate need).
- Think of any possibilities for system rules of choosing the kernel?
- One possibility is to use cross-validation. Break up the data into a training set and a validation set. Then, perform training on the training set with varying bandwidths. Select the bandwidth that minimizes the error on the validation set.

# Parzen Window-Based Classifiers

- During training, we can make the error arbitrarily low by making the window sufficiently small, but this will have an ill-effect during testing (which is our ultimate need).
- Think of any possibilities for system rules of choosing the kernel?
- One possibility is to use cross-validation. Break up the data into a training set and a validation set. Then, perform training on the training set with varying bandwidths. Select the bandwidth that minimizes the error on the validation set.
- There is little theoretical justification for choosing one window width over another.

# $k_n$ Nearest Neighbor Methods

- Selecting the best window / bandwidth is a severe limiting factor for Parzen window estimators.
- $k_n$-NN methods circumvent this problem by making the window size a function of the actual training data.

# $k_n$ Nearest Neighbor Methods

- Selecting the best window / bandwidth is a severe limiting factor for Parzen window estimators.
- $k_n$-NN methods circumvent this problem by making the window size a function of the actual training data.
- The basic idea here is to center our window around $\mathbf{x}$ and let it grow until it captures $k_n$ samples, where $k_n$ is a function of $n$.
  - These samples are the $k_n$ nearest neighbors of $\mathbf{x}$.
  - If the density is high near $\mathbf{x}$ then the window will be relatively small leading to good resolution.
  - If the density is low near $\mathbf{x}$, the window will grow large, but it will stop soon after it enters regions of higher density.

# $k_n$ Nearest Neighbor Methods

- Selecting the best window / bandwidth is a severe limiting factor for Parzen window estimators.
- $k_n$-NN methods circumvent this problem by making the window size a function of the actual training data.
- The basic idea here is to center our window around $\mathbf{x}$ and let it grow until it captures $k_n$ samples, where $k_n$ is a function of $n$.
  - These samples are the $k_n$ nearest neighbors of $\mathbf{x}$.
  - If the density is high near $\mathbf{x}$ then the window will be relatively small leading to good resolution.
  - If the density is low near $\mathbf{x}$, the window will grow large, but it will stop soon after it enters regions of higher density.
  - In either case, we estimate $p_n(\mathbf{x})$ according to

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n} \tag{22}$$

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n}$$

- We want $k_n$ to go to infinity as $n$ goes to infinity thereby assuring us that $k_n/n$ will be a good estimate of the probability that a point will fall in the window of volume $V_n$.

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n}$$

- We want $k_n$ to go to infinity as $n$ goes to infinity thereby assuring us that $k_n/n$ will be a good estimate of the probability that a point will fall in the window of volume $V_n$.
- But, we also want $k_n$ to grow sufficiently slowly so that the size of our window will go to zero.

$$p_n(\mathbf{x}) = \frac{k_n}{nV_n}$$

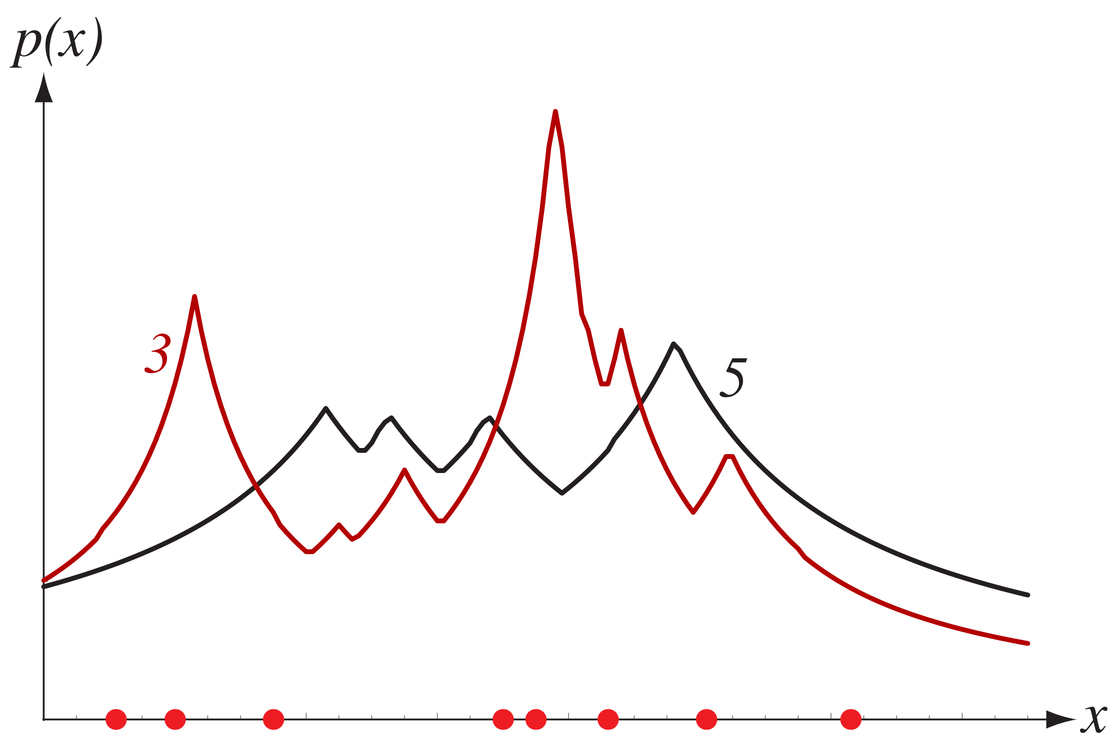
- We want $k_n$ to go to infinity as $n$ goes to infinity thereby assuring us that $k_n/n$ will be a good estimate of the probability that a point will fall in the window of volume $V_n$.
- But, we also want $k_n$ to grow sufficiently slowly so that the size of our window will go to zero.
- Thus, we want $k_n/n$ to go to zero.

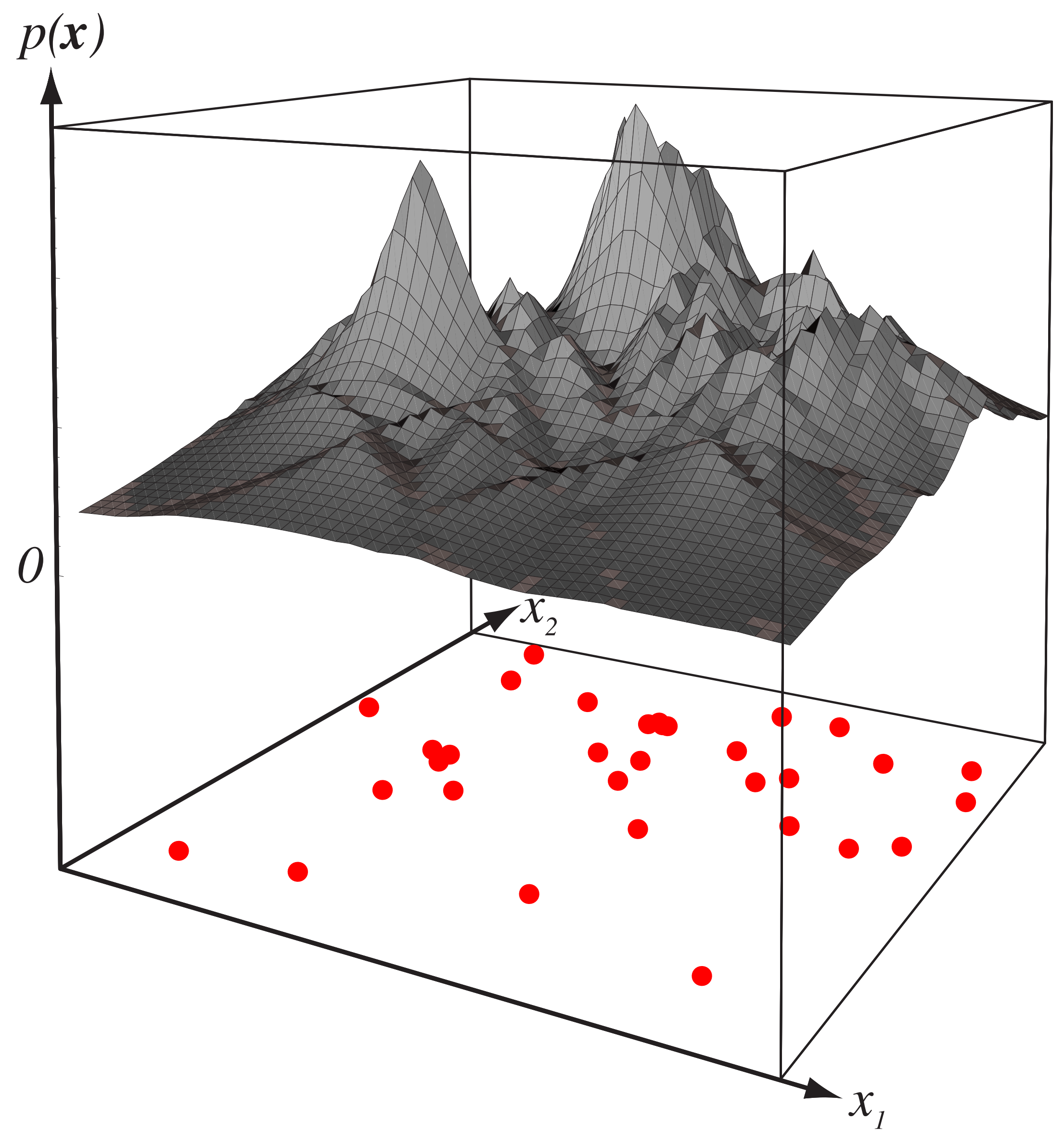
$$p_n(\mathbf{x}) = \frac{k_n}{nV_n}$$

- We want $k_n$ to go to infinity as $n$ goes to infinity thereby assuring us that $k_n/n$ will be a good estimate of the probability that a point will fall in the window of volume $V_n$.
- But, we also want $k_n$ to grow sufficiently slowly so that the size of our window will go to zero.
- Thus, we want $k_n/n$ to go to zero.
- Recall these conditions from the earlier discussion; these will ensure that $p_n(\mathbf{x})$ converges to $p(\mathbf{x})$ as $n$ approaches infinity.

# Examples of $k_n$-NN Estimation

- Notice the discontinuities in the slopes of the estimate.



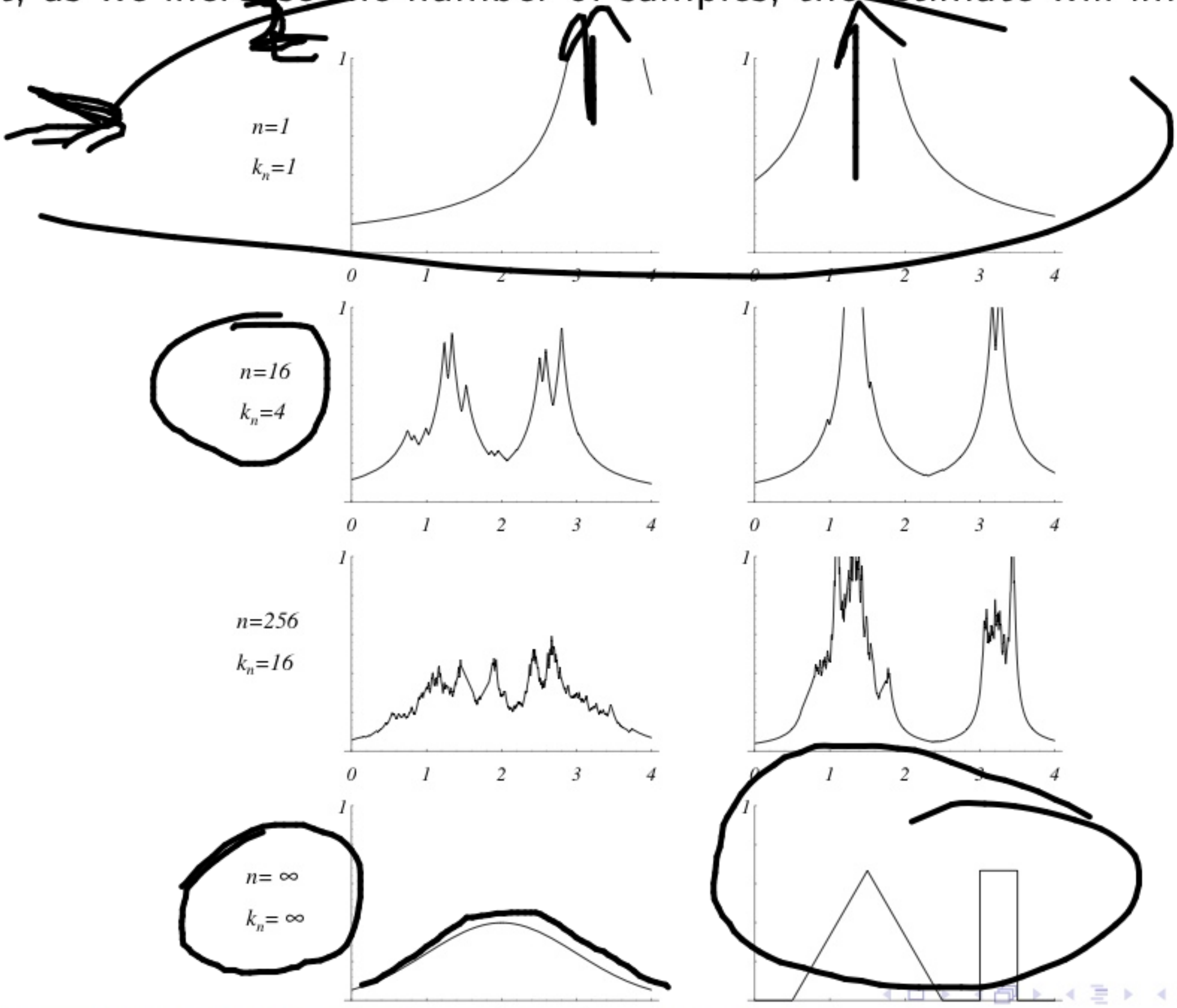



# $k$-NN Estimation From 1 Sample

- We don't expect the density estimate from 1 sample to be very good, but in the case of $k$-NN it will diverge!
- With $n = 1$ and $k_n = \sqrt{n} = 1$, the estimate for $p_n(x)$ is

$$p_n(x) = \frac{1}{2|x - x_1|} \tag{23}$$

But, as we increase the number of samples, the estimate will improve.



# Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?

# Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).

# Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
- One could obtain a sequence of estimates by taking $k_n = k_1\sqrt{n}$ and choose different values of $k_1$.

# Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
- One could obtain a sequence of estimates by taking $k_n = k_1\sqrt{n}$ and choose different values of $k_1$.
- But, like the Parzen window size, one choice is as good as another absent any additional information.

# Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
- One could obtain a sequence of estimates by taking $k_n = k_1\sqrt{n}$ and choose different values of $k_1$.
- But, like the Parzen window size, one choice is as good as another absent any additional information.
- Similarly, in classification scenarios, we can base our judgement on classification error.

# $k$-NN Posterior Estimation for Classification

- We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|\mathbf{x})$ from a set of $n$ labeled samples.

# $k$-NN Posterior Estimation for Classification

- We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|\mathbf{x})$ from a set of $n$ labeled samples.
- Place a window of volume $V$ around $\mathbf{x}$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.

# $k$-NN Posterior Estimation for Classification

- We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|\mathbf{x})$ from a set of $n$ labeled samples.
- Place a window of volume $V$ around $\mathbf{x}$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.
- The estimate for the joint probability is thus

$$p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV} \tag{24}$$

$$p_n(x) = \sum_{\omega_i} p(x, \omega_i)$$

# $k$-NN Posterior Estimation for Classification

- We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|\mathbf{x})$ from a set of $n$ labeled samples.
- Place a window of volume $V$ around $\mathbf{x}$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.
- The estimate for the joint probability is thus

$$p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV} \tag{24}$$

- A reasonable estimate for the posterior is thus

$$P_n(\omega_i|\mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_c p_n(\mathbf{x}, \omega_c)} = \frac{k_i}{k} \tag{25}$$

$$\frac{k_i}{k}$$

# $k$-NN Posterior Estimation for Classification

- We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|\mathbf{x})$ from a set of $n$ labeled samples.
- Place a window of volume $V$ around $\mathbf{x}$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.
- The estimate for the joint probability is thus

$$p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV} \tag{24}$$

- A reasonable estimate for the posterior is thus

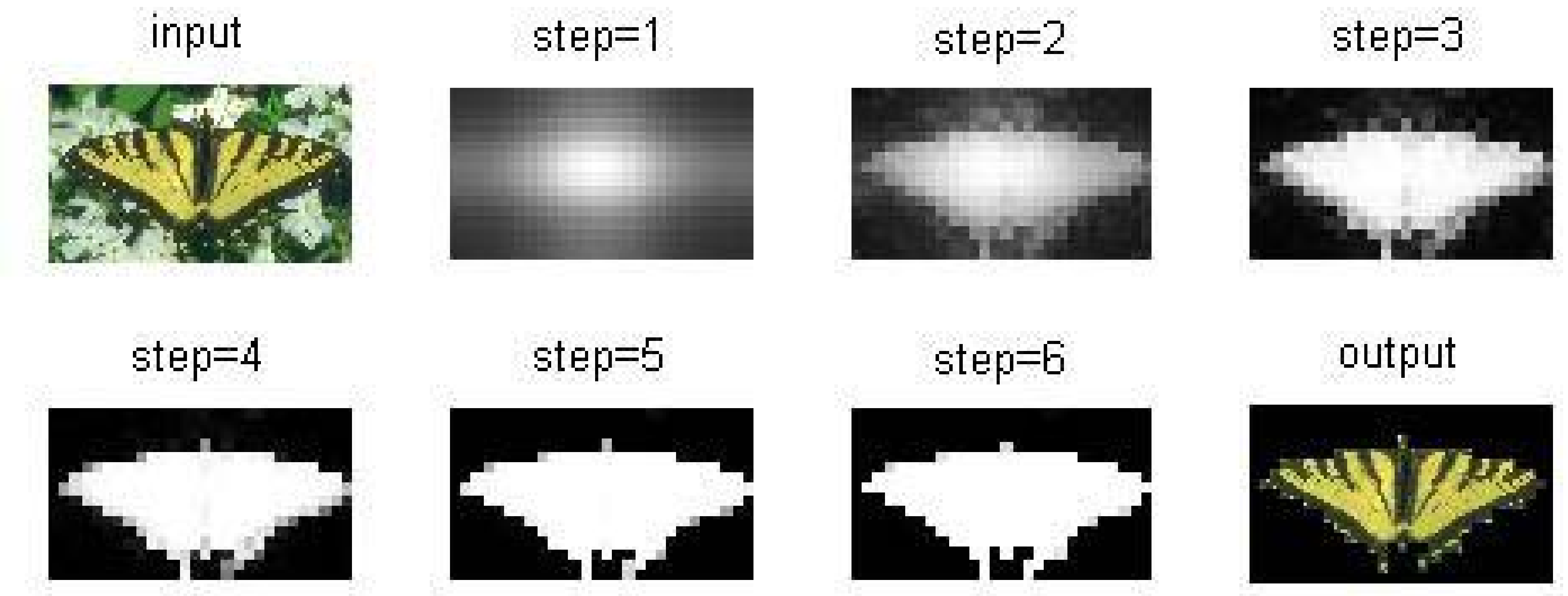
$$P_n(\omega_i|\mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_c p_n(\mathbf{x}, \omega_c)} = \frac{k_i}{k} \tag{25}$$

- Hence, the posterior probability for $\omega_i$ is simply the fraction of samples within the window that are labeled $\omega_i$. This is a simple and intuitive result.

# Example: Figure-Ground Discrimination

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**

- Figure-ground discrimination is an important low-level vision task.
- Want to separate the pixels that contain some foreground object (specified in some meaningful way) from the background.



# Example: Figure-Ground Discrimination

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**

- This paper presents a method for figure-ground discrimination based on non-parametric densities for the foreground and background.
- They use a subset of the pixels from each of the two regions.
- They propose an algorithm called **iterative sampling-expectation** for performing the actual segmentation.
- The required input is simply a region of interest (mostly) containing the object.

# Example: Figure-Ground Discrimination

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**

- Given a set of $n$ samples $S = \{\mathbf{x}_i\}$ where each $\mathbf{x}_i$ is a $d$-dimensional vector.
- We know the kernel density estimate is defined as

$$\hat{p}(\mathbf{y}) = \frac{1}{n\sigma_1 \ldots \sigma_d} \sum_{i=1}^{n} \prod_{j=1}^{d} \varphi\left(\frac{\mathbf{y}_j - \mathbf{x}_{ij}}{\sigma_j}\right) \tag{26}$$

where the same kernel $\varphi$ with different bandwidth $\sigma_j$ is used in each dimension.

# The Representation

## Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

- The representation used here is a function of RGB:

$$r = R/(R+G+B) \tag{27}$$
$$g = G/(R+G+B) \tag{28}$$
$$s = (R+G+B)/3 \tag{29}$$

- Separating the chromaticity from the brightness allows them to us a wider bandwidth in the brightness dimension to account for variability due to shading effects.
- And, much narrower kernels can be used on the $r$ and $g$ chromaticity channels to enable better discrimination.

# The Color Density

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**

- Given a sample of pixels $S = \{\mathbf{x}_i = (r_i, g_i, s_i)\}$, the color density estimate is given by

$$\hat{P}(\mathbf{x} = (r, g, s)) = \frac{1}{n} \sum_{i=1}^{n} K_{\sigma_r}(r - r_i) K_{\sigma_g}(g - g_i) K_{\sigma_s}(s - s_i) \tag{30}$$

where we have simplified the kernel definition:

$$K_\sigma(t) = \frac{1}{\sigma} \varphi\left(\frac{t}{\sigma}\right) \tag{31}$$

- They use Gaussian kernels

$$K_\sigma(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{t}{\sigma}\right)^2\right] \tag{32}$$

with a different bandwidth in each dimension.

# Data-Driven Bandwidth

## Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

- The bandwidth for each channel is calculated directly from the image based on sample statistics.

$$\sigma \approx 1.06\hat{\sigma}n^{-1/5} \tag{33}$$

where $\hat{\sigma}^2$ is the sample variance.

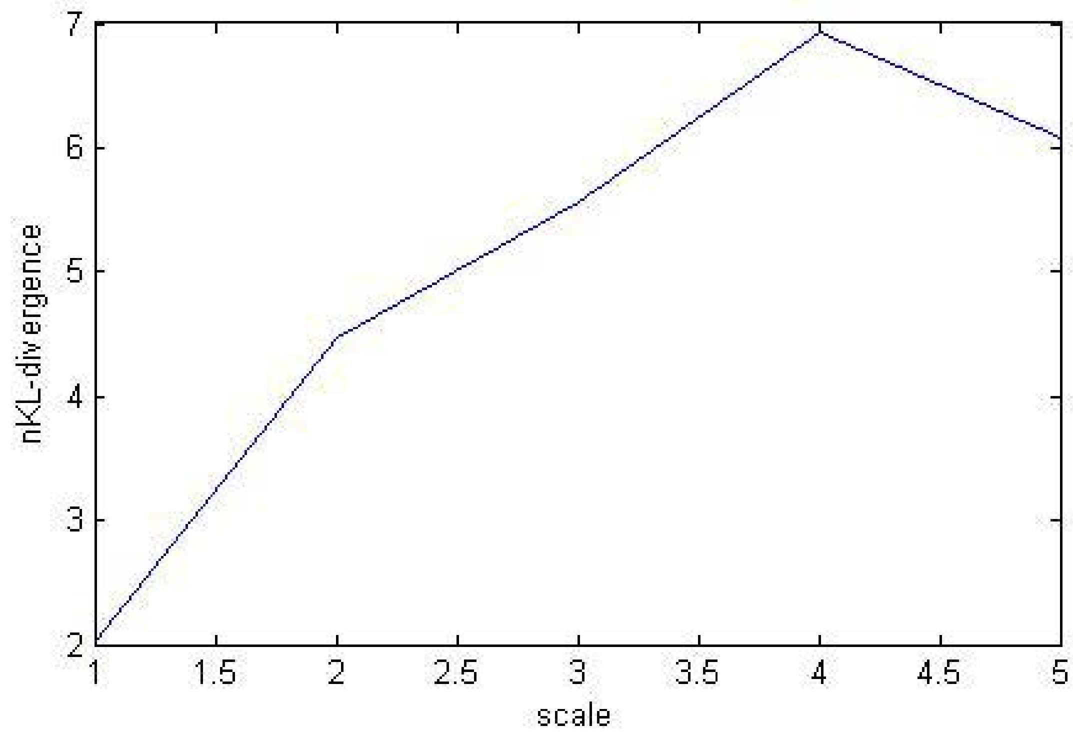
# Initialization: Choosing the Initial Scale

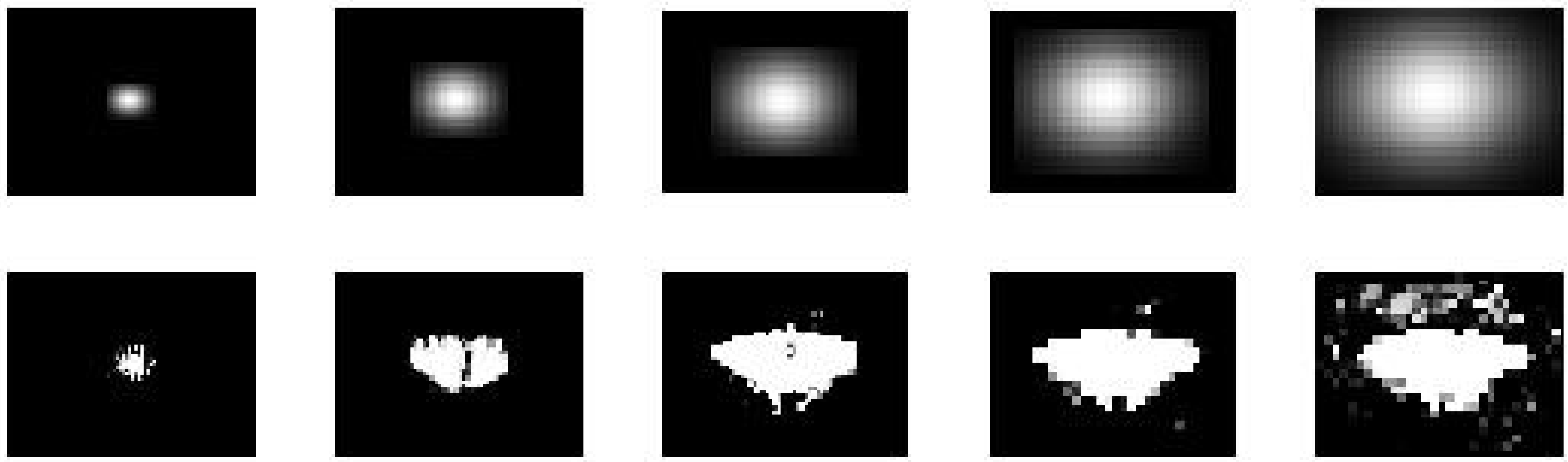
**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**

- For initialization, they compute a distance between the foreground and background distribution by varying the scale of a single Gaussian kernel (on the foreground).
- To evaluate the "significance" of a particular scale, they compute the normalized KL-divergence:

$$\text{nKL}(\hat{P}_{fg}||\hat{P}_{bg}) = \frac{-\sum_{i=1}^{n} \hat{P}_{fg}(\mathbf{x}_i) \log \frac{\hat{P}_{fg}(\mathbf{x}_i)}{\hat{P}_{bg}(\mathbf{x}_i)}}{\sum_{i=1}^{n} \hat{P}_{fg}(\mathbf{x}_i)} \tag{34}$$

where $\hat{P}_{fg}$ and $\hat{P}_{bg}$ are the density estimates for the foreground and background regions respectively. To compute each, they use about 6% of the pixels (using all of the pixels would lead to quite slow performance).


nKL-divergence
scale
7
6
5
4
3
2
1
1.5
2
2.5
3
3.5
4
4.5
5



# Iterative Sampling-Expectation Algorithm

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**

- Given the initial segmentation, they need to refine the models **and** labels to adapt better to the image.
- However, this is a chicken-and-egg problem. If we know the labels, we could compute the models, and if we knew the models, we could compute the best labels.

# Iterative Sampling-Expectation Algorithm

## Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

- Given the initial segmentation, they need to refine the models **and** labels to adapt better to the image.
- However, this is a chicken-and-egg problem. If we know the labels, we could compute the models, and if we knew the models, we could compute the best labels.
- They propose an EM algorithm for this. The basic idea is to alternate between estimating the probability that each pixel is of the two classes, and then given this probability to refine the underlying models.
- EM is guaranteed to converge (but only to a local minimum).

1. Initialize using the normalized KL-divergence.

1. Initialize using the normalized KL-divergence.
2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the 'M' step (because we have a non-parametric density).

1. Initialize using the normalized KL-divergence.
2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the 'M' step (because we have a non-parametric density).
3. Update the pixel assignment based on maximum likelihood (the 'E' step).

1. Initialize using the normalized KL-divergence.
2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).
3. Update the pixel assignment based on maximum likelihood (the ‘E’ step).
4. Repeat until stable.

1. Initialize using the normalized KL-divergence.
2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).
3. Update the pixel assignment based on maximum likelihood (the ‘E’ step).
4. Repeat until stable.

- One can use a hard assignment of the pixels and the kernel density estimator we’ve discussed, or a soft assignment of the pixels and then a weighted kernel density estimate (the weight is between the different classes).

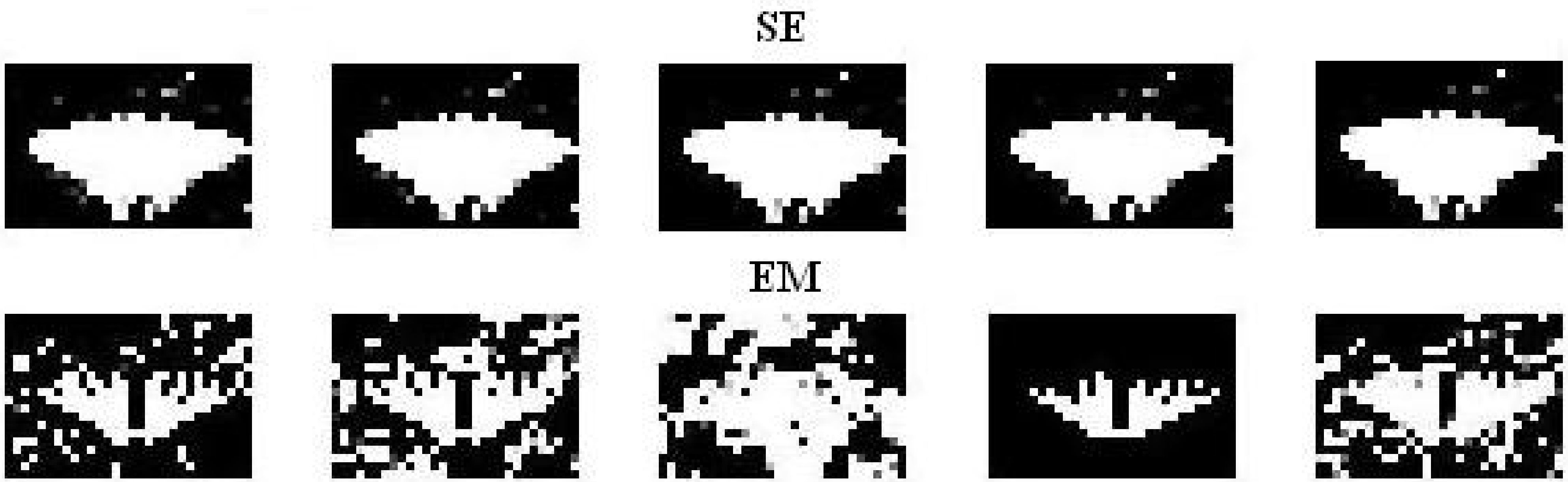
1. Initialize using the normalized KL-divergence.
2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the 'M' step (because we have a non-parametric density).
3. Update the pixel assignment based on maximum likelihood (the 'E' step).
4. Repeat until stable.

- One can use a hard assignment of the pixels and the kernel density estimator we've discussed, or a soft assignment of the pixels and then a weighted kernel density estimate (the weight is between the different classes).
- The overall probability of a pixel belonging to the foreground class

$$\hat{P}_{fg}(\mathbf{y}) = \frac{1}{Z} \sum_{i=1}^{n} \hat{P}_{fg}(\mathbf{x}_i) \prod_{j=1}^{d} K\left(\frac{y_j - x_{ij}}{\sigma_j}\right) \tag{35}$$

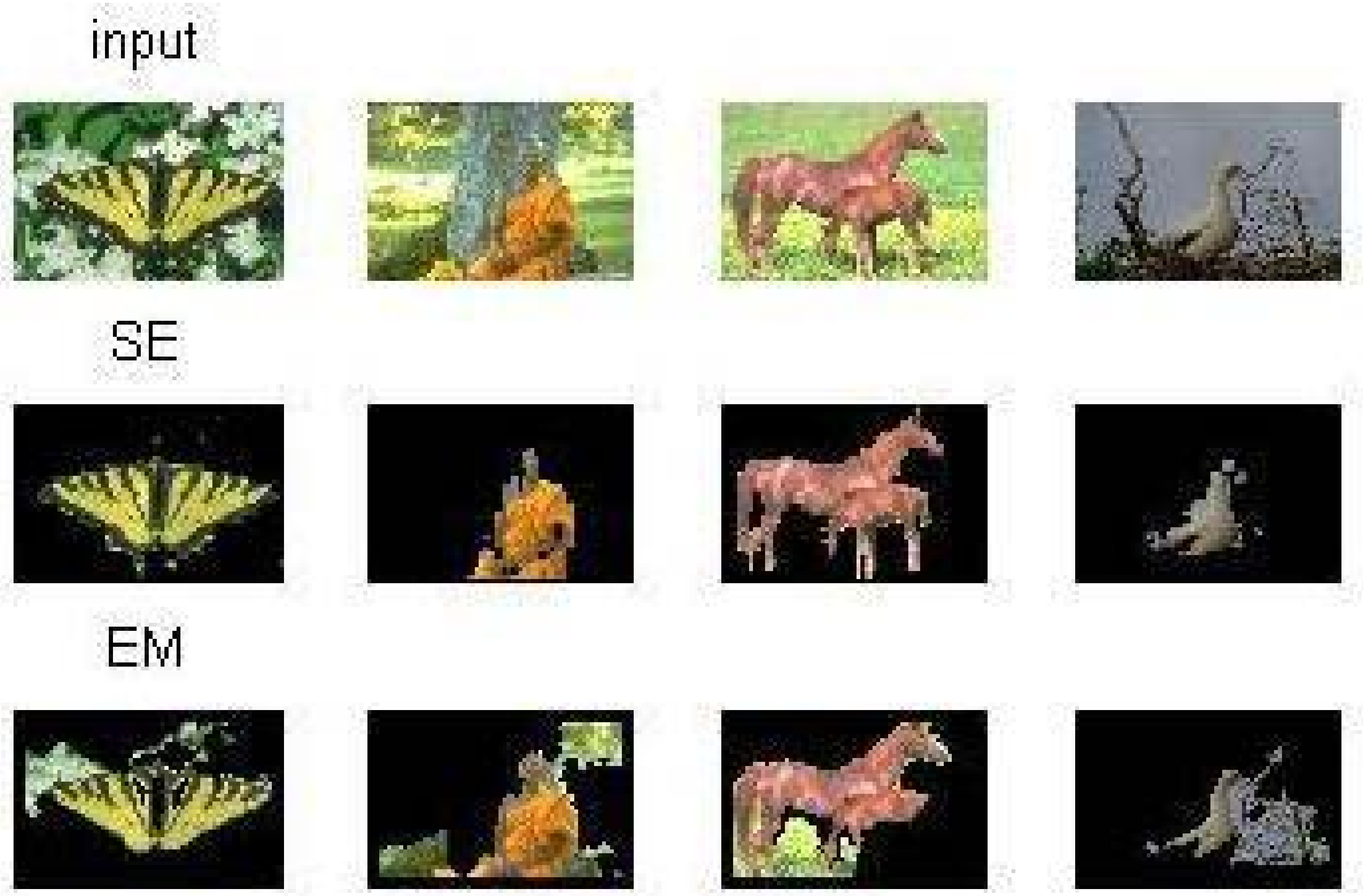
# Results: Stability

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**



# Results

**Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.**



# Results

## Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.