Chapter 4

Nonparametric Techniques

Bayes Theorem for Classification

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j) \cdot P(\omega_j)}{p(\mathbf{x})} \quad (1 \leq j \leq c) \quad \text{(Bayes Formula)}$$

To compute posterior probability $P(\omega_j|\mathbf{x})$, we need to know:

Prior probability: $P(\omega_j)$ Likelihood: $p(\mathbf{x}|\omega_j)$

- **Case I:** $p(\mathbf{x}|\omega_j)$ has certain **parametric form** $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$

Maximum-Likelihood (ML) Estimation

Bayesian Parameter Estimation

Bayes Theorem for Classification (Cont.)

Potential problems for Case I

The assumed parametric form may not fit the ground-truth density encountered in practice, e.g.:

Assumed parametric form: Unimodal (单峰, such as Gaussian pdf)

Ground-truth form: Multimodal (多峰)

- **Case II:** $p(\mathbf{x}|\omega_j)$ doesn't have **parametric form**

Let the data speak for themselves! → **Parzen Windows**, **k_n-nearest-neighbor**

Density Estimation

General settings

Feature space: $\mathcal{F} = \mathbf{R}^d$

Feature vector: $\mathbf{x} \in \mathcal{F}$

pdf function: $\mathbf{x} \sim p(\cdot)$

How to estimate $p(\mathbf{x})$ from the training examples?

Fundamental fact

The probability of a vector **x** **falling into a region** $\mathcal{R} \subset \mathcal{F}$:

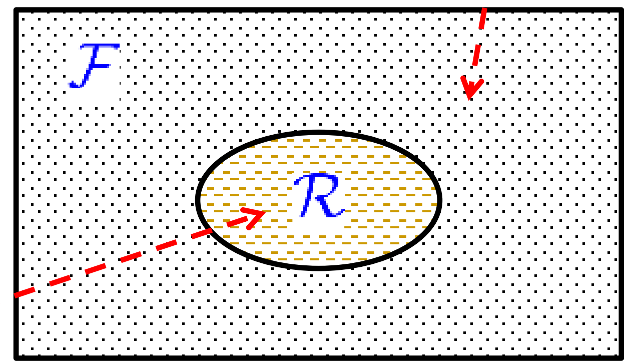
$$P = \Pr[\mathbf{x} \in \mathcal{R}] = \int_{\mathcal{R}} p(\mathbf{x}')\, d\mathbf{x}'$$

A smoothed/averaged version of $p(\mathbf{x})$

Density Estimation (Cont.)

$\Pr[\mathbf{x} \notin \mathcal{R}] = 1 - P$

$$P = \Pr[\mathbf{x} \in \mathcal{R}] = \int_{\mathcal{R}} p(\mathbf{x}')\, d\mathbf{x}'$$



$\Pr[\mathbf{x} \in \mathcal{R}] = P$

Given *n* examples (*i.i.d.*) $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ with $\mathbf{x}_i \sim p(\cdot)\ (1 \leq i \leq n)$

Let X be the (discrete) **random variable** representing **the number of examples** falling into $\mathcal{R}$

X will take Binomial distribution (二项分布): $X \sim \mathcal{B}(n, P)$

Density Estimation (Cont.)

$$\Pr[X = r] = \binom{n}{r} P^r (1-P)^{n-r} \quad (0 \leq r \leq n)$$

$X \sim \mathcal{B}(n, P)$

$$\mathcal{E}[X] = nP$$

Table 3.1 [pp.109]

$$\boxed{P = \frac{\mathcal{E}[X]}{n}}$$

Assume $\mathcal{R}$ is small

$p(\cdot)$ hardly varies within $\mathcal{R}$

$$P = \Pr[\mathbf{x} \in \mathcal{R}] = \int_{\mathcal{R}} p(\mathbf{x}')\, d\mathbf{x}'$$

$$\simeq p(\mathbf{x}) \int_{\mathcal{R}} 1\, d\mathbf{x}'$$ ($\mathbf{x}$ is a point within $\mathcal{R}$)

$$\boxed{P \simeq p(\mathbf{x})\, V}$$ (V is the volume enclosed by $\mathcal{R}$)

Density Estimation (Cont.)

$$P = \frac{\mathcal{E}[X]}{n}$$

$$P \simeq p(\mathbf{x})\, V$$

$$p(\mathbf{x}) = \frac{\mathcal{E}[X]/n}{V}$$

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

$X \sim \mathcal{B}(n, P)$

X **peaks sharply** about $\mathcal{E}[X]$ when *n* is large enough

Let *k* be the **actual value of *X*** after observing the *i.i.d.* training examples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$

$$k \simeq \mathcal{E}[X]$$

"Peaks Sharply"

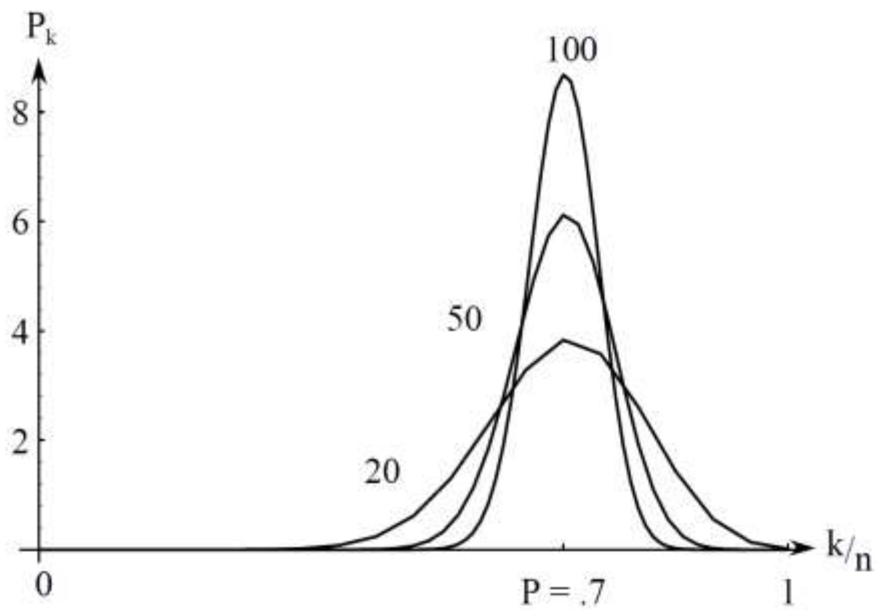


Figure 4.1: The probability P_k of finding k patterns in a volume where the space averaged probability is P as a function of k/n. Each curve is labelled by the total number of patterns n. For large n, such binomial distributions peak strongly at $k/n = P$ (here chosen to be 0.7).

Low of big numbers:

In probability theory, the law of large numbers is a mathematical theorem that states that the **average** of the results obtained from a large number of independent and identical random samples converges to the **true value**.

Density Estimation (Cont.)

$\mathcal{R} \Rightarrow \mathcal{R}_n$ (containing $\mathbf{x}$)

To show the explicit relationships with n:

$$p(\mathbf{x}) = \frac{k/n}{V} \Rightarrow p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$

Quantities:

V_n: volume of $\mathcal{R}_n$

n: # training examples

k_n: # training examples falling within $\mathcal{R}_n$

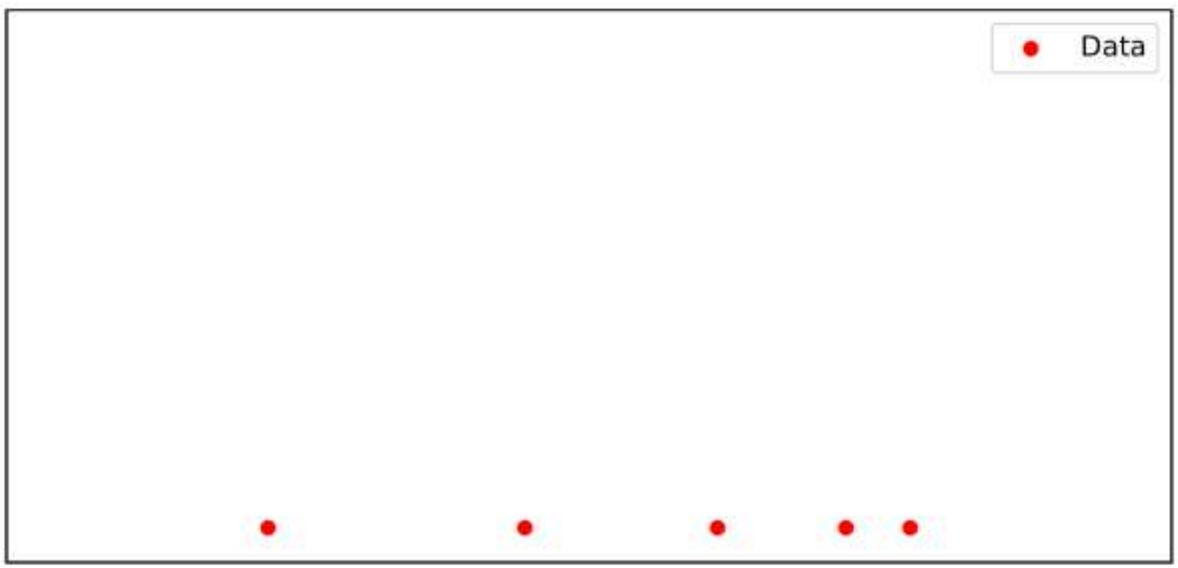
Fix V_n and determine k_n $\Rightarrow$ Parzen Windows

Fix k_n and determine V_n $\Rightarrow$ k_n-nearest-neighbor

KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

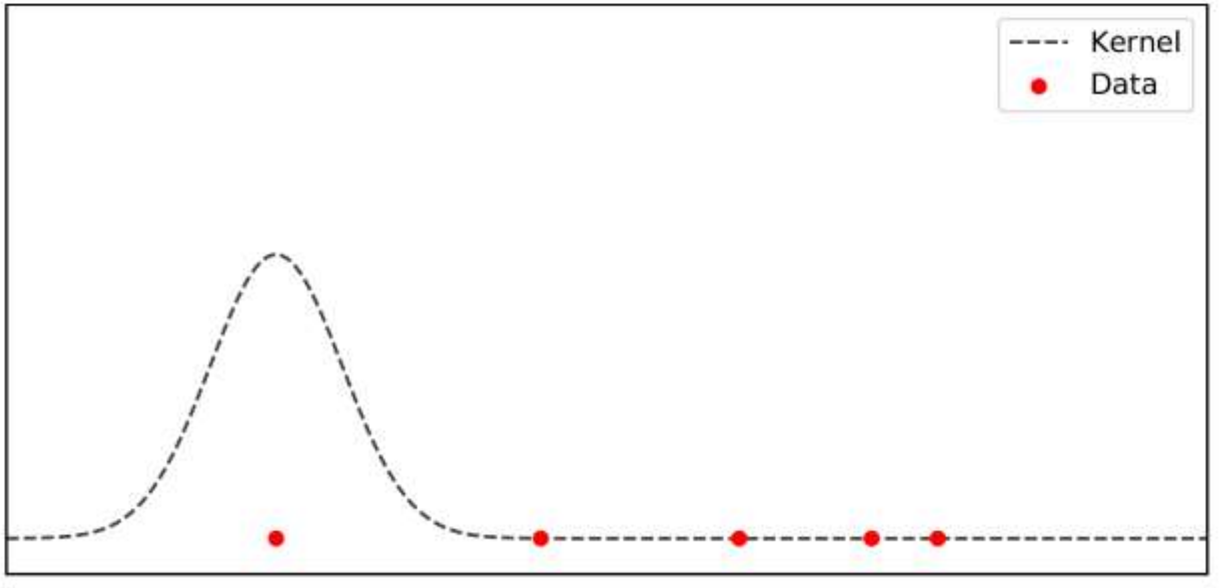
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

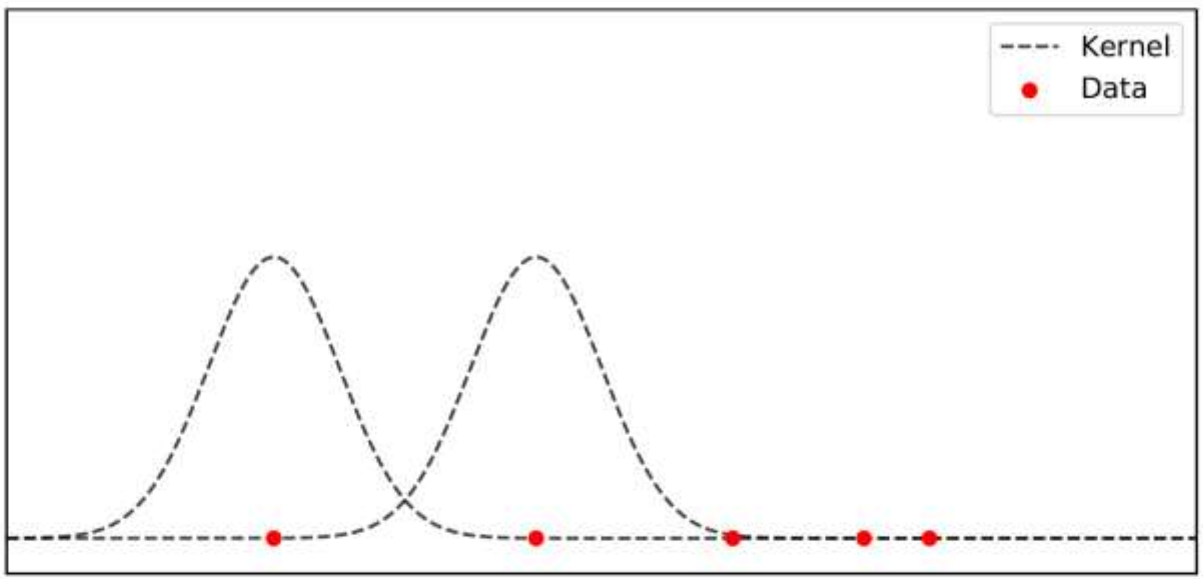
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

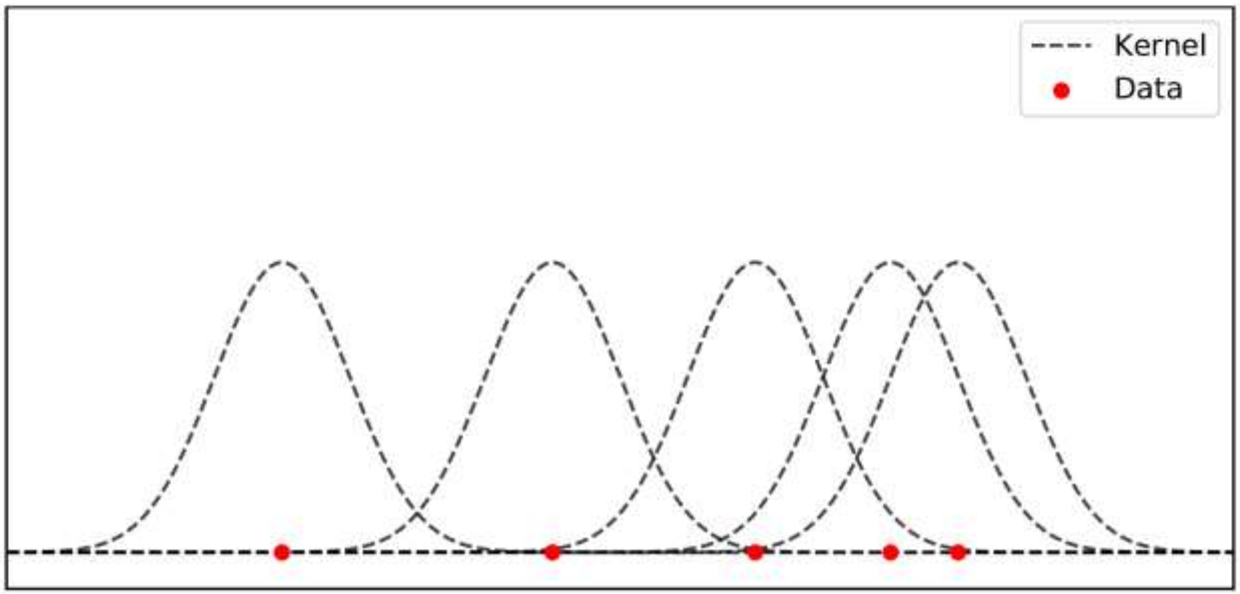
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

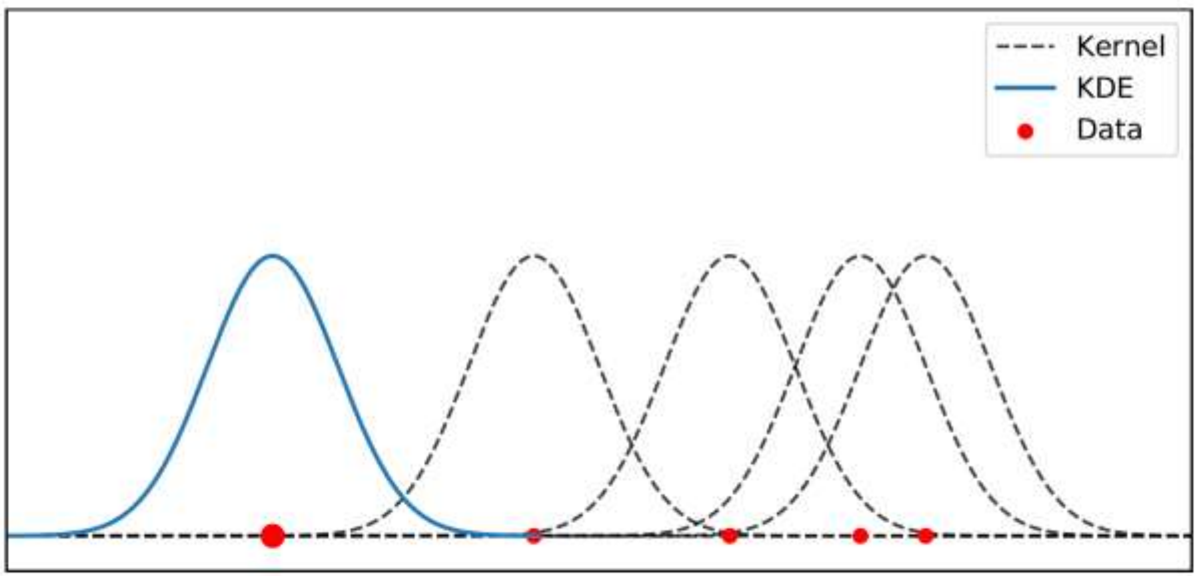
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

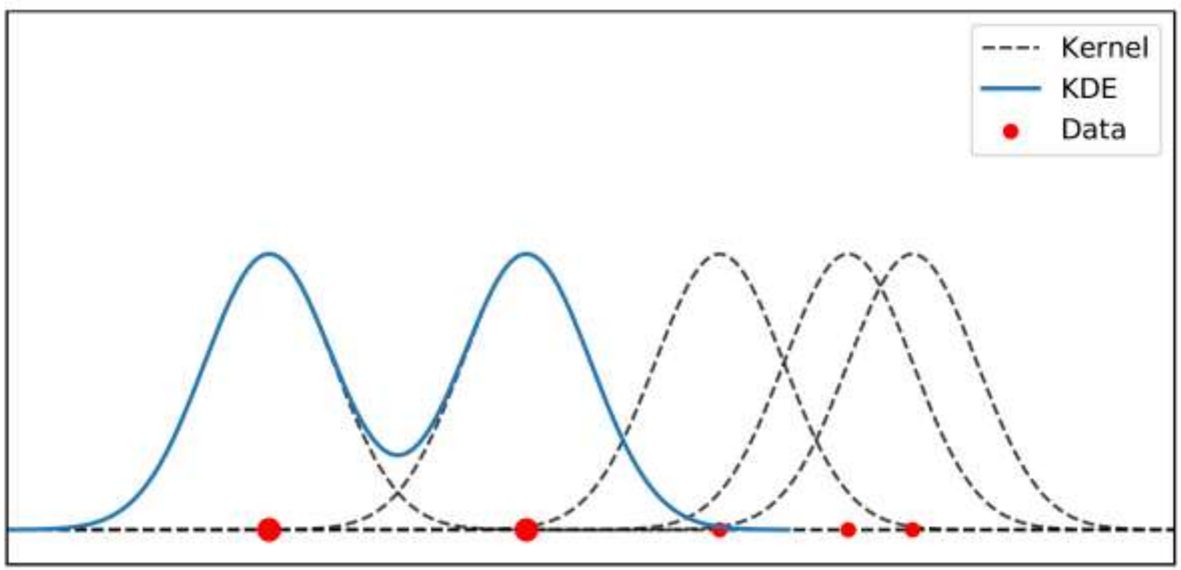
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

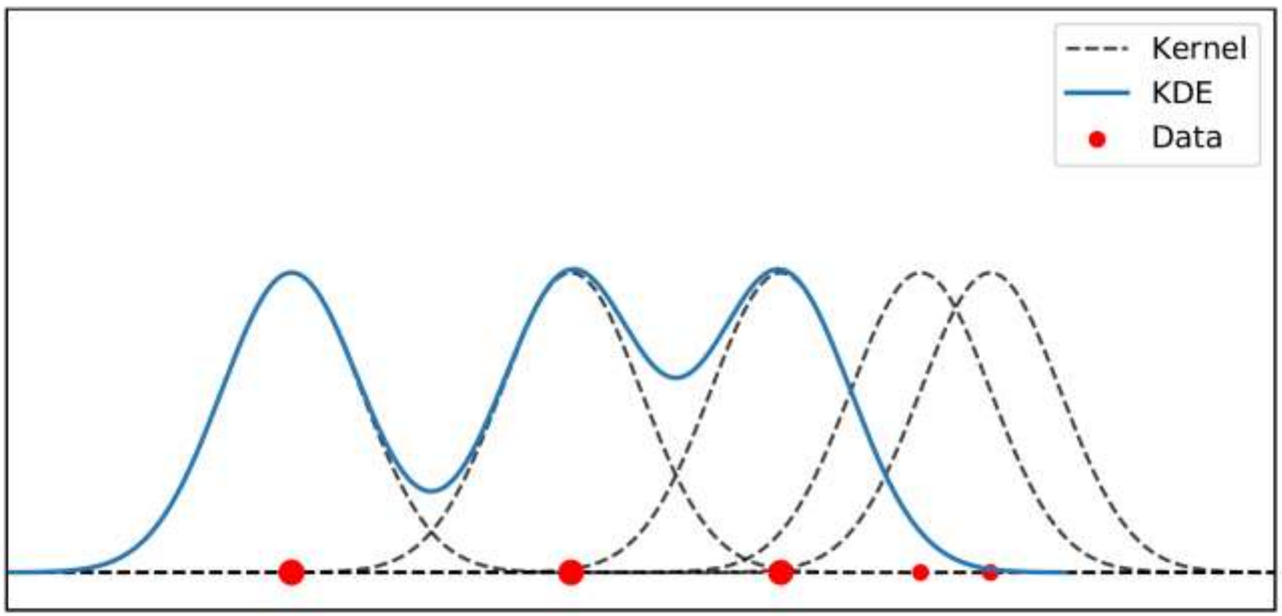
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

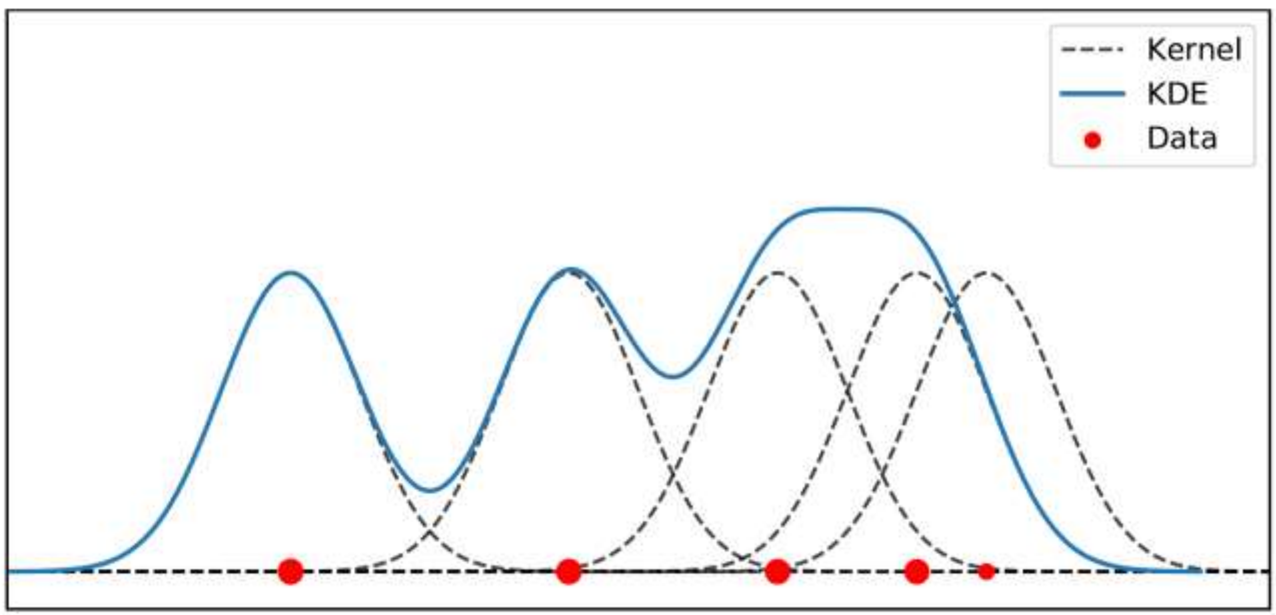
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

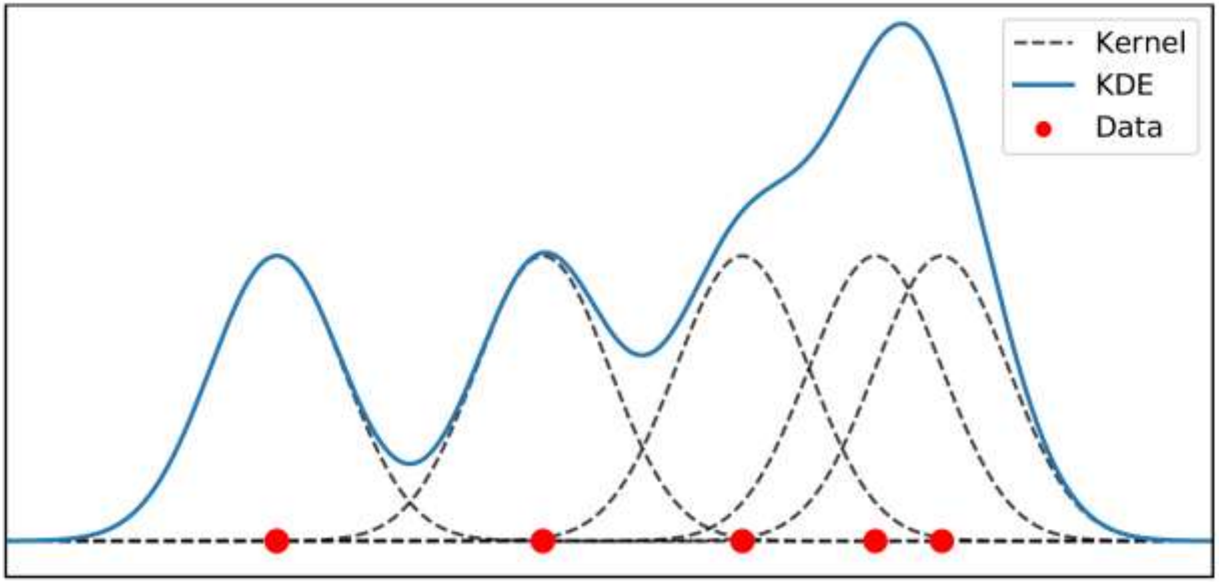
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



KDE – A Visual Example

On every data point x_i, we place a kernel function K. The kernel density estimate is

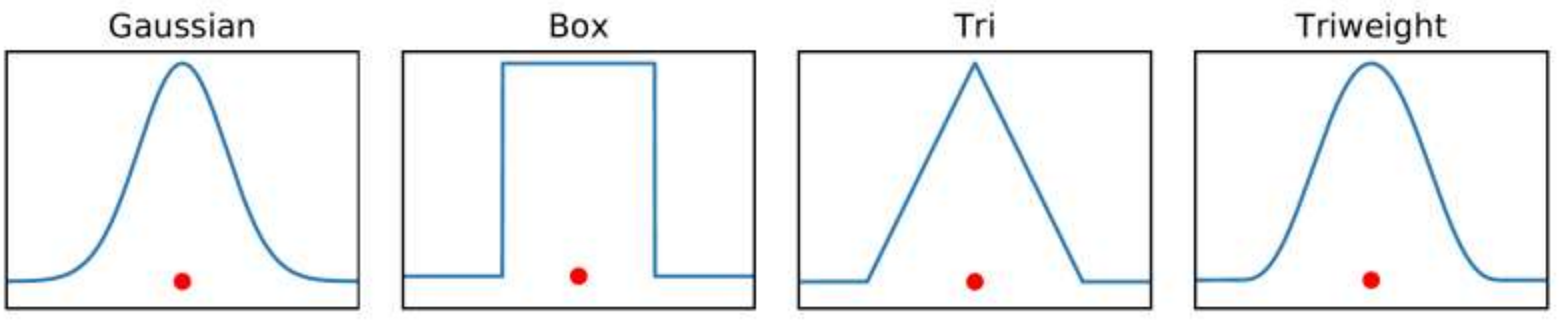
$$\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i).$$



Choice of Kernel Function

The kernel function K is typically

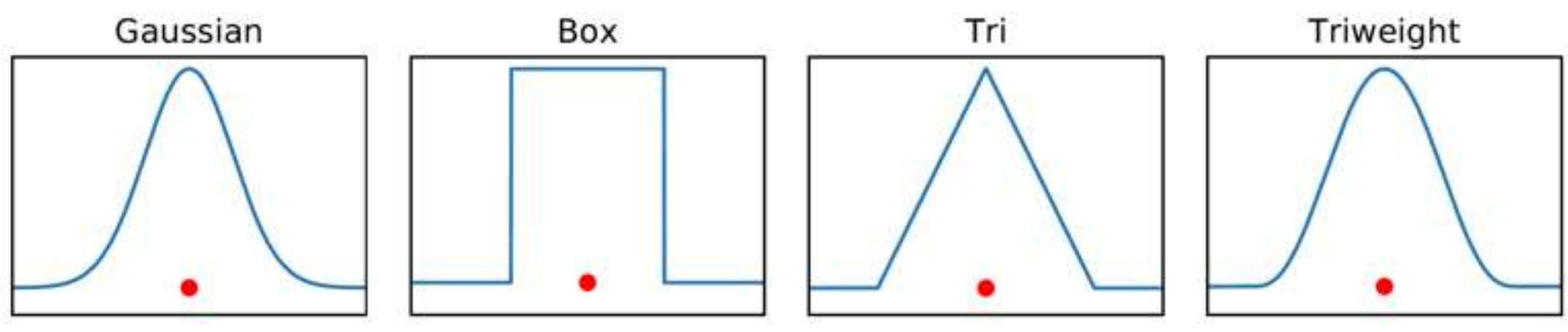
- everywhere non-negative: $K(x) \geq 0$ for every x
- symmetric: $K(x) = K(-x)$ for every x
- decreasing: $K'(x) \leq 0$ for every $x > 0$.



Choice of Kernel Function

The kernel function K is typically

- Everywhere non-negative $K(x) \geq 0$
- Symmetric: $K(x) = K(-x)$ for every x.



Parzen Windows

$$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$ Fix V_n, and then determine k_n

Assume $\mathcal{R}_n$ is a d-dimensional hypercube (超立方体)

The length of each edge is h_n

$$V_n = h_n^d$$

Determine k_n with window function (窗口函数), a.k.a. kernel function (核函数), potential function (势函数), etc.

Emanuel Parzen (1929-2016)

Parzen Windows (Cont.)

Window function: $\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq 1/2; \quad j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases}$

$\varphi(\mathbf{u})$ defines a **unit hypercube** centered at the origin

$\varphi\left(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\right) = 1 \iff$ $\mathbf{x}_i$ falls within the hypercube of volume V_n centered at $\mathbf{x}$

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

Parzen Windows (Cont.)

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

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

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

An average of functions of $\mathbf{x}$ and $\mathbf{x}_i$

$\varphi(\cdot)$ is not limited to be the hypercube window function of Eq.9 [pp.164]

$\varphi(\cdot)$ could be any pdf function:

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

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

Parzen Windows (Cont.)

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

$\varphi(\cdot)$ being a pdf function ⇒ $p_n(\cdot)$ being a pdf function

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

Integration by substitution (换元积分)

Let $\mathbf{u} = (\mathbf{x}-\mathbf{x}_i)/h_n$

$$= \frac{1}{nV_n}\sum_{i=1}^{n}\int h_n^d\,\varphi(\mathbf{u})\,d(\mathbf{u}) = \frac{1}{n}\sum_{i=1}^{n}\int \varphi(\mathbf{u})\,d(\mathbf{u}) = 1$$

雅可比行列式

window function (being pdf) $\varphi(\cdot)$ + window width h_n + training data $\mathbf{x}_i$ ⇒ Parzen pdf $p_n(\cdot)$

Parzen Windows (Cont.)

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

$\varphi(\cdot)$ being a pdf function $\Rightarrow$ $p_n(\cdot)$ being a pdf function

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

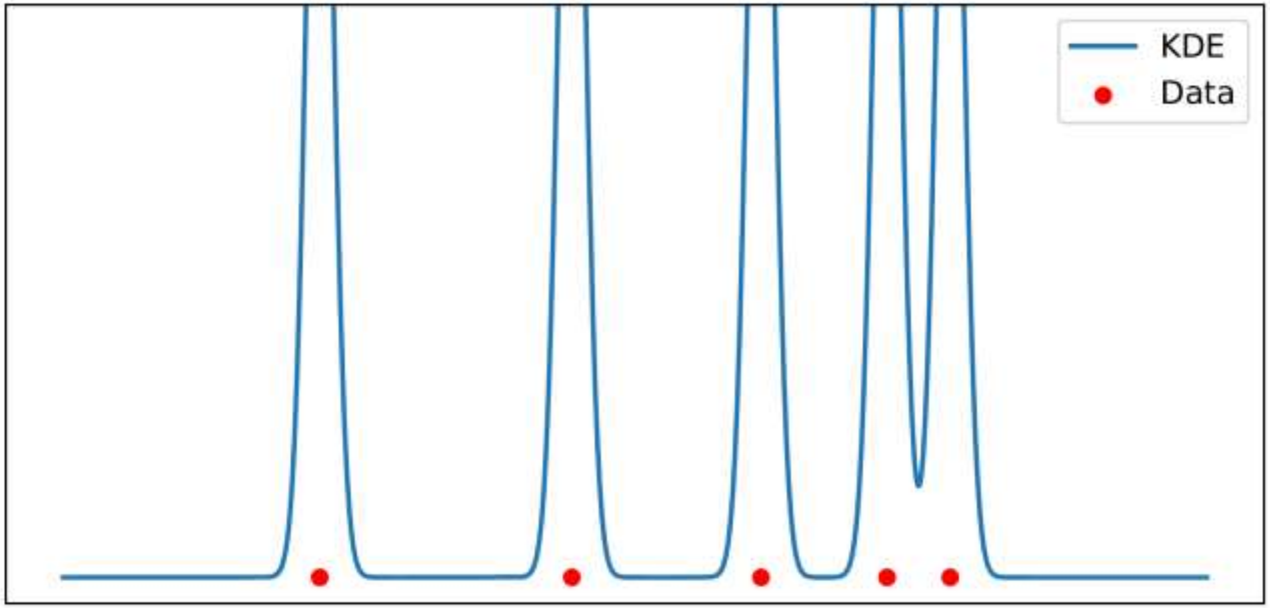
What is the effect of h_n ("window width") on the Parzen pdf?

- $p_n(\mathbf{x})$: **superposition** (叠加) of n interpolations (插值)
- $\mathbf{x}_i$: contributes to $p_n(\mathbf{x})$ based on its "**distance**" from $\mathbf{x}$ (i.e. "$\mathbf{x}$-$\mathbf{x}_i$")

Window Width – An Example

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

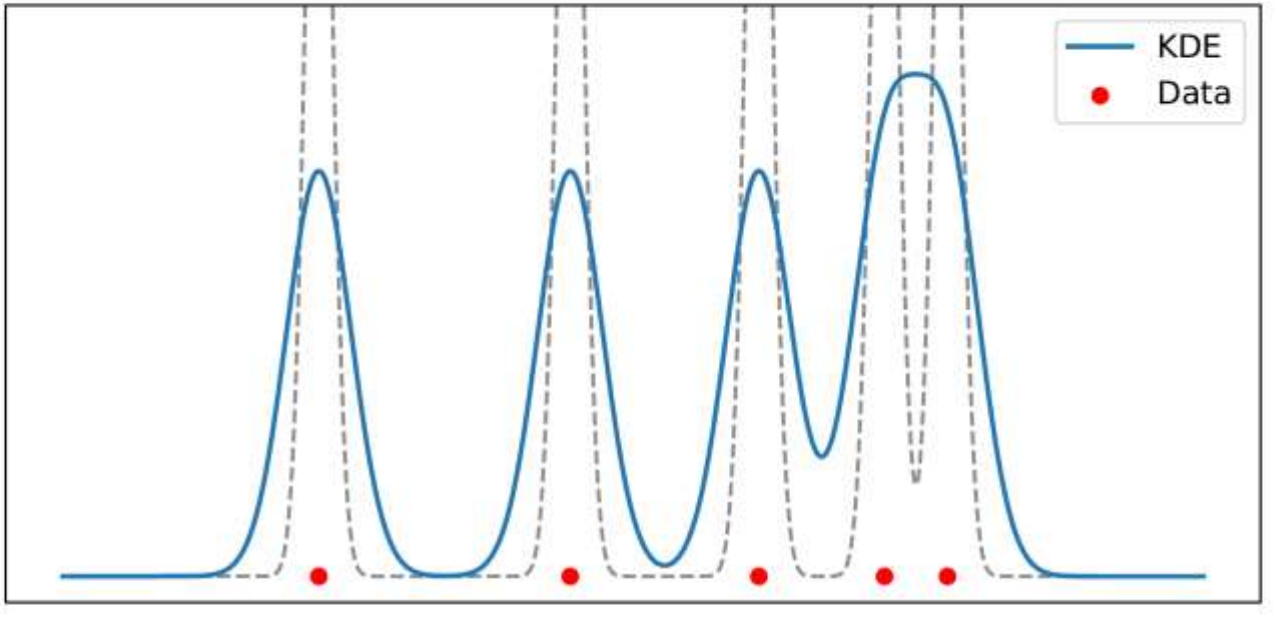
$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Window Width – An Example

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

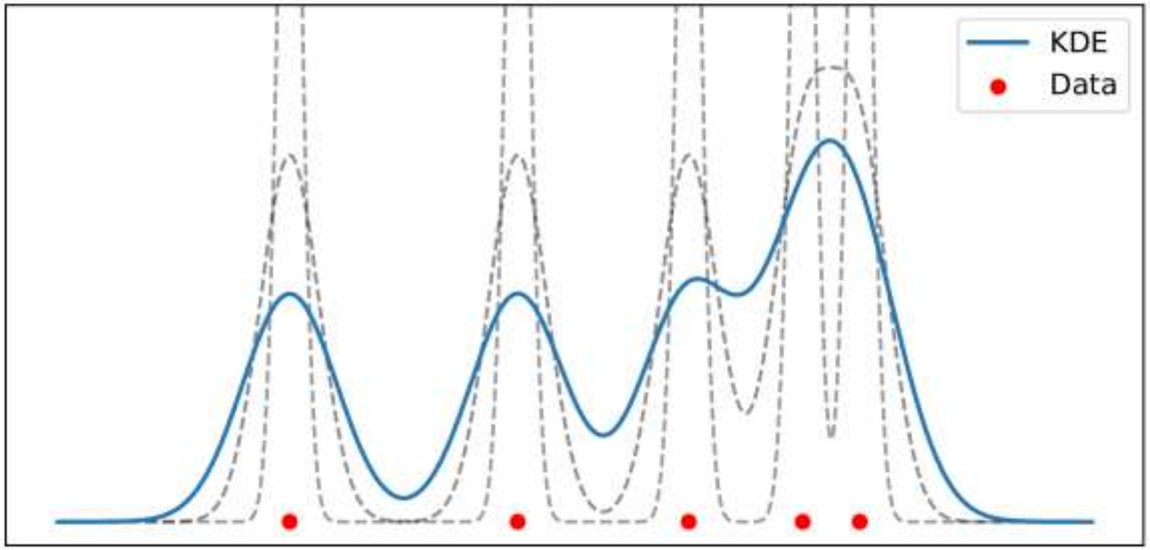
$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Window Width – An Example

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

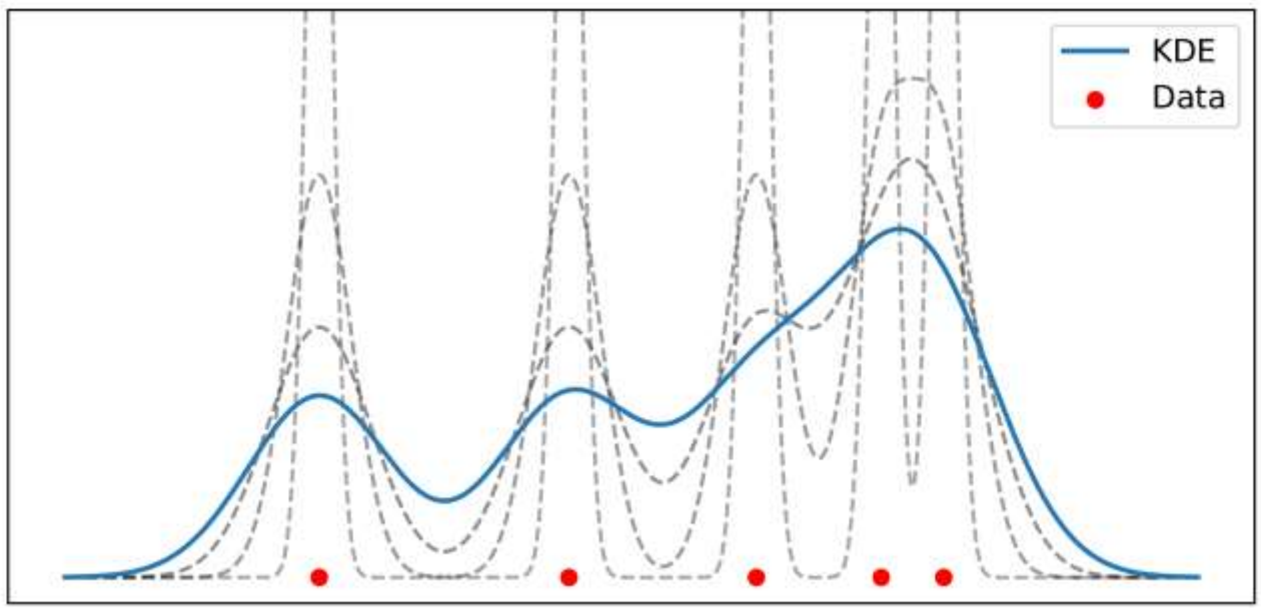
$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x-x_i}{h}\right).$$



Window Width – An Example

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

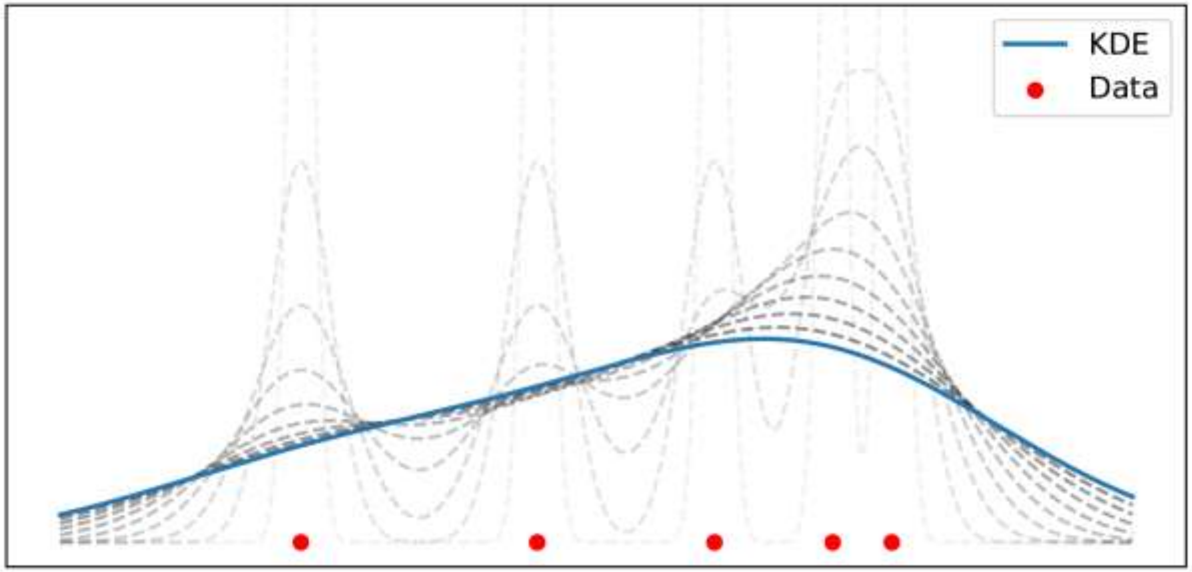
$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Window Width – An Example

We use h to control for the *bandwidth* of $\hat{f}(x)$ by writing

$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right).$$



Parzen Windows (Cont.)

The effect of h_n ("window width")

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

What do "amplitude" and "width" mean for a function?

Affects the *amplitude* (vertical scale, 幅度)

Affects the *width* (horizontal scale, 宽度)

For $\varphi(\mathbf{u})$:

$|\varphi(\mathbf{u})| \leq a$ *(amplitude)* ⇒ $|\delta_n(\mathbf{x})| \leq (1/h_n^d) \cdot a$

$|u_j| \leq b_j$ *(width)* $(j = 1, \ldots, d)$ ⇒ $|x_j| \leq h_n \cdot b_j \;\; (j = 1, \ldots, d)$

For $\delta_n(\mathbf{x})$:

Parzen Windows (Cont.)

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

$\delta_n(\cdot)$ being a pdf function

$$\int \delta_n(\mathbf{x})\, d\mathbf{x} = \int \frac{1}{h_n^d}\varphi\left(\frac{\mathbf{x}}{h_n}\right)\, d\mathbf{x}$$

Integration by substitution

Let $\mathbf{u} = \mathbf{x}/h_n$

$$= \int \frac{1}{h_n^d} \cdot \varphi(\mathbf{u}) \cdot h_n^d\, d\mathbf{u} = \int \varphi(\mathbf{u})\, d\mathbf{u} = 1$$

Case I

$h_n \uparrow$ ⇒ Function amplitude $\downarrow$ (vertical scale)

$h_n \uparrow$ ⇒ Function width $\uparrow$ (horizontal scale)

Case II

$h_n \downarrow$ ⇒ Function amplitude $\uparrow$ (vertical scale)

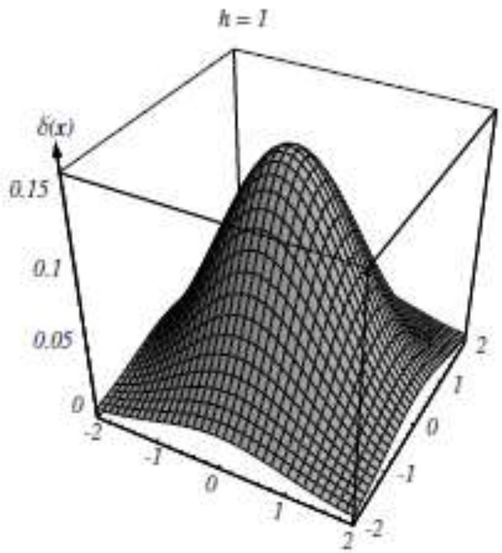
$h_n \downarrow$ ⇒ Function width $\downarrow$ (horizontal scale)

Parzen Windows (Cont.)

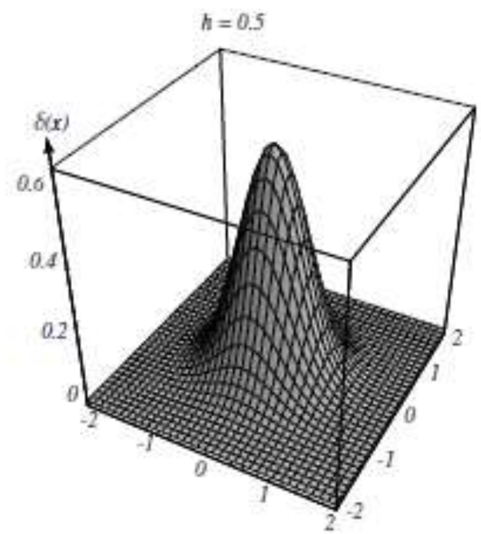
$$\delta_n(\mathbf{x}) = \frac{1}{h_n^d}\varphi\left(\frac{\mathbf{x}}{h_n}\right)$$

Suppose $\varphi(\cdot)$ being a 2-d *Gaussian pdf*

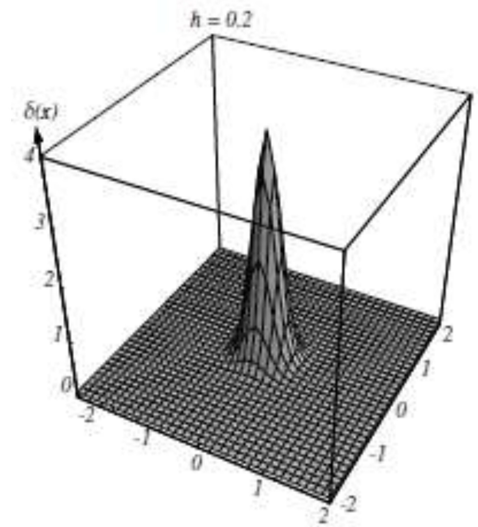
The shape of $\delta_n(\mathbf{x})$ with decreasing values of h_n



h=1.0



h=0.5



h=0.2

Parzen Windows (Cont.)

$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(\mathbf{x}-\mathbf{x}_i), \text{ where } \delta_n(\mathbf{x}) = \frac{1}{h_n^d}\varphi\left(\frac{\mathbf{x}}{h_n}\right)$$

- h_n very large ➔ $\delta_n(\mathbf{x})$ being *broad* with *small amplitude*

  $p_n(\mathbf{x})$ will be the superposition of n broad, slowly changing (慢变) functions, i.e. being *smooth* (平滑) with *low resolution* (低分辨率)

- h_n very small ➔ $\delta_n(\mathbf{x})$ being *sharp* with *large amplitude*

  $p_n(\mathbf{x})$ will be the superposition of n sharp pulses (尖脉冲), i.e. being *variable/unstable* (易变) with *high resolution* (高分辨率)

A compromised value (折衷值) of h_n should be sought for limited number of training examples

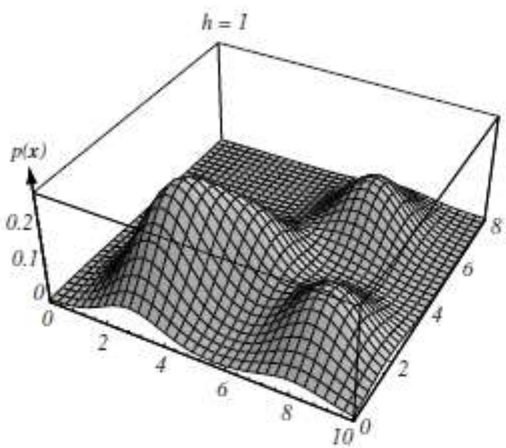
Parzen Windows (Cont.)

More illustrations:
Subsection 4.3.3 [pp.168]

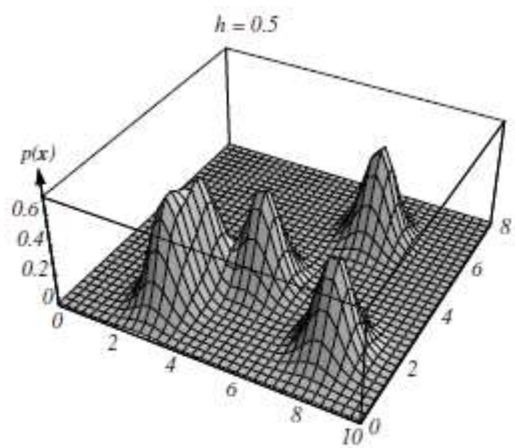
$$p_n(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(\mathbf{x}-\mathbf{x}_i), \text{ where } \delta_n(\mathbf{x}) = \frac{1}{h_n^d}\varphi\left(\frac{\mathbf{x}}{h_n}\right)$$

Suppose $\varphi(\cdot)$ being a 2-d *Gaussian pdf* and *n*=5

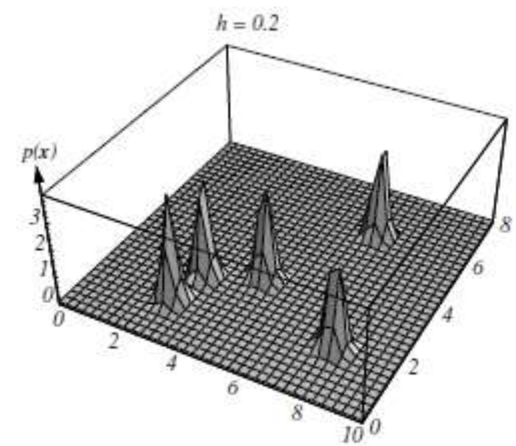
The shape of $p_n(\mathbf{x})$ with decreasing values of h_n



h=1.0



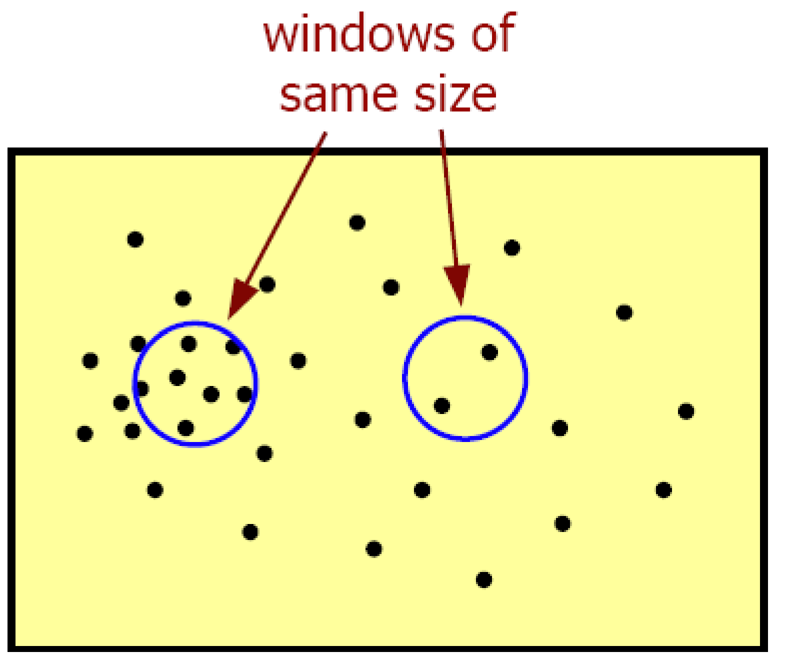
h=0.5



h=0.2

k_n-Neareast-Neighbor

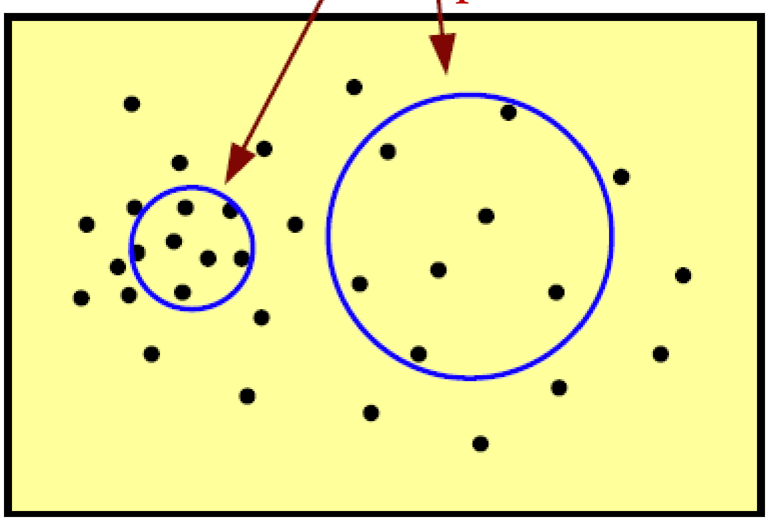
- If the distribution of $p(x)$ is non-uniform, using the same width of window in the entire feature space may not get satisfactory results.



k_n-Neareast-Neighbor

- A method to solve the problem of Parzen window estimation with fixed window width
  - The window width is not fixed, but the number of samples k around x is fixed.
  - k is denoted as k_n , because it usually depends on the total number of samples n
  - The density around x is large, the window width becomes smaller (high resolution)
  - The density around x is small, the window width becomes larger (low resolution)
  - The k_n samples included by the window are called the k_n nearest neighbors of x

The same number of samples in the window



k_n-Neareast-Neighbor

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

Fix k_n, and then determine V_n

specify k_n ➔ center a cell about **x** ➔ grow the cell until capturing k_n *nearest examples* ➔ return cell volume as V_n

The principled rule to specify k_n [pp.175]

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

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

A rule-of-thumb choice for k_n:

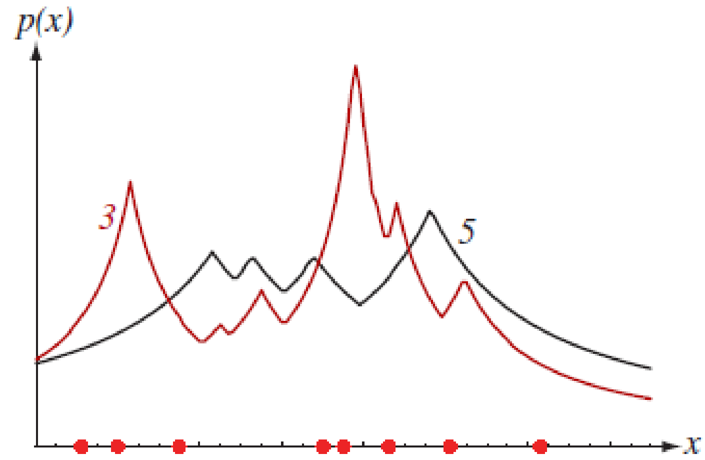
$$k_n = \sqrt{n}$$

k_n-Neareast-Neighbor (Cont.)

Eight points in one dimension (n=8, d=1)

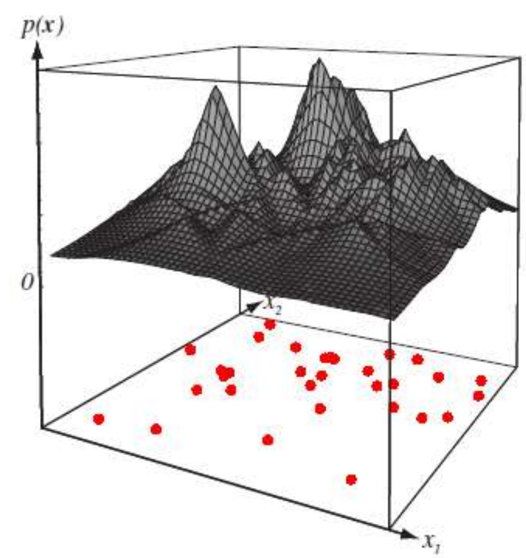
red curve: k_n=3

black curve: k_n=5



Thirty-one points in two dimensions (n=31, d=2)

black surface: k_n=5



Related Topic

Nearest Neighbor Rule &

Distance Metric

Nearest-Neighbor (NN) Rule (最近邻准则)

Classification with nearest-neighbor rule

Given the label space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ and a set of *n* labeled training examples $\mathcal{D}^n = \{(\mathbf{x}_i, \theta_i) \mid 1 \leq i \leq n\}$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $\theta_i \in \Omega$

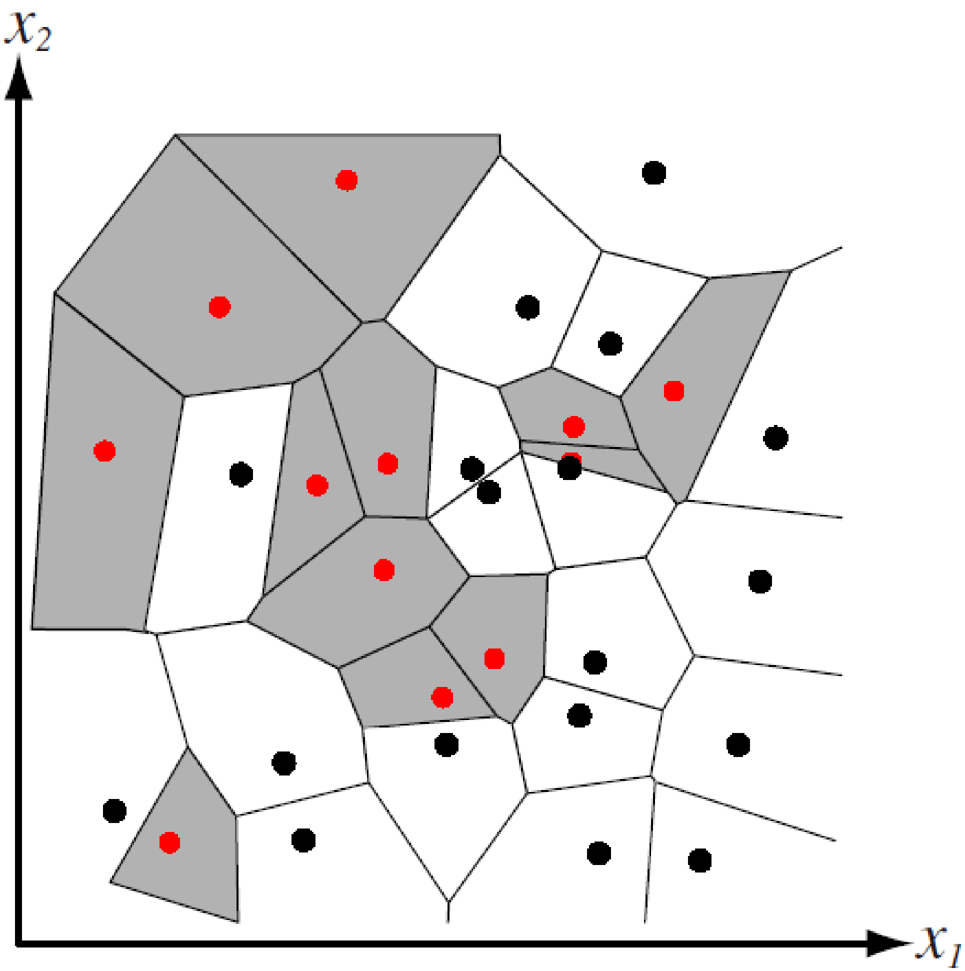
for test example $\mathbf{x}$, *identify* $\mathbf{x}' = \mathrm{argmin}_{\mathbf{x}_i \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}} D(\mathbf{x}_i, \mathbf{x})$ *and then assign the label* θ' *associated with* $\mathbf{x}'$ *to* $\mathbf{x}$

$D(\mathbf{a}, \mathbf{b})$: *distance metric between two vectors* **a** *and* **b**, *e.g. the Euclidean distance*

Basic assumption: $P(\omega_i \mid \mathbf{x}') \simeq P(\omega_i \mid \mathbf{x})$

as $n \to \infty$

Voronoi tessellation (维诺图)



Each training example **x** leads to a cell in the Voronoi tessellation

- *any point in the cell is closer to* **x** *than to any other training examples*
- *partition the feature space into n cells*
- *any point in the cell shares the same class label as* **x**

Error Rate of Nearest Neighbor Rule

$P(e \mid \mathbf{x})$: The probability of making an erroneous classification on **x** based on nearest-neighbor rule

$P(e)$: The **average probability of error** based on nearest-neighbor rule: $P(e) = \int P(e \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$

$P^*(e \mid \mathbf{x})$: The **minimum** possible value of $P(e \mid \mathbf{x})$, i.e. the one given by *Bayesian decision rule*: $P^*(e \mid \mathbf{x}) = 1 - \max_{1 \le i \le c} P(\omega_i \mid \mathbf{x})$

$P^*(e)$: The **Bayes risk** (*under zero-one loss*): $P^*(e) = \int P^*(e \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x}$

Error bounds of nearest neighbor rule

$$P^*(e) \le P(e) \le P^*(e) \left(2 - \frac{c}{c-1} P^*(e) \right)$$

(c: # class labels)

k-Nearest-Neighbor (kNN) Rule (k-近邻准则)

Classification with k-nearest-neighbor rule

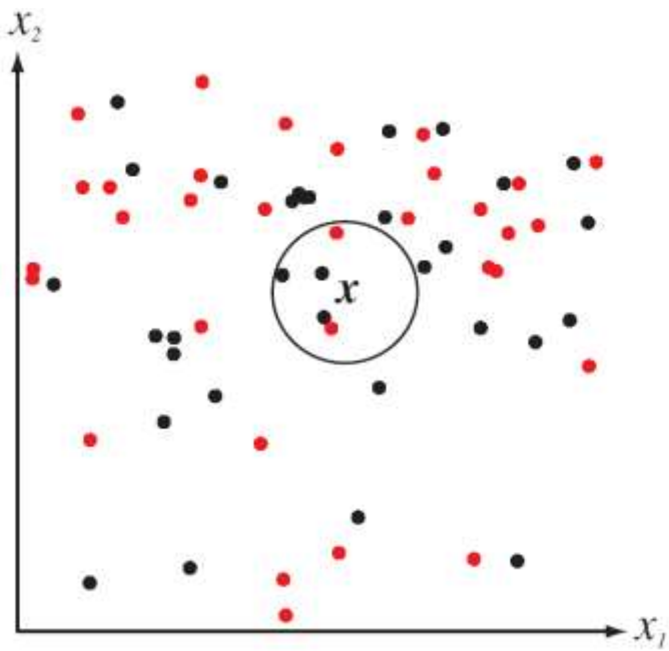
Given the label space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$ and a set of n labeled training examples $\mathcal{D}^n = \{(\mathbf{x}_i, \theta_i) \mid 1 \le i \le n\}$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $\theta_i \in \Omega$

for test example $\mathbf{x}$, *identify* $S' = \{\mathbf{x}_i \mid \mathbf{x}_i \text{ is among the } k\text{NN of } \mathbf{x}\}$ *and then assign the most frequent label w.r.t.* S', *i.e.* $\arg\max_{\omega_i \in \Omega} \sum_{\mathbf{x}_i \in S'} 1_{\theta_i = \omega_i}$ *to* $\mathbf{x}$.

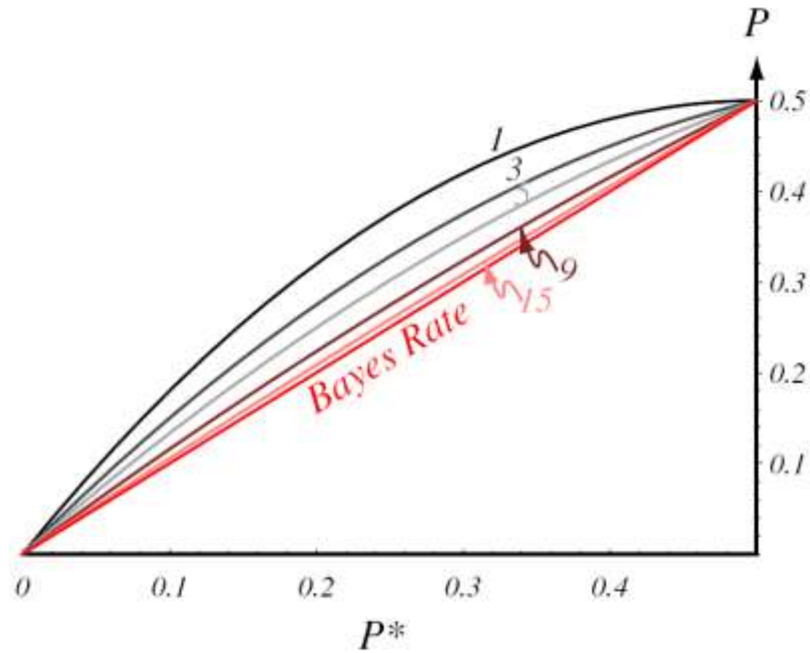
1_π : *an indicator function which returns 1 if predicate* π *holds, and 0 otherwise*

For binary classification problem (c=2), an odd value of k is generally used to avoid ties

k-Nearest-Neighbor (kNN) Rule (Cont.)



c=2, k=5



the error rate of kNN rule (i.e. P) lower-bounded by the Bayes risk (i.e. P*) for binary classification (c=2)

Computational Complexity of *k*NN Rule

Given *n* labeled training examples in *d*-dimensional feature space, the computational complexity of classifying one test example is $O(dn)$

General ways of reducing computational burden

- **Partial distance:** $D_r(\mathbf{a}, \mathbf{b}) = \left(\sum_{j=1}^{r} (a_j - b_j)^2\right)^{\frac{1}{2}} \quad (r < d)$
- **Pre-structuring:**

  *create some form of **search tree**, where nearest neighbors are recursively identified following the tree structure*

- **Editing/Pruning/Condensing:**

  *eliminate **"redundant"** ("useless") examples from the training set, e.g. example surrounded by training examples of the same class label*

Properties of Distance Metric

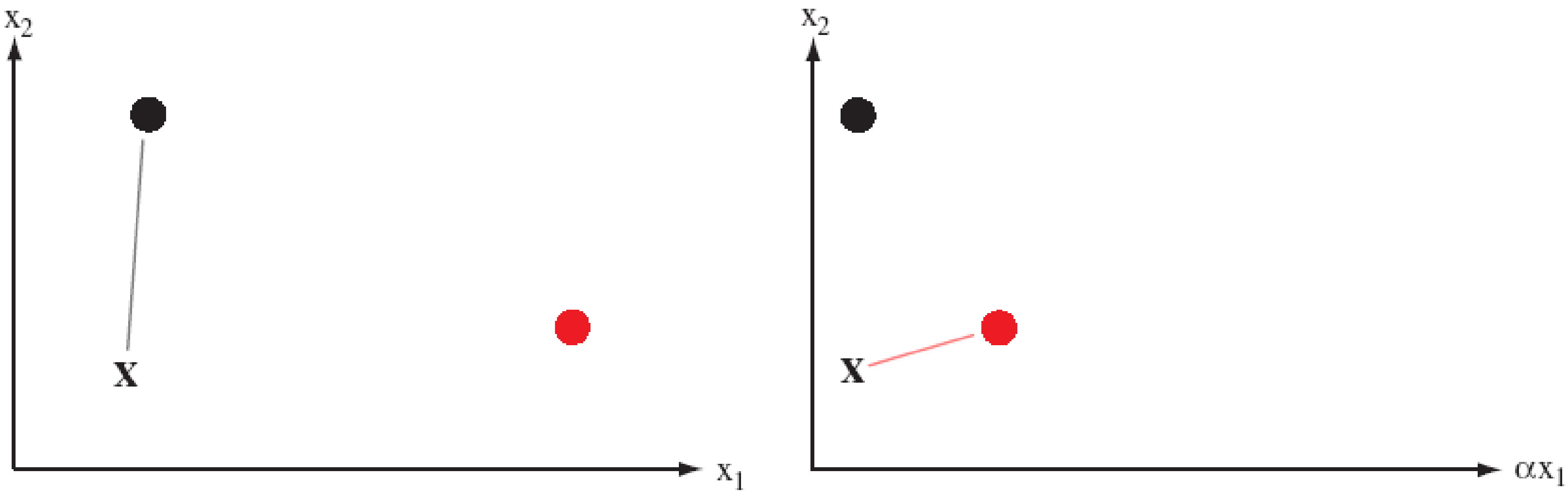
The NN/kNN rule depends on the use of distance metric to identify nearest neighbors

Four properties of distance metric

- **non-negativity:** $D(\mathbf{a}, \mathbf{b}) \geq 0$
  非负性
- **reflexivity:** $D(\mathbf{a}, \mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$
  自反性
- **symmetry:** $D(\mathbf{a}, \mathbf{b}) = D(\mathbf{b}, \mathbf{a})$
  对称性
- **triangle inequality:** $D(\mathbf{a}, \mathbf{b}) + D(\mathbf{b}, \mathbf{c}) \geq D(\mathbf{a}, \mathbf{c})$
  三角不等式

Potential Issue of Euclidean Distance

$$D(\mathbf{a}, \mathbf{b}) = \left(\sum_{j=1}^{d} (a_j - b_j)^2\right)^{\frac{1}{2}}$$ (Euclidean distance)



Scaling the features ➔ change the distance relationship

Possible solution: normalize each feature into equal-sized intervals, e.g. [0, 1]

Minkowski Distance Metric

$$L_k(\mathbf{a},\mathbf{b}) = \left(\sum_{j=1}^{d} |a_j - b_j|^k\right)^{\frac{1}{k}} \quad (k > 0)$$

(a.k.a. L_k norm)

- *k*=2: *Euclidean* distance
- *k*=1: *Manhattan* distance (*city block* distance)

$$L_1(\mathbf{a},\mathbf{b}) = \sum_{j=1}^{d} |a_j - b_j|$$

- *k*= ∞ : L_∞ distance

$$L_\infty(\mathbf{a},\mathbf{b}) = \max_{1\le j\le d} \ |a_j - b_j|$$

Distance Metric Between Sets

Tanimoto distance

$$D_{Tanimoto}(S_1, S_2) = \frac{n_1 + n_2 - 2n_{12}}{n_1 + n_2 - n_{12}}$$

$$(n_1 = |S_1|,\ n_2 = |S_2|,\ n_{12} = |S_1 \cap S_2|)$$

Example: treat each word as a set of characters

Which word out of 'cat', 'pots' and 'patches' mostly resembles 'pat'?

cat

$$S_1 = \{p, a, t\}$$

$$S_2 = \{c, a, t\}$$

$$S_3 = \{p, o, t, s\}$$

$$S_4 = \{p, a, t, c, h, e, s\}$$

$$D_{Tanimoto}(S_1, S_2) = \frac{3+3-2*2}{3+3-2} = 0.5$$

$$D_{Tanimoto}(S_1, S_3) = \frac{3+4-2*2}{3+4-2} = 0.6$$

$$D_{Tanimoto}(S_1, S_4) = \frac{3+7-2*3}{3+7-3} = 0.571$$

Distance Metric Between Sets (Cont.)

Hausdorff distance

$$D_H(S_1, S_2) = \max\left(\max_{\mathbf{s}_1 \in S_1} \min_{\mathbf{s}_2 \in S_2} D(\mathbf{s}_1, \mathbf{s}_2), \max_{\mathbf{s}_2 \in S_2} \min_{\mathbf{s}_1 \in S_1} D(\mathbf{s}_2, \mathbf{s}_1)\right)$$

($D(\mathbf{s}_1, \mathbf{s}_2)$: any distance metric between $\mathbf{s}_1$ and $\mathbf{s}_2$)

Example: Hausdorff distance between two sets of feature vectors

$$S_1 = \{(0.1, 0.2)^t,\ (0.3, 0.8)^t\} \qquad S_2 = \{(0.5, 0.5)^t,\ (0.7, 0.3)^t\}$$

$$\begin{aligned} D_H(S_1, S_2) &= \max\left(\max(0.5, 0.36), \max(0.36, 0.61)\right) \\ &= \max\left(0.5, 0.61\right) \\ &= 0.61 \end{aligned}$$

Summary

- Basic settings for nonparametric techniques
  - Let the data speak for themselves
    - Parametric form not assumed for class-conditional pdf
  - Estimate class-conditional pdf from training examples

    ➔ Make predictions based on Bayes Formula
- Fundamental result in density estimation

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

n: # training examples

V_n: volume of region $\mathcal{R}_n$ containing $\mathbf{x}$

k_n: # training examples falling within $\mathcal{R}_n$

Summary (Cont.)

- Parzen Windows: Fix V_n ➔ Determine k_n
  - Effect of h_n (window width): A compromised value for a fixed number of training examples should be chosen

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

$\varphi(\cdot)$ being a pdf function ➔ $p_n(\cdot)$ being a pdf function

window function (being pdf) $\varphi(\cdot)$ + window width h_n + training data $\mathbf{x}_i$ ➔ Parzen pdf $p_n(\cdot)$

Summary (Cont.)

- k_n-nearest-neighbor: Fix k_n ➔ Determine V_n

specify k_n ➔ center a cell about **x** ➔ grow the cell until capturing k_n *nearest examples* ➔ return cell volume as V_n

The principled rule to specify k_n [pp.175]

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

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

A rule-of-thumb choice for k_n: $k_n = \sqrt{n}$

Summary (Cont.)

- Nearest neighbor (NN) rule & distance metric
  - Classification with NN rule: Voronoi tessellation
  - Error bounds of NN rule w.r.t. Bayes risk

$$P^*(e) \leq P(e) \leq P^*(e)\left(2 - \frac{c}{c-1}P^*(e)\right)$$

  - Classification with kNN rule
  - Reducing computational complexity
    - *Partial distance, pre-structuring, Editing/Pruning/Condensing*
  - Distance metric
    - *non-negativity, reflexivity, symmetry, triangle inequality*
    - *Minkowski distance, Tanimoto distance, Hausdorff distance*