Chapter 4: Nonparametric Techniques

Mingmin Chi

SCS Fudan University, Shanghai, China

Generative vs. Discriminative

There are two schools of thought in ML/PR communities:

1. Generative:
   - Estimate class models from data
   - Compute the discriminative function
   - Plug in your data - get the answer
2. Discriminative:
   - Estimate the discriminative function
   - Plug in your data - get the answer

Density Estimation

Density estimation is at the core of generative pattern recognition

$$P(a < x < b) = \int_a^b p(x)dx$$

$$\text{mean: } E[x] = \int xp(x)dx$$

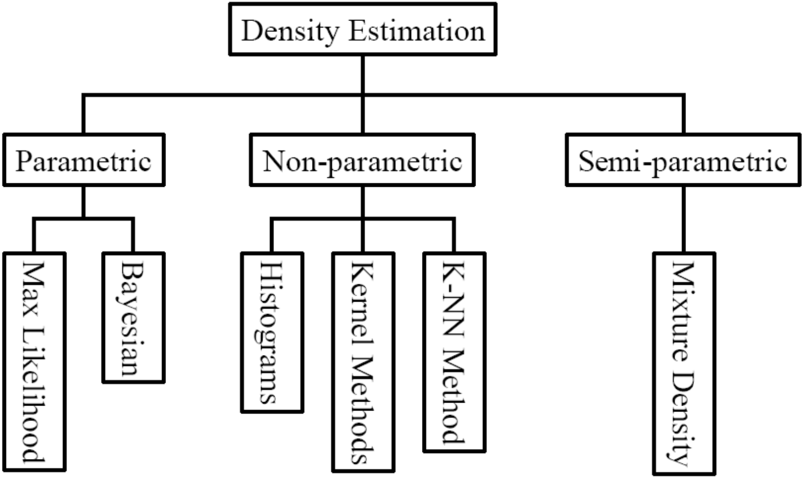
$$\begin{aligned}\text{covariance: } & E[(x - E[x])(x - E[x])^\top] \\ & = \int (x - E[x])(x - E[x])^\top p(x)dx\end{aligned}$$

$$\text{function mean: } E[f(x)] = \int f(x)p(x)dx$$

Minimum expected risk

$$R^* = \int \min_{\omega} [R(\alpha|\mathbf{x})] \, p(x)dx$$

Categories of Density Estimation



Setting

- Data: $\mathcal{D} = \mathcal{D}_{j\,j=1}^{C}$
- Assume that $\mathcal{D}_j$ contains no information about $\omega_i, \forall i \neq j$
- We abandon the class label:

$$p(\mathbf{x}|\omega_i \times) \Rightarrow p(\mathbf{x})$$

but

$$p(\mathbf{x}|\omega_i)$$

Setting

- Data: $\mathcal{D} = \mathcal{D}_{j}{}_{j=1}^{C}$
- Assume that $\mathcal{D}_j$ contains no information about $\omega_i, \forall i \neq j$
- We abandon the class label:

$$p(\mathbf{x}|\omega_i \times) \Rightarrow p(\mathbf{x})$$

but

$$p(\mathbf{x}|\omega_i) \neq p(\mathbf{x})$$

Setting

- Data: $\mathcal{D} = \mathcal{D}_{j}{}_{j=1}^{C}$
- Assume that $\mathcal{D}_j$ contains no information about $\omega_i, \forall i \neq j$
- We abandon the class label:

$$p(\mathbf{x}|\omega_i \times) \Rightarrow p(\mathbf{x})$$

but

$$p(\mathbf{x}|\omega_i) \neq p(\mathbf{x})$$

Goal

Model the probability density function $p(\mathbf{x})$, given a finite number of data points, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, drawn from it.

Three Methods

1. Parametric
   - Good: small number of parameters
   - Bad: choice of the parametric form
2. Non-parametric
   - Good: data dictates the approximator
   - Bad: Large number of parameters
3. Semi-parametric
   - Good: combine the best of both worlds
   - Bad: harder to design
   - Good again: design can be subject to optimization

Parametric Density Estimation

Estimate the density from a given functional family

$$\begin{aligned} \text{Given} &: p(\mathbf{x}|\theta) = f(\mathbf{x}, \theta) \\ \text{Find} &: \qquad \theta \end{aligned}$$

Two methods of parameter estimation:

Parametric Density Estimation

Estimate the density from a given functional family

$$\begin{aligned} \text{Given}: & \quad p(\mathbf{x}|\theta) = f(\mathbf{x}, \theta) \\ \text{Find}: & \quad \theta \end{aligned}$$

Two methods of parameter estimation:

1. *Maximum likelihood* method
   - Parameters are viewed as unknown but fixed values
2. *Bayesian* method
   - Parameters are random variables that have their distributions

Nonparametric Density Estimation

Non-parametric methods do not assume any particular form for $p(\mathbf{x})$

1. Histograms
2. Kernel Methods
3. k-nn method

General Idea

$\hat{P}(\mathbf{x})$ is a discrete approximation of $p(\mathbf{x})$

1. Count a number of times that $\mathbf{x}$ lies in the i-th bin

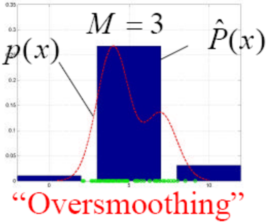
$$H(i) = \sum_{j=1}^{n} I(\mathbf{x} \in B_i), \forall i = 1, 2, \cdots, m$$

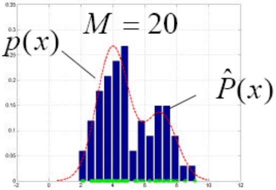
2. Normalize

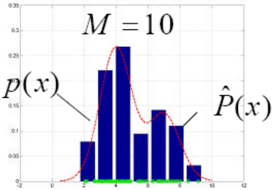
$$\hat{P}(i) = \frac{H(i)}{\sum_{j=1}^{m} H(j)}$$

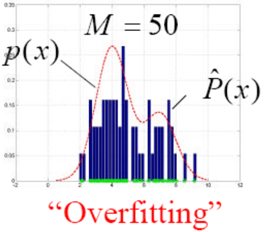
Illustrations

How many bins?









Properties

- Good
  - Once it is constructed, the data can be discarded
  - Quick and intuitive

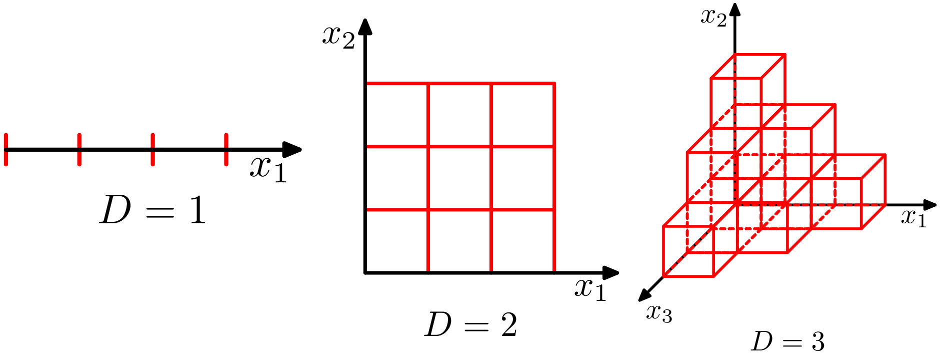
Properties

- Good
  - Once it is constructed, the data can be discarded
  - Quick and intuitive
- Bad
  - Very sensitive to the number of bins, m
  - Estimated density is not smooth
  - Poor generalization in higher dimensions

Curse of Dimensionality (1)

- Imagine we build a histogram of a 1-d feature (e.g., Hue)
  - 10 bins
  - 1 bin $=$ 10% of the input space
  - need at least 10 points to populate every bin
- Add another feature (e.g., saturation)
  - 10 bins again
  - 1 bin $=$ 1% of the input space
  - need at least 100 points to populate every bin
- Add another feature (e.g., value)
  - 10 bins again
  - 1 bin $=$ 0.1% of the input space
  - need at least 1000 points to populate every bin

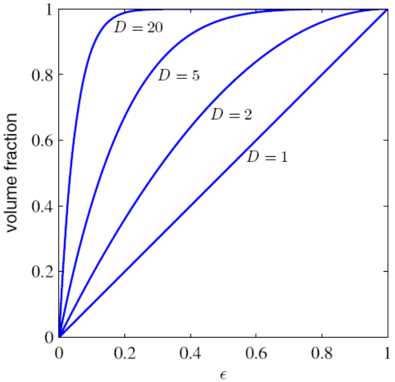
Curse of Dimensionality (2)



Curse of Dimensionality (3)

- Volume of a cube in $\mathbb{R}^d$ with side l: $V_l = l^d$
- Volume of a cube with side $l - \epsilon$: $V_\epsilon = (l - \epsilon)^d$
- Volume of the ϵ-shell:

$$\Delta = V_l - V_\epsilon = l^d - (l - \epsilon)^d$$



Volume fraction of the ϵ-shell to the cube:

$$\frac{\Delta}{V_l} = \frac{l^d - (l - \epsilon)^d}{l^d} = 1 - \left(1 - \frac{\epsilon}{l}\right)^d \rightarrow 1 \text{ as } d \rightarrow \infty$$

General Reasoning (1)

By definition

$$P(\mathbf{x} \in \mathbb{R}) = P = \int_{\mathbb{R}} p(\mathbf{x}')dx'$$

If we have n i.i.d. points drawn from $p(\mathbf{x})$:

$$P(k) = \frac{n!}{k!(n-k)!} P^k (1-P)^{n-k} = B(n, P)$$

where

- P: Prob that k of particular $\mathbf{x}$-es are in $\mathbb{R}$
- $B(n, P)$: binomial distribution of k

General Reasoning (2)

Mean and variance of $B(n, P)$

- Mean:

$$\mu = E[k] = nP \Rightarrow P = E[k/n]$$

- Variance:

$$\sigma^2 = E[(k-\mu)^2] = nP(1-P)$$
$$\Rightarrow E\left[(k/n-P)^2\right] = \frac{\sigma^2}{n^2} = \frac{P(1-P)}{n}$$

When n is large

- $E[k/n]$ is a good estimate of P
- P is distributed around this estimate with vanishing variance

$$\Rightarrow P \simeq \frac{k}{n}$$

General Reasoning (3)

$$P \simeq \frac{k}{n}$$

Under mild assumption

- Assume $p(\mathbf{x})$ is continuous and $\mathbb{R}$ is small
- p only varies very slightly

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

General Reasoning (3)

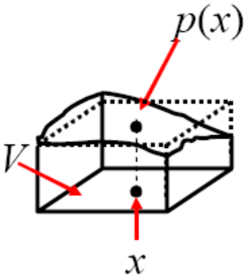
$$P \simeq \frac{k}{n}$$

Under mild assumption

- Assume $p(\mathbf{x})$ is continuous and $\mathbb{R}$ is small
- p only varies very slightly

$$P = \int_{\mathbb{R}} p(\mathbf{x}')d\mathbf{x}' \simeq p(\mathbf{x})V$$

where V: volume of $\mathbb{R}$



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

General Reasoning (4)

Given n data points - how do we really estimate $p(\mathbf{x})$?

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

- Fix V and count how many points k it encloses -

General Reasoning (4)

Given n data points - how do we really estimate $p(\mathbf{x})$?

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

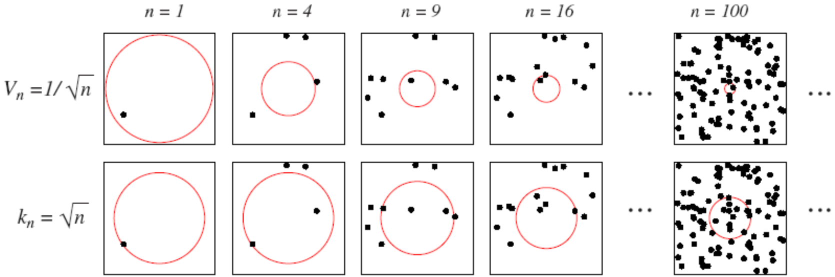
- Fix V and count how many points k it encloses - Kernel methods
- Fix k and vary V until it encloses k points -

General Reasoning (4)

Given n data points - how do we really estimate $p(\mathbf{x})$?

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

- Fix V and count how many points k it encloses - Kernel methods
- Fix k and vary V until it encloses k points - k-Nearest Neighbors (k-NN)



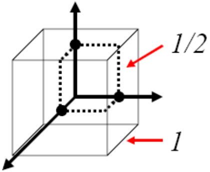
Kernel Function

We choose V by specifying a hypercube with a side h:

$$V = h^d$$

Mathematically:

$$H(\mathbf{x}) = \begin{cases} 1, & |\mathbf{x}^d| < 1/2 \;\; j = 1, \cdots, d \\ 0, & \text{otherwise} \end{cases}$$



this is kernel function, which satisfies the following conditions:

$$H(\mathbf{x}) \geq 0, \forall \mathbf{x}, \text{ and } \int H(\mathbf{x}) d\mathbf{x} = 1$$

Parzen Window

A hypercube with side h centered at $\mathbf{x}_i$:

Parzen Window

A hypercube with side h centered at $\mathbf{x}_i$:

$$H\left((\mathbf{x}-\mathbf{x}_i)/h\right)$$

H can help count the points in a volume V around any $\mathbf{x}$:

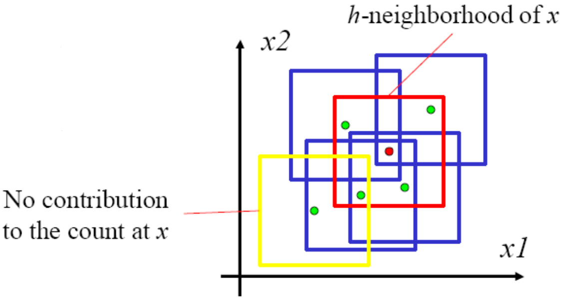
Parzen Window

A hypercube with side h centered at $\mathbf{x}_i$:

$$H\left((\mathbf{x}-\mathbf{x}_i)/h\right)$$

H can help count the points in a volume V around any $\mathbf{x}$:

$$\sum_{i=1}^{n} H\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$



Rectangular Kernel

So the number of points in h-neighborhood of **x**

$$\sum_{i=1}^{n} H\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

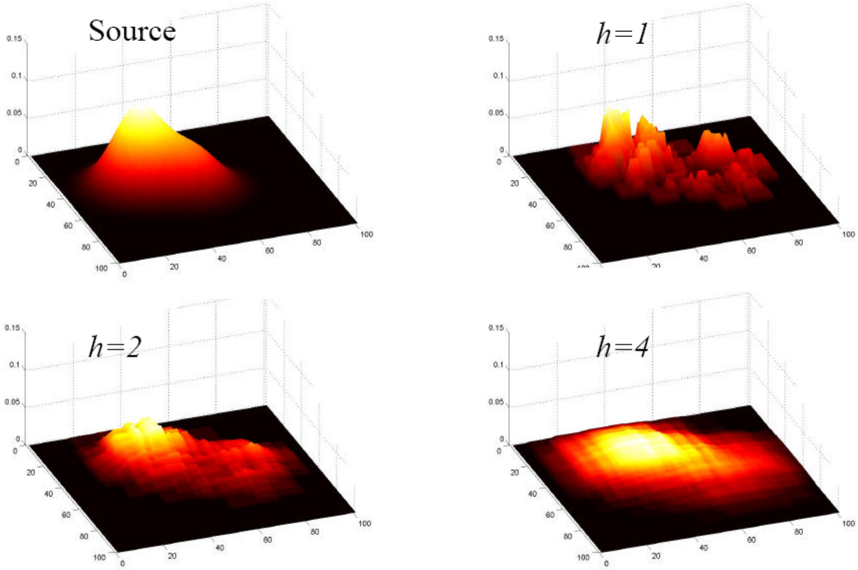
is easily converted to the density estimate:

$$\tilde{p}(\mathbf{x}) = \frac{k(\mathbf{x})}{nV} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\frac{1}{h^d} H\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}_{K(\mathbf{x},\mathbf{x}_i)}$$

where

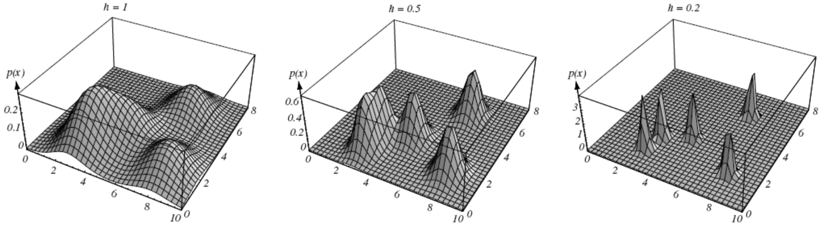
$$\int \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x},\mathbf{x}_i)d\mathbf{x} = \frac{1}{n}\sum_{i=1}^{n}\int K(\mathbf{x},\mathbf{x}_i)d\mathbf{x} = 1$$
$$\Rightarrow \int \tilde{p}(\mathbf{x})d\mathbf{x} = 1$$

Example



Analysis

- if h is very large, $\tilde{p}(\mathbf{x})$ is the superposition of n broad functins, and is a smooth "out-of-focus" estimate of $p(\mathbf{x})$
- if h is very small, $\tilde{p}(\mathbf{x})$ is the superposition of n sharp pulses centered at the samples and is a "noisy" estimate of $p(\mathbf{x})$
- as h approaches zero, $K(\mathbf{x}, \mathbf{x}_i)$ approaches a Dirac delta function centered at $\mathbf{x}_i$, and $\tilde{p}(\mathbf{x})$ is the superposition of delta functions



Smoothed Window Function

- The problem is as in histograms

Smoothed Window Function

- The problem is as in histograms - it is discontinuous
- we can choose a smoother function, s.t.,

$$\tilde{p}(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \quad \text{and} \quad \int \tilde{p}(\mathbf{x}) d\mathbf{x} = 1$$

(ensured by kernel conditions)

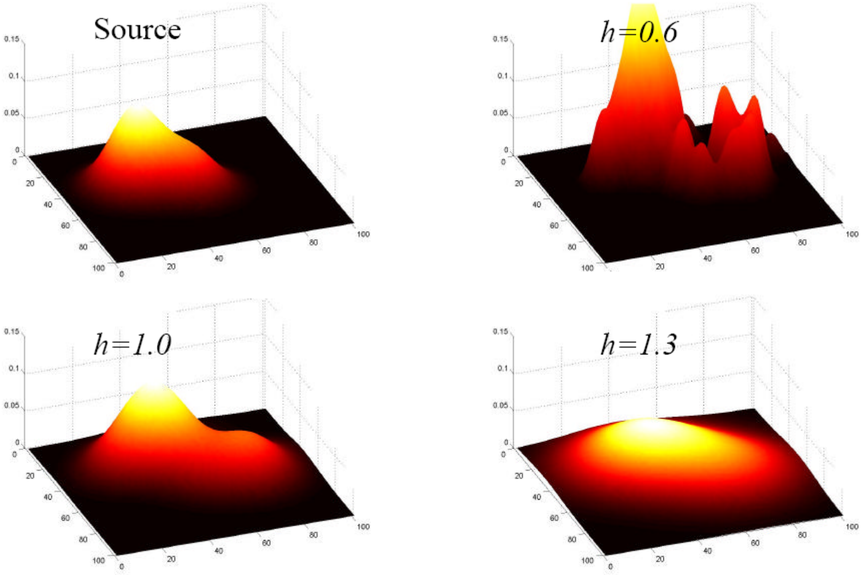
- e.g., a spherical Gaussian

$$K(\mathbf{x}, \mathbf{x}_i) = \frac{1}{(\sqrt{2\pi} h)^d} \exp\left(-\frac{||\mathbf{x} - \mathbf{x}_i||^2}{2h^2} \right)$$

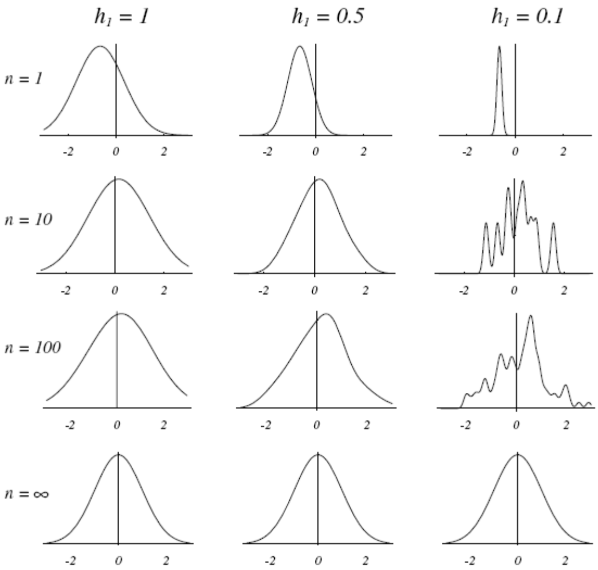
so

$$\tilde{p}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(\sqrt{2\pi} h)^d} \exp\left(-\frac{||\mathbf{x} - \mathbf{x}_i||^2}{2h^2} \right)$$

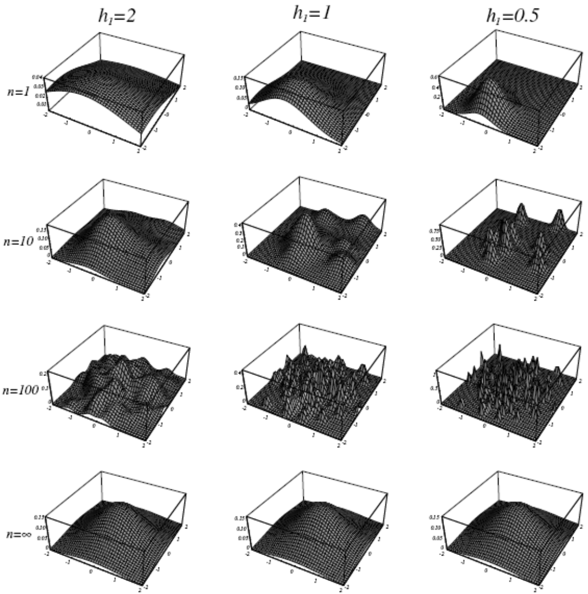
Example



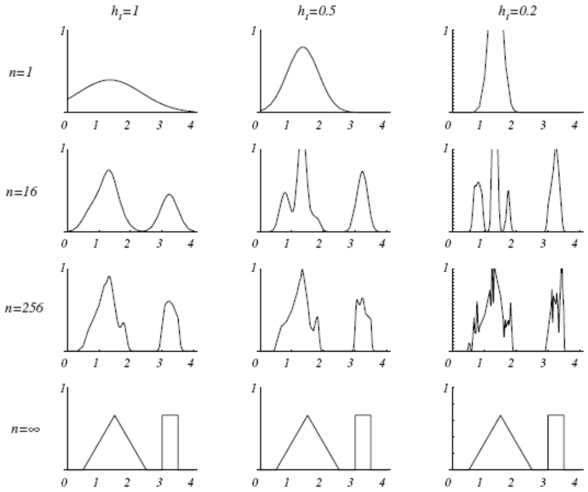
Unimodal 1d-Gaussian



Unimodal 2d-Gaussian

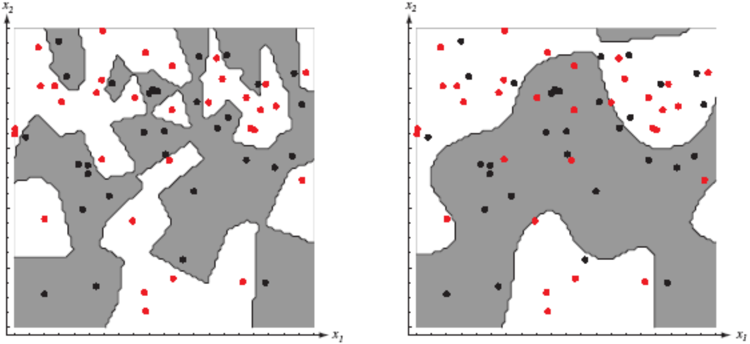


Bi-modal

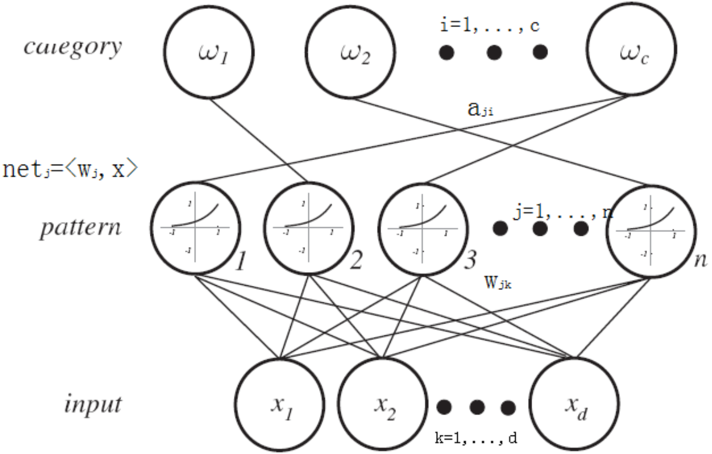


Parzen Windows for Classification

- Densities estimated using Parzen windows can be used for classification using Bayesian decision
- Training error can be made arbitrarily low by making the window width sufficiently small



PNN: Topology



PNN: Principle

The net activation:

$$net_j = \mathbf{w}_j^\top \mathbf{x}$$

The window function for Parzen windows algorithm:

$$\begin{aligned}
\phi\left(\frac{x - \mathbf{w}_j}{h_n}\right) &\propto \exp(-(\mathbf{x} - \mathbf{w}_j)^\top(\mathbf{x} - \mathbf{w}_j)/2\sigma^2) \\
&= \exp(-(\mathbf{x}^\top\mathbf{x} + \mathbf{w}_j^\top\mathbf{w}_j - 2\mathbf{x}^\top\mathbf{w}_j)/2\sigma^2) \\
&= \exp(net_j - 1)/\sigma^2
\end{aligned}$$

where the input is normalized, i.e., $x_{jk} \leftarrow x_{jk}/(\sum_q^d x_{jq}^2)^{1/2}$ and set $w_{jk} \leftarrow x_{jk}$

PNN: Training

Algorithm 1 (PNN training)

begin initialize $j = 0, n = \#\text{patterns}$
  do $j \leftarrow j + 1$
    normalize : $x_{jk} \leftarrow x_{jk} / \left(\sum_{i}^{d} x_{ji}^2 \right)^{1/2}$
    train : $w_{jk} \leftarrow x_{jk}$
    if $\mathbf{x} \in \omega_i$ **then** $a_{ji} \leftarrow 1$
  until $j = n$

PNN: Test

Algorithm 2 (PNN classification)

```
begin initialize  j = 0, x = test pattern
        do  j ← j + 1
            z_j ← w_j^t x
            if a_ji = 1 then  g_i ← g_i + exp[(z_j − 1)/σ^2]
        until  j = n
    return class ← arg max_i g_i(x)
end
```

Problem with Kernel Estimation

- Need to choose the width parameter h

Problem with Kernel Estimation

- Need to choose the width parameter h
  - empirically choosing
  - adaptively choosing, e.g., $h_j = hd_{jk}$ -

Problem with Kernel Estimation

- Need to choose the width parameter h
  - empirically choosing
  - adaptively choosing, e.g., $h_j = hd_{jk}$ - d_{jk} the distance from $\mathbf{x}_j$ to k-th nearest neighbor
- need to store all data to represent the density
  - leading to Mixture density estimation

K-Nearest Neighbors

Recall that

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

- We fix k (typically $k = \sqrt{n}$) and expand V to contain k points
- Is it a true density?

e.g., $n = 1, k_n = \sqrt{n} = 1$, then,

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

K-Nearest Neighbors

Recall that

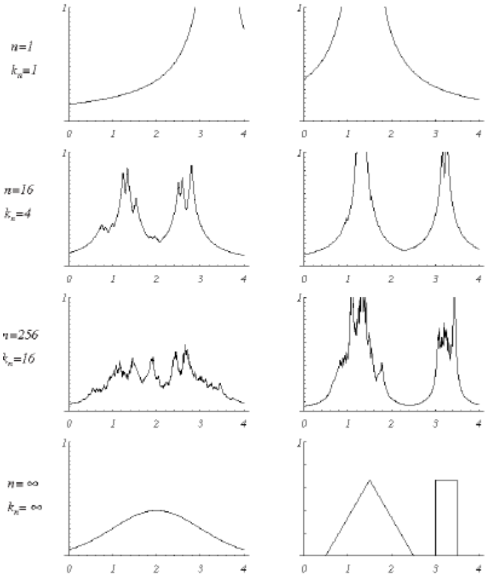
$$\tilde{p}(\mathbf{x}) = \frac{k}{nV}$$

- We fix k (typically $k = \sqrt{n}$) and expand V to contain k points
- Is it a true density?

e.g., $n = 1, k_n = \sqrt{n} = 1$, then,

$$p_n\tilde{(}\mathbf{x})(\mathbf{x}) = \frac{k}{nV} = \frac{1}{2|\mathbf{x} - \mathbf{x}_1|}$$

It is useful for a number of theoretical and practical reasons


n=1
k_n=1
n=16
k_n=4
n=256
k_n=16
n= ∞
k_n= ∞
0
1
2
3
4

k-NN Classification Rule

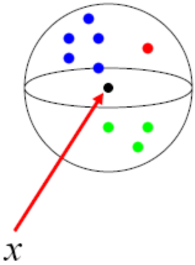
Data:

- n - total points
- n_i - points in class ω_i

Need to find the class label for a query, **x**

Expand a sphere from **x** to include k points

- k - number of neighbors of **x**
- k_i - points of class ω_i among k



k-NN Classification

- class priors are given by:

k-NN Classification

- class priors are given by: $P(\omega_i) = n_i/n$
- we can estimate conditional and marginal densities around any $\mathbf{x}$

$$p(\mathbf{x}|\omega_i) = \frac{k_i}{n_i V}, \quad p(\mathbf{x}) = \frac{k}{nV}$$

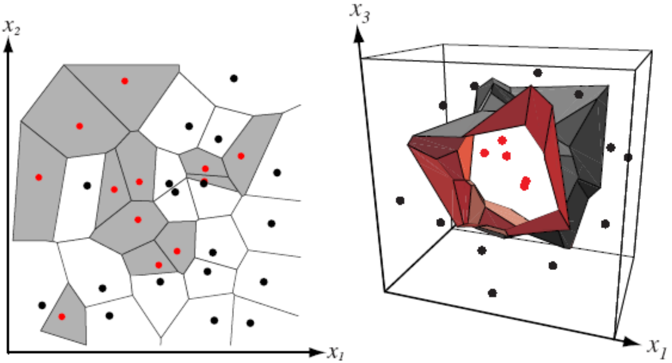
- By Bayes rule:

$$p(\omega_i|\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i)/p(\mathbf{x}) = \frac{k_i}{k}$$

- for minimum error rate classification:

$$\mathbf{x} \in \omega_m = arg \max_i k_i$$

Voronoi Diagram



Important theoretical result

In the extreme case, $k = 1$, it can be shown that

$$\text{for } P = \lim_{n\to\infty} P_n(\text{error})$$

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

$\Rightarrow$ using just a single neighbor rule, the error rate is at most twice the Bayes error!!!

Problem with Non-parametric Density Estimation

- Memory: need to store all data points
- Computation: need to compute distances to all data points every time
- Parameter choice: need to choose the smoothing parameter

Mixture Density Model

Mixture model – a linear combination of parametric densities

Number of components

$$p(x) = \sum_{j=1}^{M} p(x \mid j) P(j)$$

Component density

Component "prior"

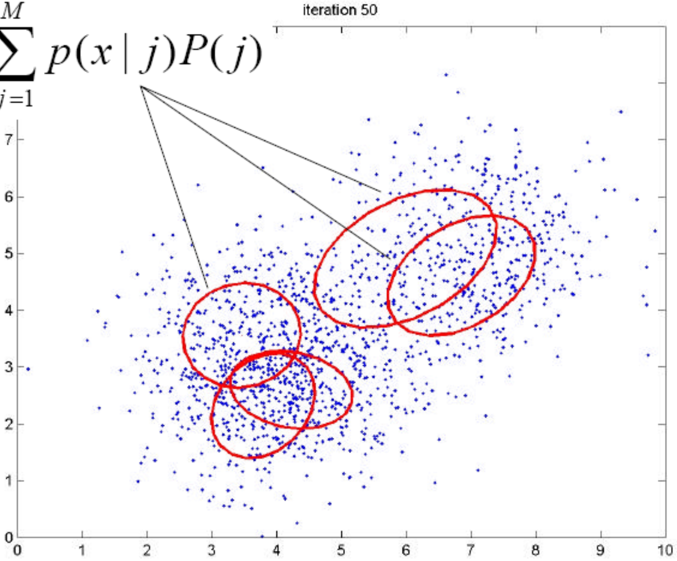
$$P(j) \geq 0, \quad \forall j \qquad \text{and} \qquad \sum_{j=1}^{M} P(j) = 1$$

Uses MUCH less "kernels" than kernel methods

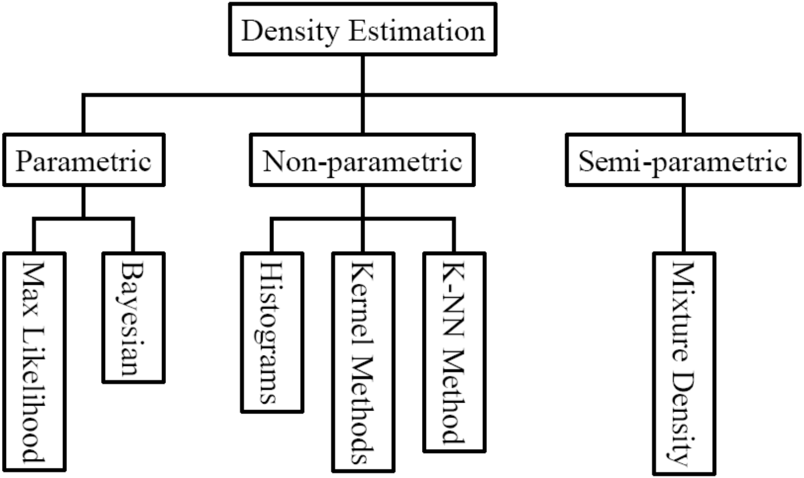
Kernels are parametric densities, subject to estimation

Example

$$p(x)=\sum_{j=1}^{M} p(x\mid j)P(j)$$



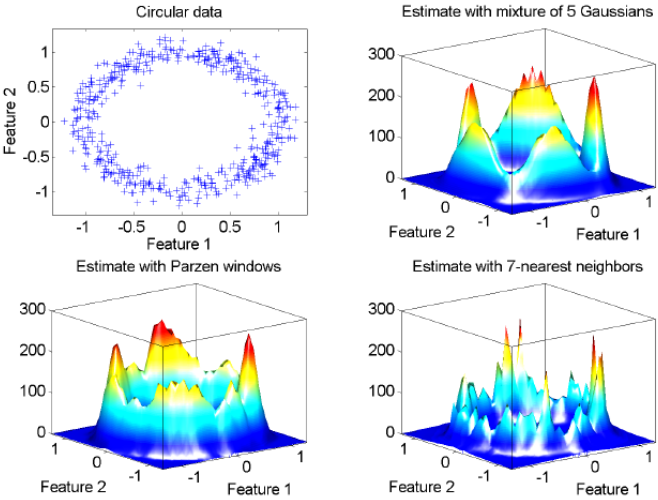
Categories of Density Estimation



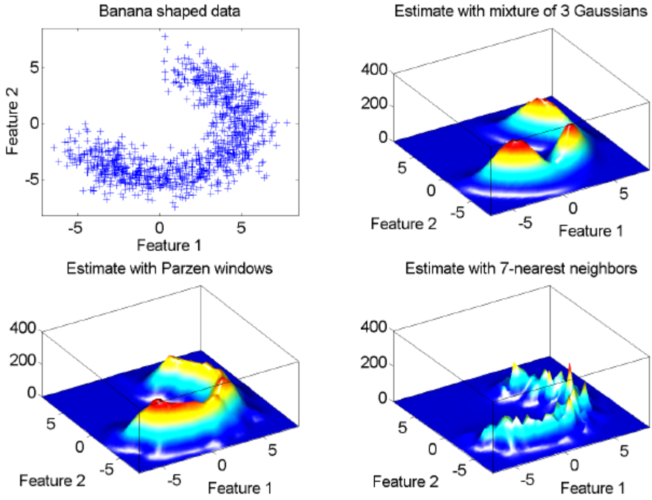
Three Methods

1. Parametric
   - Good: small number of parameters
   - Bad: choice of the parametric form
2. Non-parametric
   - Good: data dictates the approximator
   - Bad: Large number of parameters
3. Semi-parametric
   - Good: combine the best of both worlds
   - Bad: harder to design
   - Good again: design can be subject to optimization

Examples


Circular data
Feature 2
Feature 1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
Estimate with mixture of 5 Gaussians
300
200
100
0
Feature 2
Feature 1
Estimate with Parzen windows
Feature 2
Feature 1
Estimate with 7-nearest neighbors
Feature 2
Feature 1

Examples



Banana shaped data
Feature 2
Feature 1
5
0
-5
-5
0
5
Estimate with mixture of 3 Gaussians
400
200
0
Feature 2
Feature 1
Estimate with Parzen windows
400
200
0
Feature 2
Feature 1
Estimate with 7-nearest neighbors
400
200
0
Feature 2
Feature 1