

# Nonparametric Density Estimation (I)

**Jin Young Choi**

**Seoul National University**

# Outline

---

- Nonparametric Density Estimation
- Histogram Approach
- Parzen-window method
- $K_n$ -Nearest-Neighbor Estimation
- Gaussian Mixture Models
- Expectation and Maximization

# Nonparametric Density Estimation

---

- The common **parametric** forms **rarely fit** the densities actually encountered **in practice**.
- Classical parametric densities are **unimodal**, whereas many practical problems involve **multimodal** densities.
- We examine **nonparametric** procedures that can be used with **arbitrary distribution** and **without** the assumption that the **parametric forms** of the underlying densities are **known**.

# Nonparametric Density Estimation

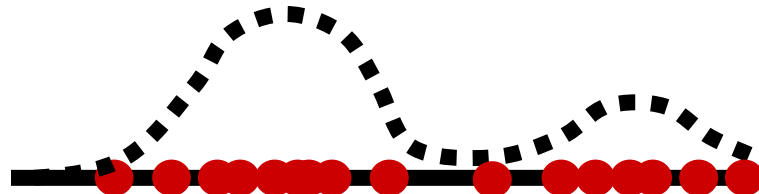
---

- There are several types of nonparametric methods:
  - Procedures for estimating the density functions  $p(\mathbf{x} | \omega_j)$  from sample patterns (Likelihood estimation).
  - Procedures for directly estimating a posteriori probability  $P(\omega_j | \mathbf{x})$
  - Nearest neighbor rule which bypass probability estimation, and go directly to decision functions.  $p(\mathbf{x} | D_j^n)$

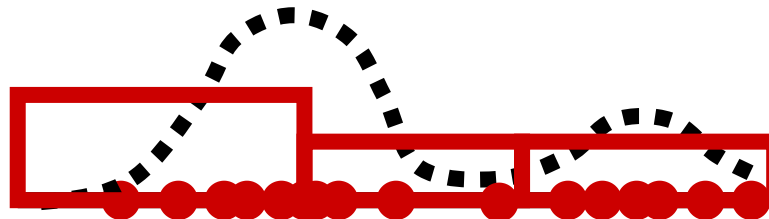
# The Histogram Method: Example

---

- Assume (one dimensional) data
- Some points were sampled from a combination of two Gaussians:



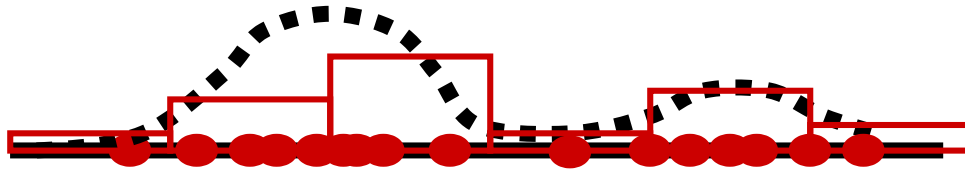
- 3 bins



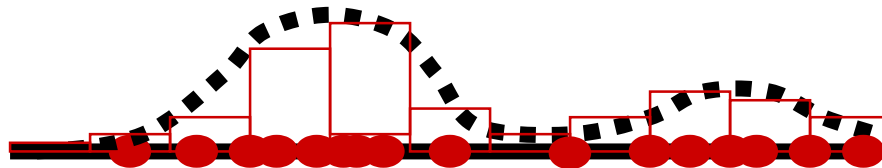
# The Histogram Method: Example

---

- 7 bins



- 11 bins



# Density Estimation

---

- The probability for  $\mathbf{x}$  to fall into  $R$  is

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

- Suppose we have  $n$  i.i.d. samples  $\mathbf{x}_1, \dots, \mathbf{x}_n$  drawn according to  $p(\mathbf{x})$ . The probability that  $k$  of them fall in  $R$  is

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

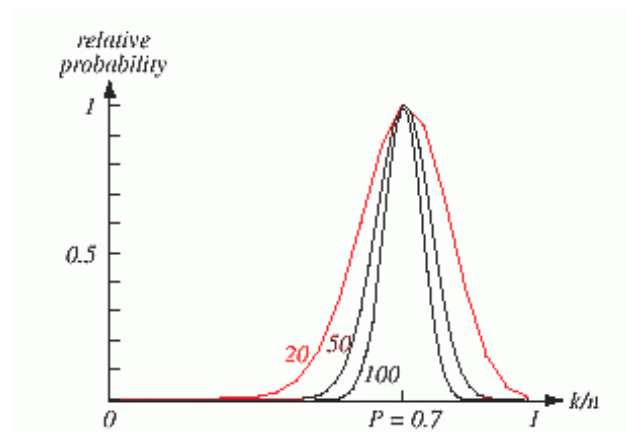
- The expected value for  $k$  is  $E[k] = np$  and variance  $\text{var}(k) = np(1-p)$ . The relative part of samples which fall into  $R$ ,  $(k/n)$ , is also a random variable for which

$$E\left[\frac{k}{n}\right] = p, \quad \text{var}\left[\frac{k}{n}\right] = \frac{p(1-p)}{n}$$

- When  $n$  is growing up, the variance is making smaller and  $\frac{k}{n}$  is becoming to be better estimator for  $p$ .

# Density Estimation

- $P_k$  sharply peaks about the mean,
- so the  $k/n$  is a good estimate of  $p$ .



- For small enough  $R$

$$p = \int_R p(\mathbf{x}') d\mathbf{x}' \approx p(\mathbf{x}) V,$$

- where  $\mathbf{x}$  is within  $R$  and  $V$  is a volume enclosed by  $R$ .
- Thus

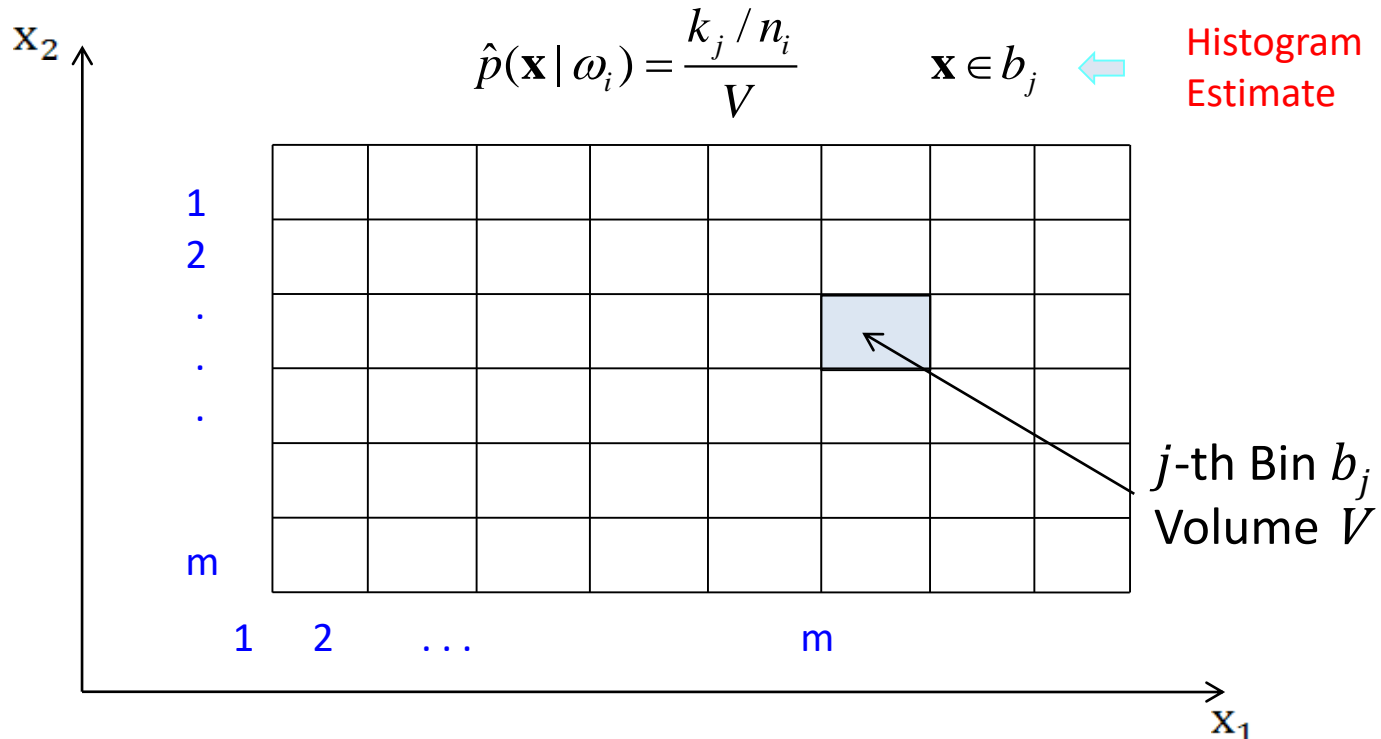
$$p(\mathbf{x}) \approx \frac{k/n}{V} \quad (*)$$

# Histogram Approach

- **Histogram** is the simplest method of estimating a p.d.f.

$n_i$  samples in class  $\omega_i$ , i.e.,  $\mathbf{X}_l \in \omega_i$ ,  $l=1, \dots, n_i$

The number of  $\mathbf{X}_l \in \omega_i$  in  $b_j$  is  $k_j$ .



# Histogram Approach

---

- The  $\hat{p}(\mathbf{x} | \omega_i)$  is constant over  $b_j$
- Let us verify that  $\hat{p}(\mathbf{x} | \omega_i)$  is a density function:

$$\int \hat{p}(\mathbf{x} | \omega_i) d\mathbf{x} = \sum_{j=1}^m \int_{b_j} \frac{k_j}{n_i V} d\mathbf{x} = \frac{1}{n_i} \sum_{j=1}^m k_j = 1$$

- We can choose the number of bins in each axis,  $m$ , and their starting points. Fixation of starting points is not critical, but  $m$  is **important**.
- It place a role of smoothing parameter. Too big  $m$  makes histogram spiky, for too little  $m$  we loose a true form of the density function

# Histogram Approach

---

- The histogram p.d.f. estimator is very effective.
- We can do it *online* : all we should do is to update the counters  $k_j$  during the run time, so we do not need to keep all the data which could be huge.
- But its usefulness is limited only to low dimensional vectors  $\mathbf{x}$ , because the number of bins,  $N_b$ , grows exponentially with dimensionality  $d$  :

$$N_b = m^d.$$

- This is the so called “curse of dimensionality”

# Three Conditions for Density Estimation

---

- Reducing the region by increasing the samples
- Let us take a growing sequence of samples  $n = 1, 2, 3 \dots$
- We take regions  $R_n$  with reduced volumes  $V_1 > V_2 > V_3 > \dots$
- Let  $k_n$  be the number of samples falling in  $R_n$
- Let  $p_n(x)$  be the  $n^{\text{th}}$  estimate for  $p(x)$
- If  $p_n(x)$  is to converge to  $p(x)$ , 3 conditions must be required:
  - $\lim_{n \rightarrow \infty} V_n = 0$ , resolution as big as possible (for smoothing)
  - $\lim_{n \rightarrow \infty} k_n = \infty$ , to preserve  $\int p(x) dx = 1$
  - $\lim_{n \rightarrow \infty} \frac{k_n}{n} = 0$  to guarantee convergence of  $p(\mathbf{x}) \approx \frac{k/n}{V}$  (\*)

$$\int \hat{p}(\mathbf{x} | \omega_i) d\mathbf{x} = \sum_{j=1}^m \int_{b_j} \frac{k_j}{n_i V} d\mathbf{x} = \frac{1}{n_i} \sum_{j=1}^m k_j = 1$$

# PARZEN WINDOW and KNN

---

- How to obtain the sequence  $R_1, R_2, \dots$ ?
- There are 2 common approaches of obtaining sequences of regions that satisfy the convergence conditions:
  - Shrink an initial region by specifying the volume  $V_n$  as some function of  $n$ , such as  $V_n = 1/\sqrt{n}$  and show that  $k_n$  and  $k_n/n$  behave properly i.e.  $p_n(x)$  converges to  $p(x)$ .
  - **This is Parzen-window (or kernel) method.**
  - Specify  $k_n$  as some function of  $n$ , such as  $k_n = \sqrt{n}$ . Here the volume  $V_n$  is grown until it encloses  $k_n$  neighbors of  $x$ .
  - **This is  $k_n$  –nearest-neighbor method.**

$$\begin{aligned}\lim_{n \rightarrow \infty} V_n &= 0, \\ \lim_{n \rightarrow \infty} k_n &= \infty, \\ \lim_{n \rightarrow \infty} \frac{k_n}{n} &= 0\end{aligned}$$

# PARZEN WINDOWS

---

- Assume that the region  $R_n$  is a  $d$  –dimensional hypercube.
- If  $h_n$  is the length of an edge of that hypercube,
- then its volume is given by  $V_n = h_n^d$ .
- Define the following **window function**:

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

- $\varphi(\mathbf{u})$  defines a unit hypercube centered at the origin.
- $\varphi((\mathbf{x} - \mathbf{x}_i)/h_n) = 1$  , if  $x_i$  falls within the hypercube of volume  $V_n$  centered at  $x$ , and is zero otherwise.
- The number of samples in this hypercube is given by:

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

## ■ PARZEN WINDOWS cont.

---

- Since

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

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

- Rather than limiting ourselves to the hypercube window, we can use a more general class of window functions.
- The window function is being used for [interpolation](#). Each sample contributing to the estimate in accordance with its distance from  $\mathbf{x}$ .
- $p_n(\mathbf{x})$  must:
  - *be nonnegative*
  - *integrate to 1.*

# PARZEN WINDOWS cont.

---

- This can be assured by requiring the window function itself be a density function, i.e.,

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

- Effect of the window size  $h_n$  on  $p(x)$

- Define the function

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

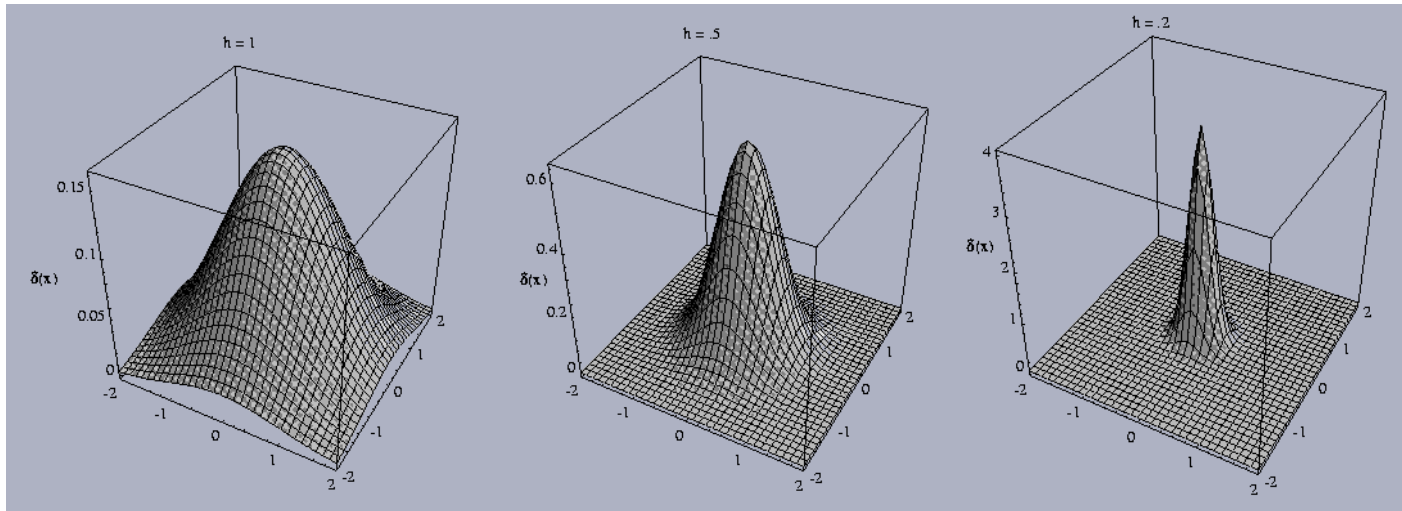
- then, we write  $p_n(\mathbf{x})$  as the average

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

- Since  $V_n = h_n^d$ ,  $h_n$  affects both the amplitude and the width of  $\delta_n(\mathbf{x})$

# PARZEN WINDOWS cont.

---



- Examples of two-dimensional circularly symmetric normal Parzen windows for 3 different values of  $h$ .
- If  $h_n$  is **very large**, the amplitude of  $\delta_n(\mathbf{x})$  is small, and  $\mathbf{x}$  must be far from  $\mathbf{x}_i$  since  $\delta_n(\mathbf{x} - \mathbf{x}_i)$  decreases slowly from  $\delta_n(\mathbf{0})$

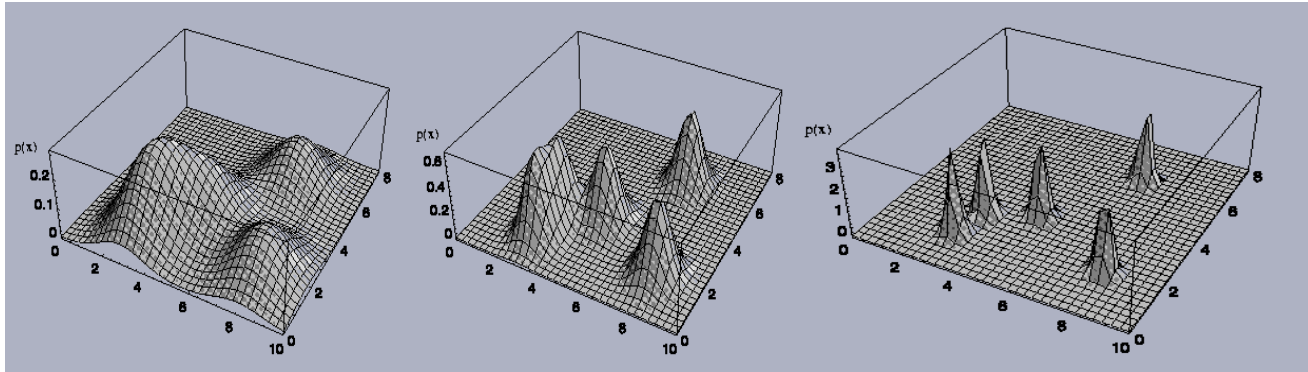
# PARZEN WINDOWS cont.

---

- In this case  $p_n(\mathbf{x})$  is the **superposition** of  $n$  **broad**, slowly varying functions, and is very smooth "**out-of-focus**" estimate for  $p(\mathbf{x})$ .
- If  $h_n$  is **very small**, the peak value of  $\delta_n(\mathbf{x} - \mathbf{x}_i)$  is large, and occurs near  $\mathbf{x} = \mathbf{x}_i$ .
- In this case,  $p_n(\mathbf{x})$  is the superposition of  $n$  **sharp** pulses centered at the samples: an erratic, "**noisy**" estimate.
- As  $h_n$  **approaches zero**,  $\delta_n(\mathbf{x} - \mathbf{x}_i)$  approaches a Dirac delta function centered at  $\mathbf{x}_i$ , and  $p_n(\mathbf{x})$  approaches a superposition of delta functions centered at the samples.

# PARZEN WINDOWS cont.

---



- 3 Parzen-window density estimates using 5 samples,
  - The choice of  $h_n$  (or  $V_n$ ) has an important effect on  $p_n(x)$
  - 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**.
  - If there is limited number of samples, then seek some acceptable compromise.
  - If we have unlimited number of samples, then let  $V_n$  approach zero as  $n$  increases, and have  $p_n(x)$  converge to the unknown density  $p(x)$ .

# PARZEN WINDOWS cont.

---

- **Example 1:**  $p(\mathbf{x})$  is a zero-mean, unit variance, univariate normal density. Let the widow function be of the same form:

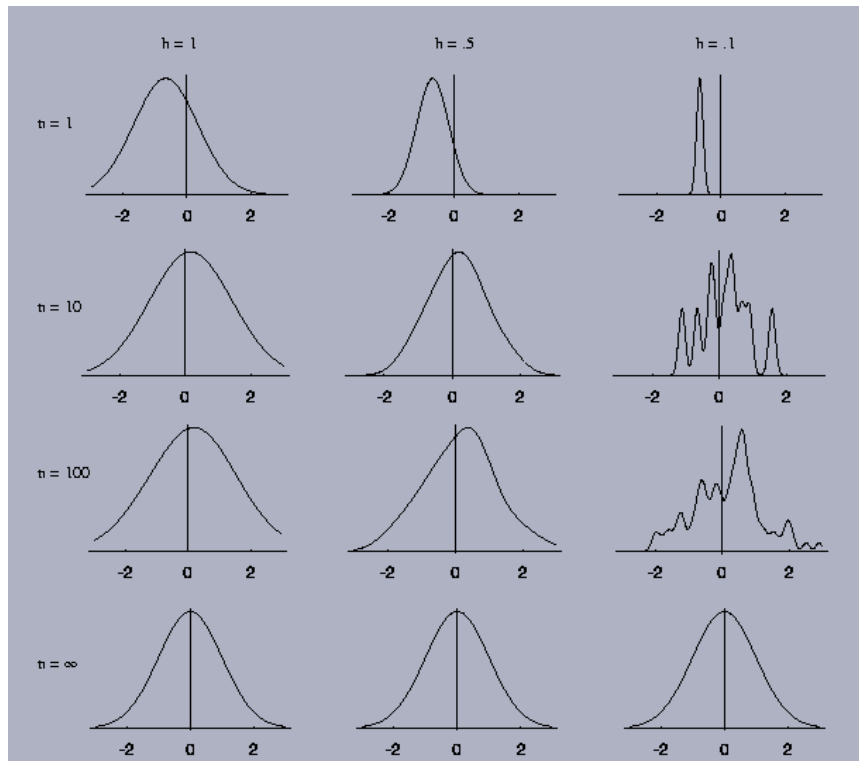
- $$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$

- Let  $h_n = h_1 / \sqrt{n}$  where  $h_1$  is a parameter
- $p_n(\mathbf{x})$  is an average of normal densities centered at the samples:

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

# Examples cont.

- Generate a set of normally distributed random samples.
- Vary  $n$  and  $h_1$ .

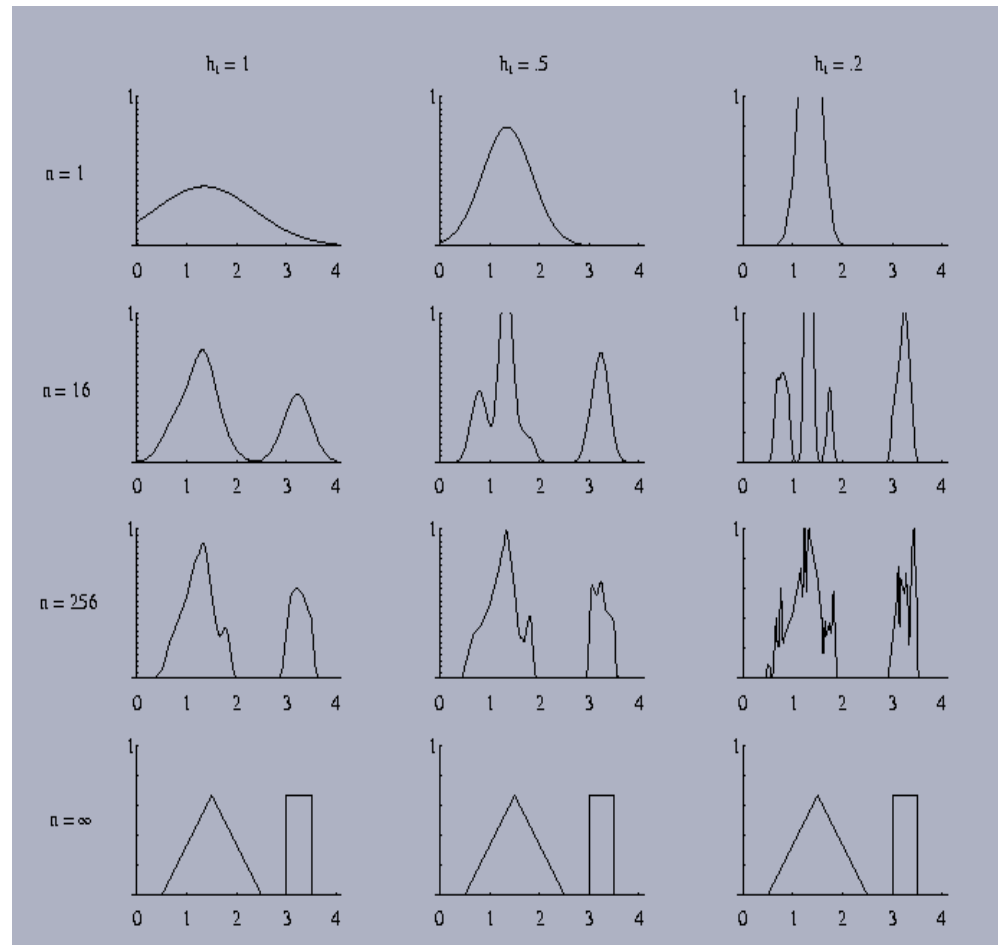


- ✓ The results depend both on  $n$  and  $h_1$ .
- ✓ For  $n = 1$ ,  $p_n(\mathbf{x})$  is merely a single Gaussian centered about the first sample, which has neither the mean nor the variance of the true distribution.
- ✓ For  $n = 10$  and  $h_1 = 0.1$ , the contributions of the individual samples are discernible. This is not the case for  $h_1 = 1$  and  $h_1 = 0.5$ .

# Examples cont.

## ■ Example 2:

- ✓ Let  $\varphi(u)$  and  $h_n$  be the same as in Example 1.
- ✓ but let the unknown density be a mixture of uniform and a triangle density.
- ✓ The case  $n = 1$  tells more about the window function than it tells about the unknown density.
- ✓ For  $n = 16$ , none of the estimates is good.
- ✓ For  $n = 256$ , and  $h_1 = 1$ , the estimates are beginning to appear acceptable.



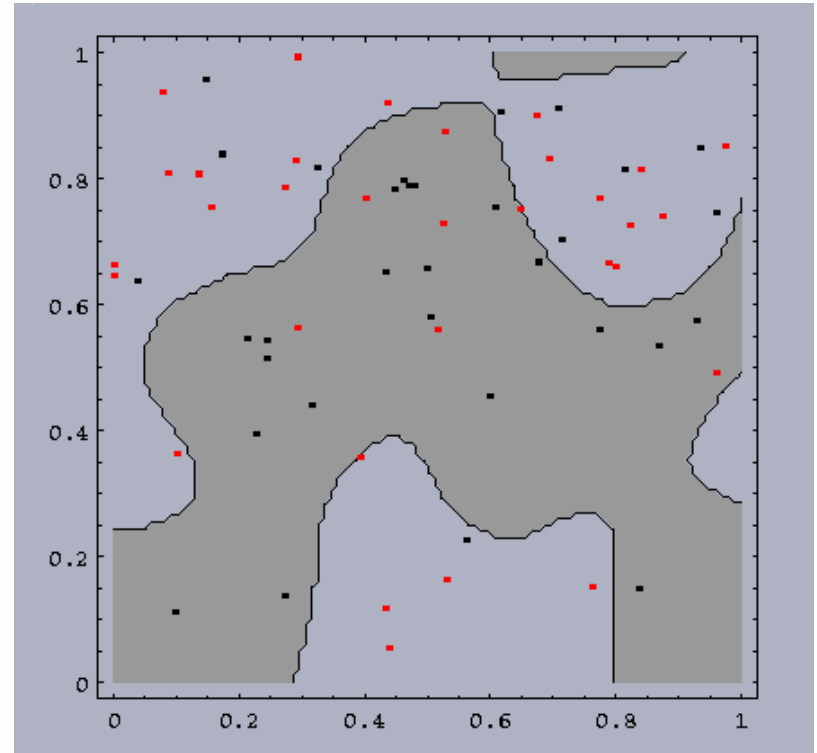
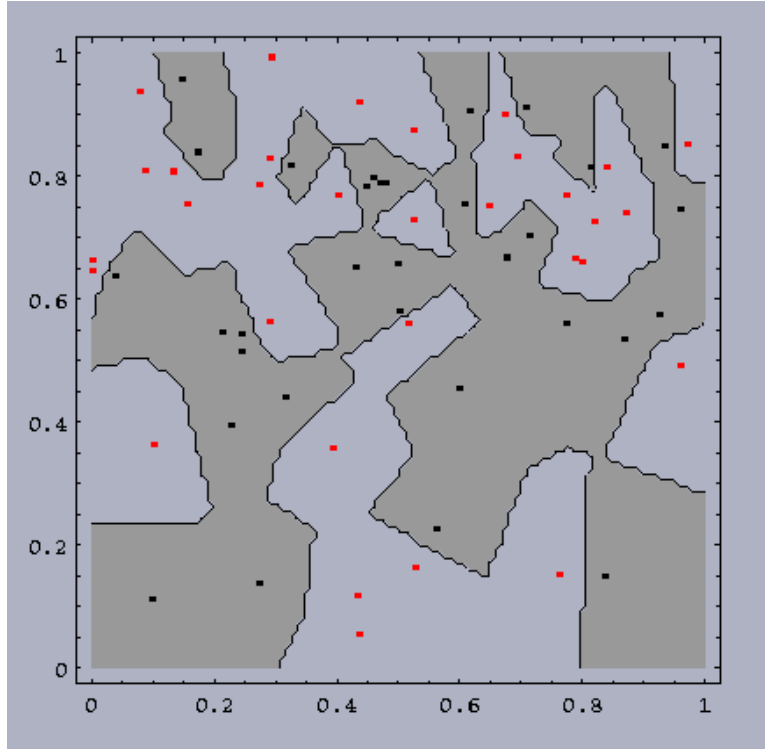
# Examples cont.

---

- Classification
  - To make a classification we should:
    - ✓ Estimate the density for each category using Parzen-window method.
    - ✓ Classify a test point by the label corresponding to the maximum posterior.
  - The decision regions for a Parzen-window classifier depend upon the choice of window function.

## Examples cont.

---



Small  $h$ : more complicated boundaries.    Large  $h$ : Less complicated boundaries.

# Examples cont.

---

- A small  $h$  would be appropriate for the higher density region, while a large  $h$  for the lower density region.
- No single window width is ideal overall.
- In general, the training error can be made arbitrarily low by making the window width sufficiently small.
- **Remember**, the goal of creating a classifier is to classify novel patterns, and a low training error **does not guarantee a small test error**.

# Advantages of Nonparametric Techniques

---

- Generality: same procedure can be used for unimodal normal and multimodal mixture.
- We do not need to make assumption about the distribution ahead of time.
- With enough samples, we are assured of convergence to an arbitrarily complicated target density

# Disadvantages of Nonparametric Techniques

---

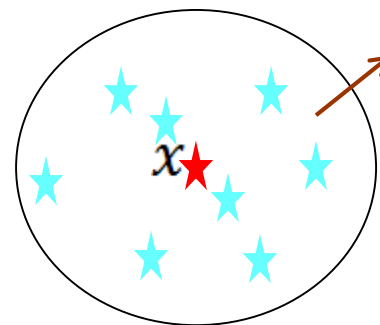
- Number of samples needed may be very large (much larger than would be required if we knew the form of the unknown density) .
- Severe requirements for computation time and storage.
- The large number of samples grows exponentially with the dimensionality of the feature space ("curse of dimensionality")
- Sensitivity to the choice of the window size:
  - Too small: most of the volume will be empty, and the estimate  $p_n(\mathbf{x})$  will be very erratic.
  - Too large: important variations may be lost due to averaging.
- It may be the case that a cell volume appropriate for one region of the feature space might be entirely unsuitable in a different region.

# $K_n$ -Nearest-Neighbor Estimation

---

- To estimate  $p(x)$  from  $n$  training samples, we center a cell about  $x$  and let it grow until it captures  $k_n$  samples, where  $k_n$  is some specified function of  $n$ .
  - These samples are the  $k_n$  **nearest-neighbors** of  $x$ .
  - If the density is high near  $x$ , the cell will be relatively small
- $\Rightarrow$  good resolution.

$$p(\mathbf{x}) \approx \frac{k/n}{V} \quad (*)$$
$$\lim_{n \rightarrow \infty} V_n = 0, \quad \lim_{n \rightarrow \infty} k_n = \infty, \quad \lim_{n \rightarrow \infty} \frac{k_n}{n} = 0$$



# $K_n$ -Nearest-Neighbor Estimation

---

- If we take  $p_n(x) = \frac{k_n / n}{V_n}$  we want  $k_n \rightarrow \infty$  when
- $n \rightarrow \infty$  for  $k_n / n$  be a good estimate of probability
- But  $k_n$  should grow **sufficiently slow** so that the **volume** of the cell captured  $k_n$  samples will **shrink to zero**.
- Thus  $\lim_{n \rightarrow \infty} k_n / n \rightarrow 0$  is necessary and sufficient for  $p_n(x)$  to converge to  $p(x)$ .
- If  $k_n = \sqrt{n}$  and assume that  $p_n(x)$  is good approximation for  $p(x)$ ,  
i.e.,  $V_n \approx 1 / (\sqrt{n} p(x))$ .
- Thus  $V_n \approx V_1 / \sqrt{n}$  but with  $V_1 = 1 / p(x)$  determined by the nature of the data

# Comparison of Density Estimators

---

- Parzen window estimates
  - require the storage of all the observations
  - $n$  evaluations of the kernel function for each estimate
  - Computational complexity:  $O(dn)$ , parallel circuit
- Nearest neighbor estimates
  - also require the storage of all the observations
  - Computational complexity:  $O(dn)$ , parallel circuit, 3 algorithms
- Histogram estimates
  - do not require storage for all the observations,
  - Just require storage for description of the bins.
  - But the number of the bins grows exponentially with dimension.

# Interim Summary

Histogram

From histogram to density estimation

Convergence conditions

Parzen window

Parzen window Function (Kernel)

Cube kernel, Gaussian kernel

Window size and performance

Classification

$K_n$  – Nearest Neighbor

$$p(\mathbf{x}) \approx \frac{k/n}{V} \quad (*)$$
$$\lim_{n \rightarrow \infty} V_n = 0, \quad \lim_{n \rightarrow \infty} k_n = \infty, \quad \lim_{n \rightarrow \infty} \frac{k_n}{n} = 0$$

$$\lim_{n \rightarrow \infty} p_n(x) = p(x).$$

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

$$k_n = \sqrt{n}$$

$$V_n \approx 1/(\sqrt{n}p(x))$$



Adaptive window size

$K_n$  – Nearest Neighbor

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

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



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

Same window size

Parzen window

# Nonparametric Density Estimation (II)

**Jin Young Choi**

**Seoul National University**

# Outline

---

- Nonparametric Density Estimation
- Histogram Approach
- Parzen-window method
- $K_n$ -Nearest-Neighbor Estimation
- Gaussian Mixture Models
- Expectation and Maximization

# Interim Summary

Histogram

From histogram to density estimation

Convergence conditions

Parzen window

Parzen window Function (Kernel)

Cube kernel, Gaussian kernel

Window size and performance

Classification

$K_n$  – Nearest Neighbor

$$p(\mathbf{x}) \approx \frac{k/n}{V} \quad (*)$$
$$\lim_{n \rightarrow \infty} V_n = 0, \quad \lim_{n \rightarrow \infty} k_n = \infty, \quad \lim_{n \rightarrow \infty} \frac{k_n}{n} = 0$$

$$\lim_{n \rightarrow \infty} p_n(x) = p(x).$$

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

$$k_n = \sqrt{n}$$

$$V_n \approx 1/(\sqrt{n}p(x))$$



Adaptive window size

$K_n$  – Nearest Neighbor

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

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



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

Same window size

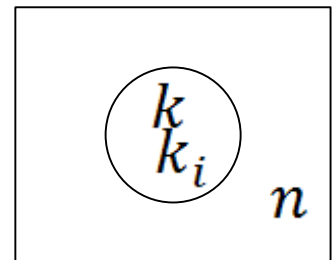
Parzen window

# Classification with K-NN and Parzen window: Estimation of a posteriori probabilities

---

- The  $k$ -NN (and Parzen window) techniques can be used to estimate the a posteriori probabilities  $p(\omega_i | \mathbf{x})$  from a set of  $n$  labeled samples.
- Suppose that we place a cell of volume  $V$  around  $x$  and capture  $k$  samples:

- $k_i$  are labeled  $\omega_i$
- $k - k_i$  have other labels.



- A simple estimate for the joint probability density  $p(\mathbf{x}, \omega_i)$  is

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

- A reasonable estimate for  $p(\omega_i | \mathbf{x})$  is

$$p_n(\omega_i | \mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^C p_n(\mathbf{x}, \omega_j)} = \frac{k_i}{k}$$

Instance-Based Learning Classifier  
Approximated Minimum error classifier

# Classification With Nearest Neighbor Rule

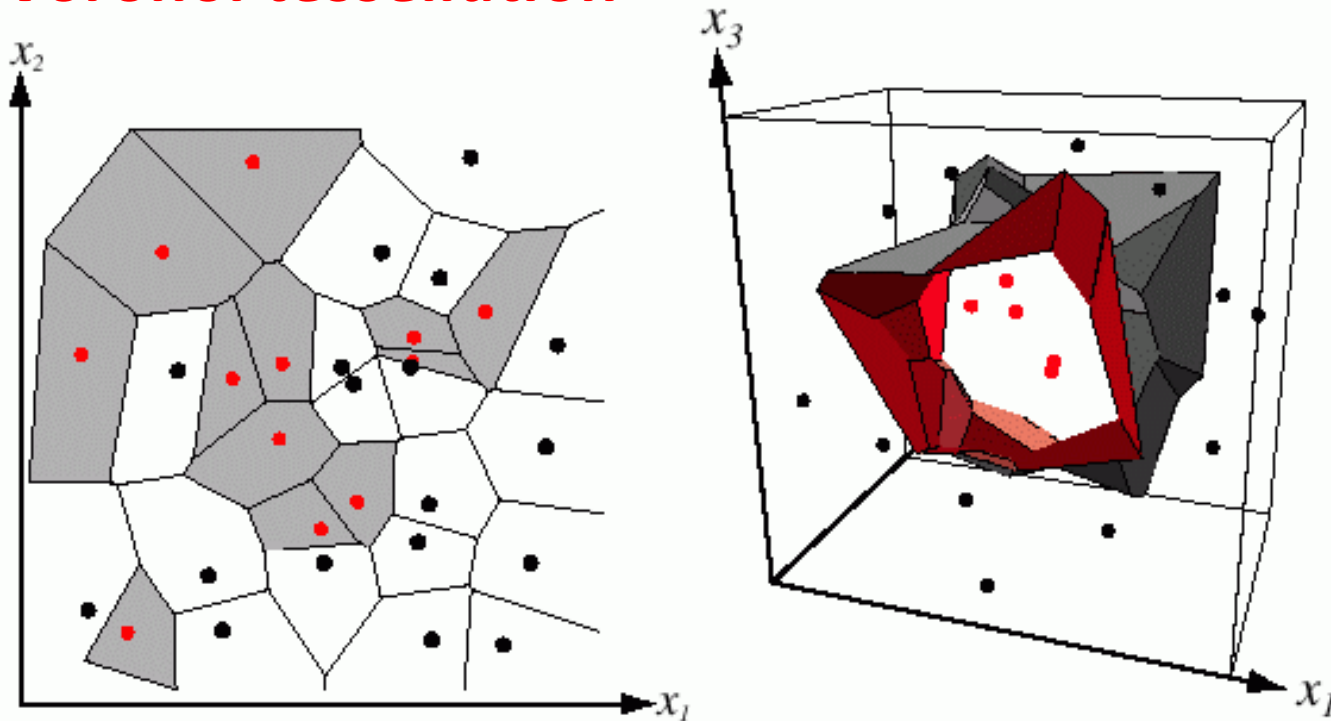
---

- Let  $D^n = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  denote a set of  $n$  labeled prototypes and let
- $\mathbf{x}' \in D^n$  be the prototype nearest to a test point  $x$ . Then
- **The Nearest Neighbor Rule:** assign the label of  $\mathbf{x}'$  to  $\mathbf{x}$ .
- This rule is suboptimal, but when the number of prototypes is **large**, its error is **never worse** than **twice the Bayes rate**.

# Classification With Nearest Neighbor Rule

---

## *Voronoi tessellation*



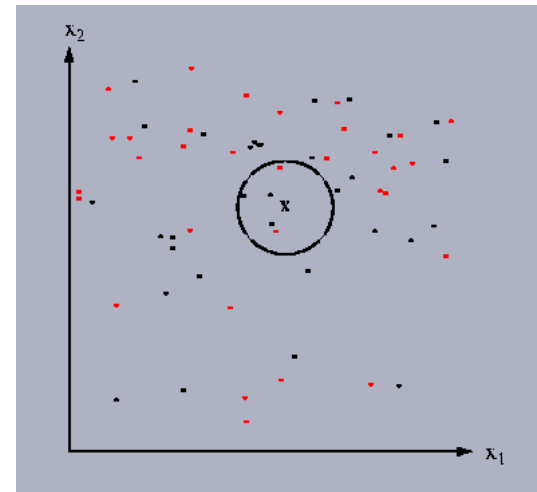
Voronoi cells

# Classification With the $k$ -Nearest Neighbor Rule

---

- The  $k$ -NN query starts at the test point and grows a spherical region until it encloses  $k$  training samples, and it labels by a majority vote of these samples.
- **Algorithm:**
  - For each sample point **Compute Distance** (sample point, test point)
  - **Sort** the distances
  - Inspect the  $k$  **smallest** distances
  - Label test point by a **majority vote**.

$$p_n(\omega_i | \mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^C p_n(\mathbf{x}, \omega_j)} = \frac{k_i}{k}$$



# Classification With the $k$ -Nearest Neighbor Rule

---

- Question.

- If a *posteriori* probabilities  $p(\omega_i | \mathbf{x})$ ,  $i = 1, 2$  for two classes are known, for example

$$p(\omega_1 | \mathbf{x}) > p(\omega_2 | \mathbf{x})$$

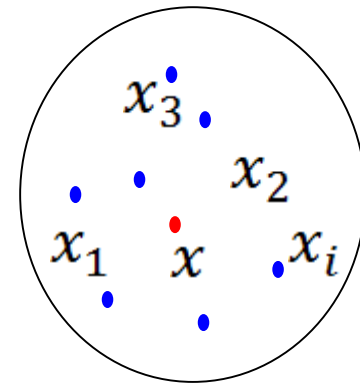
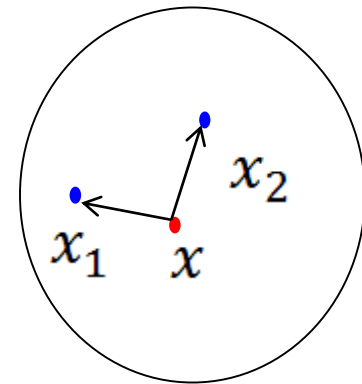
- What is a probability of choosing a class  $\omega_1$  for  $x$  with the Bayes, the nearest neighbor, the  $k$ -NN classifiers?

# Classification With the k-Nearest Neighbor Rule.

---

## ■ Answer

- Bayes: always  $w_1$
- NN:  $p(w_1 | x)$
- K-nn:  $\sum_{i=(k+1)/2}^k \binom{k}{i} p(\omega_1 | \mathbf{x})^i (1 - p(\omega_1 | \mathbf{x}))^{k-i}$



# Exercise

---

한 해안가에서 연어가 잡힐 확률은 0.6 이고 농어가 잡힐 확률은 0.4 이다.

잡힌 연어 중 40 cm 이하의 크기일 확률은 20%이고,

농어 중 40cm 이하일 확률은 3%이다.

잡은 고기가 40cm 이하 일 때 연어로 분류할 확률을 각각

Bayes, NN, K-NN(K=9) Classifier 에서 구하여라.

---

## 답안

---

$$P(\text{연어}) = 0.6, P(\text{농어}) = 0.4$$

$$P(40\text{cm이하}|\text{연어}) = 0.2, P(40\text{cm이하}|\text{농어}) = 0.03$$

Posteriori probability  $P(\text{연어}|40\text{cm이하})$  와  $P(\text{농어}|40\text{cm이하})$  를 구하면 다음과 같다.

$$P(\text{연어}|40\text{cm이하}) = \frac{P(40\text{cm이하}|\text{연어})P(\text{연어})}{P(40\text{cm이하})} = \frac{0.2*0.6}{0.6*0.2+0.4*0.03} = 90.9\%$$

$$P(\text{농어}|40\text{cm이하}) = \frac{P(40\text{cm이하}|\text{농어})P(\text{농어})}{P(40\text{cm이하})} = \frac{0.03*0.4}{0.6*0.2+0.4*0.03} = 9.09\%$$

이 posteriori probability를 이용하여 연어로 판정할 확률을 구하면 다음과 같다.

(1) Baye Classifier 의 경우 항상 연어로 분류한다. (100%)

(2) NN의 경우  $P(\text{연어}|40\text{cm이하}) = 90.9\%$  확률로 연어로 분류한다.

(3)  $K=9$ 일 때,  $\sum_{i=5}^9 \binom{9}{i} P(\text{연어}|40\text{cm이하})^i (1 - P(\text{연어}|40\text{cm이하}))^{9-i}$  를 구한다. 따라서

$$14 * (0.909)^5 * (0.0909)^4 + 84 * (0.909)^6 * (0.0909)^3 + 36 * (0.909)^7 * (0.0909)^2 + 9 * (0.909)^8 * (0.0909)^1 + (0.909)^9 * (0.0909)^0 = 0.9937837$$

약 99.4%의 확률로 연어로 분류한다.

---

# Gaussian Mixture Estimation

---

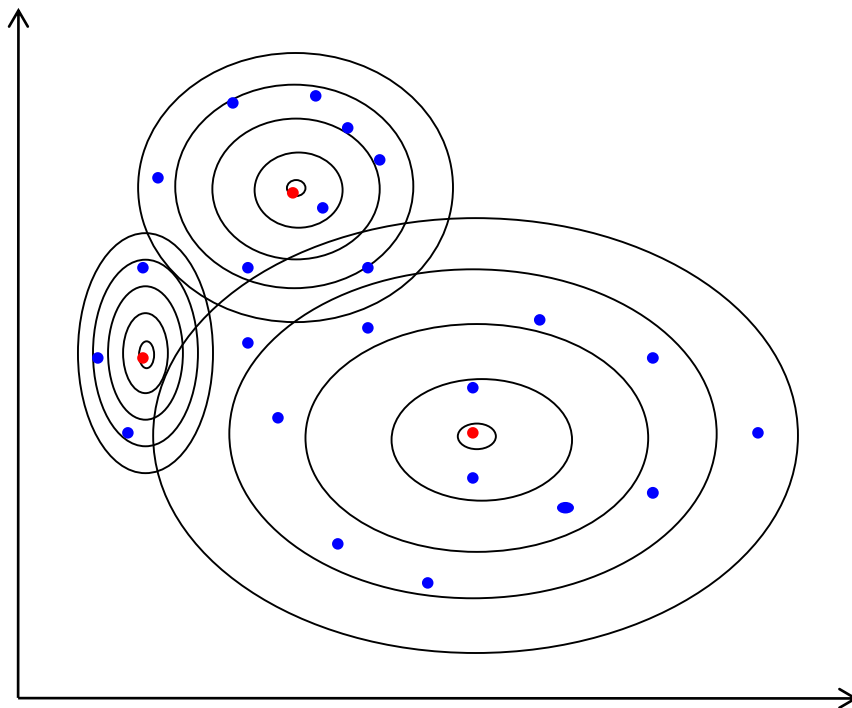
- Parzen Window

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

- Gaussian Mixture

$$p(\mathbf{x}) = \sum_{k=1}^K w_k \varphi\left(\frac{\mathbf{x} - \mu_k}{\sigma_k}\right).$$

$$p(\mathbf{x}) = \sum_{k=1}^K p(\mathbf{x} | \theta_k) p(\theta_k).$$



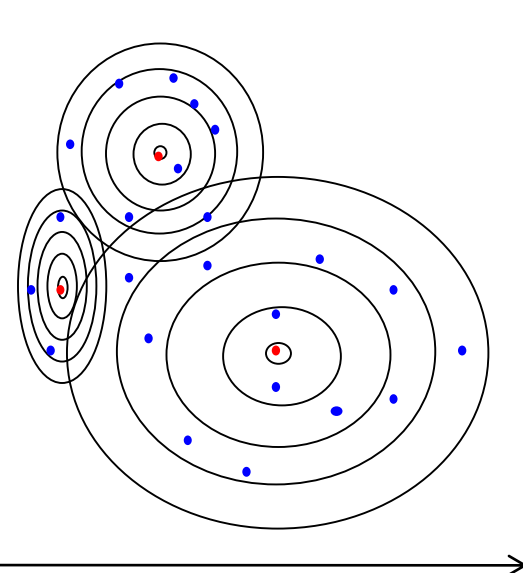
# Gaussian Mixture Estimation

---

- Gaussian Mixture Model  $p(\mathbf{x}) = \sum_{k=1}^K p(\mathbf{x} | \theta_k) p(\theta_k)$ .  
where ,  $p(x|\theta_k)$  are Gaussian density functions.
- $K, \theta_k$ , and  $p(\theta_k)$  can be estimated from a data set using Expectation-Maximization (EM) algorithm
- Example of EM:  $\theta_k = [\mu_k, \Sigma_k]$ , fixed  $K$

$$p(\mathbf{x}) = \sum_{k=1}^K w_k \varphi\left(\frac{\mathbf{x} - \mu_k}{\sigma_k}\right).$$

- Class conditional PDF
- $p(\mathbf{x}|\theta) = \sum_k p(\mathbf{x}|\theta_k)p(\theta_k|\theta)$   
 $= \sum_Z p(\mathbf{x}, Z|\theta) = \sum_{Z=k} p(\mathbf{x}|Z = k, \theta)p(Z = k|\theta)$
- $p(\mathbf{x}|\theta) = \int_Z p(\mathbf{x}, Z, |\theta) d_Z$



# Expectation-Maximization (EM)

- EM aims to find parameter values that maximize likelihood,

$$L(\theta; X) = p(X | \theta) = \sum_Z p(X, Z | \theta) = \sum_Z L(\theta; X, Z),$$

$$L(\theta; X) = p(X | \theta) = \int_Z p(X, Z | \theta) dz = \int_Z L(\theta; X, Z) dz$$

$$p(\mathbf{x} | \theta) = \sum_{k=1}^K p(\mathbf{x} | \theta_k) p(\theta_k).$$

$$p(\mathbf{x} | \theta) = \sum_{k=1}^K w_k \varphi\left(\frac{\mathbf{x} - \mu_k}{\sigma_k}\right).$$

where  $Z$  is latent variable.

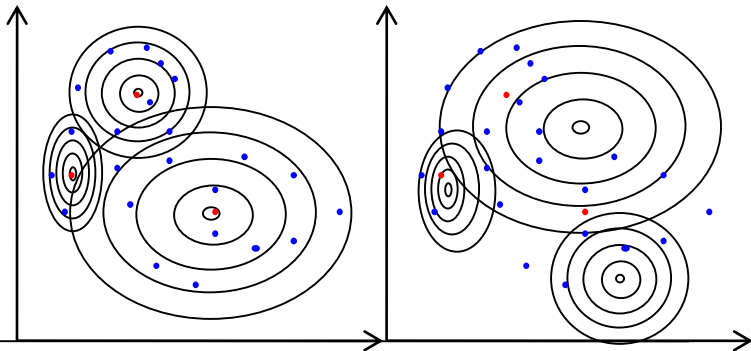
- E-step:** For given  $\theta^t$ ,  $X$ , find expectation of the likelihood on the conditional distribution of  $Z$  given  $X$  and  $\theta^t$ .

$$Q(\theta | \theta^t) = E_{Z|X, \theta^t} [\log L(\theta; X, Z)] = \sum_Z p(Z | X, \theta^t) \log L(\theta; X, Z)$$

- M-step:** Find  $\theta^{t+1}$  maximizing  $Q$ .

$$\theta^{t+1} = \arg \max_{\theta} Q(\theta | \theta^t)$$

- Repeat E-step and M-step.



# Expectation-Maximization (EM)

---

- Likelihood of GMM

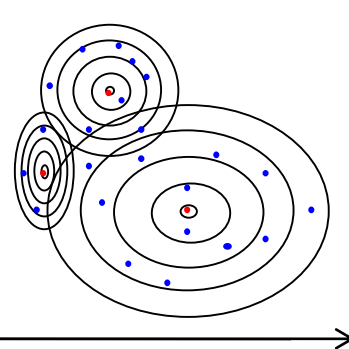
$$\begin{aligned} L(\theta; X) &= p(X | \theta) = \sum_Z p(X, Z | \theta) \left( := \sum_Z L(\theta; X, Z) \right) = \sum_Z p(X | Z, \theta) p(Z | \theta) \\ &= \sum_{Z=k} p(X | Z = k, \theta) p(Z = k | \theta) = \sum_{k=1}^K \prod_{m=1}^M p(x_m | \theta_k) p(\theta_k) = \sum_{k=1}^K \prod_{m=1}^M p(x_m; \mu_k, \Sigma_k) p(Z = k) \end{aligned}$$

$$\begin{aligned} L(\theta; X = x_m, Z = k) &= p(x_m; \mu_k, \Sigma_k) \tau_k \\ &= \exp \left( \log \tau_k - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x_m - \mu_k)^T \Sigma_k^{-1} (x_m - \mu_k) \right) \end{aligned}$$

$$\begin{aligned} \log L(\theta; X, Z = k) &= \log \prod_m L(\theta; X = x_m, Z = k) \\ &= \sum_m \log L(\theta; X = x_m, Z = k) \end{aligned}$$

- **E-step**

$$Q(\theta | \theta^t) = \sum_m \sum_{Z=k} p(Z = k | X = x_m, \theta^t) \log L(\theta; X = x_m, Z = k)$$



# Expectation-Maximization (EM)

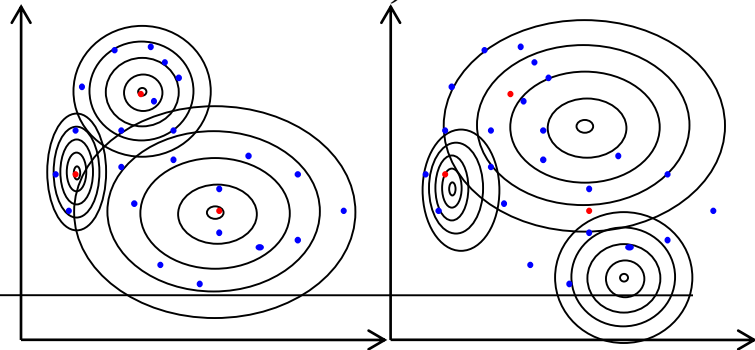
- **E-step**

$$Q(\theta | \theta^t) = \sum_m \sum_k p(Z = k | X = x_m, \theta^t) \log L(\theta; X = x_m, Z = k)$$

$$\log L(\theta; X = x_m, Z = k) = \left( \log \tau_k - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x_m - \mu_k)^T \Sigma_k^{-1} (x_m - \mu_k) \right)$$

$$T_{k,m}^t := p(Z = k | X = x_m, \theta^t) = \frac{p(X = x_m, \theta^t | Z = k) p(Z = k)}{\sum_k p(X = x_m, \theta^t | Z = k) p(Z = k)} = \frac{p(x_m; \mu_k^t, \Sigma_k^t) \tau_k^t}{\sum_k p(x_m; \mu_k^t, \Sigma_k^t) \tau_k^t}$$

$$Q(\theta | \theta^t) = \sum_m \sum_k T_{k,m}^t \left( \log \tau_k - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x_m - \mu_k)^T \Sigma_k^{-1} (x_m - \mu_k) \right)$$



# Expectation-Maximization (EM)

- **M-step**

$$Q(\theta | \theta^t) = \sum_m \sum_k T_{k,m}^t \left( \log \tau_k - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x_m - \mu_k)^T \Sigma_k^{-1} (x_m - \mu_k) \right)$$

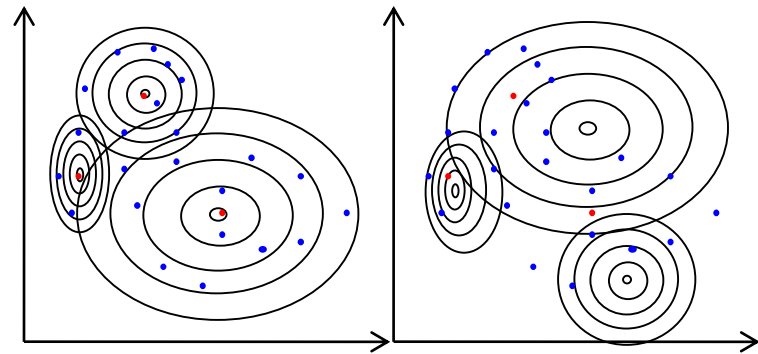
$$\tau^{t+1} = \arg \max_{\tau} \sum_m \sum_k T_{k,m}^t \log \tau_k, \quad \text{Subject to } \sum \tau_k = 1$$

$$(\mu_k^{t+1}, \Sigma_k^{t+1}) = \arg \max_{(\mu_k, \Sigma_k)} \sum_m T_{k,m}^t \left( -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x_m - \mu_k)^T \Sigma_k^{-1} (x_m - \mu_k) \right)$$

$$\tau_k^{t+1} = \frac{\sum_m T_{k,m}^t}{\sum_k \sum_m T_{k,m}^t}$$

$$\mu_k^{t+1} = \frac{\sum_m T_{k,m}^t x_m}{\sum_m T_{k,m}^t}$$

$$\Sigma_k^{t+1} = \frac{\sum_m T_{k,m}^t (x_m - \mu_k^{t+1})(x_m - \mu_k^{t+1})^T}{\sum_m T_{k,m}^t}$$



# Expectation-Maximization (EM)

- EM aims to find parameter values that maximize likelihood,

$$L(\theta; X) = p(X | \theta) = \sum_Z p(X, Z | \theta) = \sum_Z L(\theta; X, Z),$$

$$L(\theta; X) = p(X | \theta) = \int_Z p(X, Z | \theta) dz = \int_Z L(\theta; X, Z) dz$$

$$p(\mathbf{x} | \theta) = \sum_{k=1}^K p(\mathbf{x} | \theta_k) p(\theta_k).$$

$$p(\mathbf{x} | \theta) = \sum_{k=1}^K w_k \varphi\left(\frac{\mathbf{x} - \mu_k}{\sigma_k}\right).$$

where  $Z$  is latent variable.

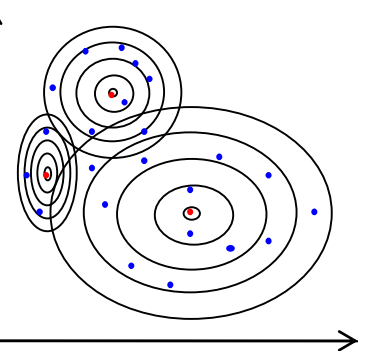
- E-step:** For given  $\theta^t$ ,  $X$ , find expectation of the likelihood on the conditional distribution of  $Z$  given  $X$  and  $\theta^t$ .

$$Q(\theta | \theta^t) = E_{Z|X, \theta^t} [\log L(\theta; X, Z)] = \sum_Z p(Z | X, \theta^t) \log L(\theta; X, Z)$$

- M-step:** Find  $\theta^{t+1}$  maximizing  $Q$ .

$$\theta^{t+1} = \arg \max_{\theta} Q(\theta | \theta^t)$$

- Repeat E-step and M-step.

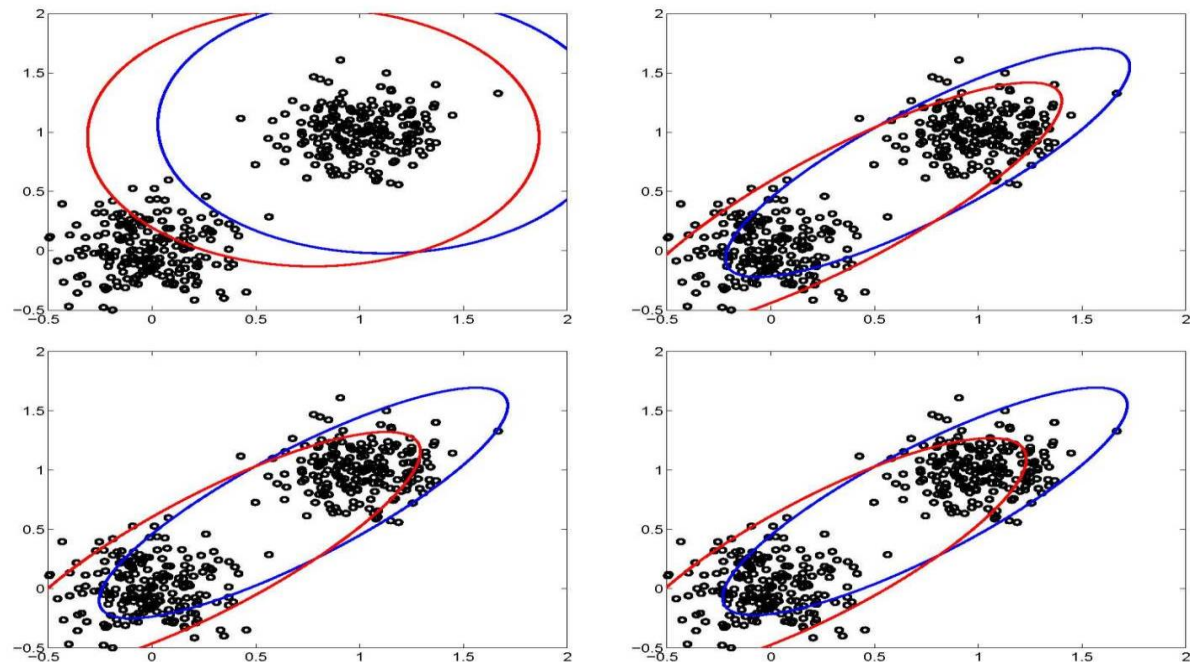


# Example

$$\tau_k^{t+1} = \frac{\sum_m T_{k,m}^t}{\sum_k \sum_m T_{k,m}^t} \quad \mu_k^{t+1} = \frac{\sum_m T_{k,m}^t x_m}{\sum_m T_{k,m}^t}$$

$$\Sigma_k^{t+1} = \frac{\sum_m T_{k,m}^t (x_m - \mu_k^{t+1})(x_m - \mu_k^{t+1})^T}{\sum_m T_{k,m}^t}$$

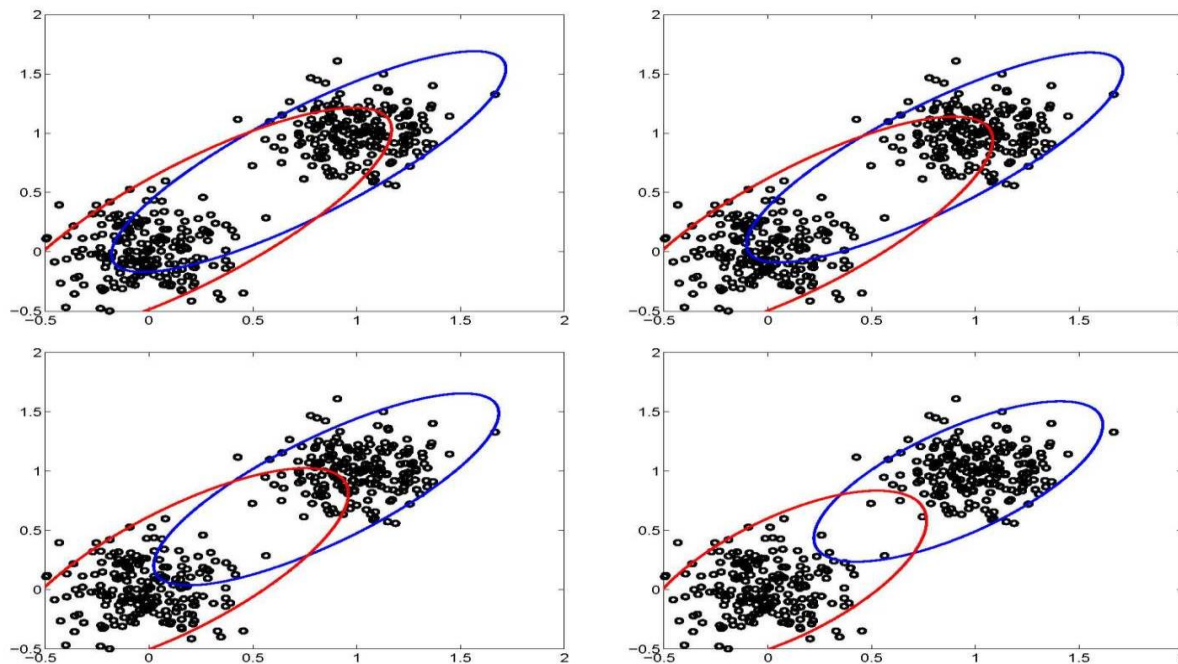
## Mixture density estimation: example



# Example

---

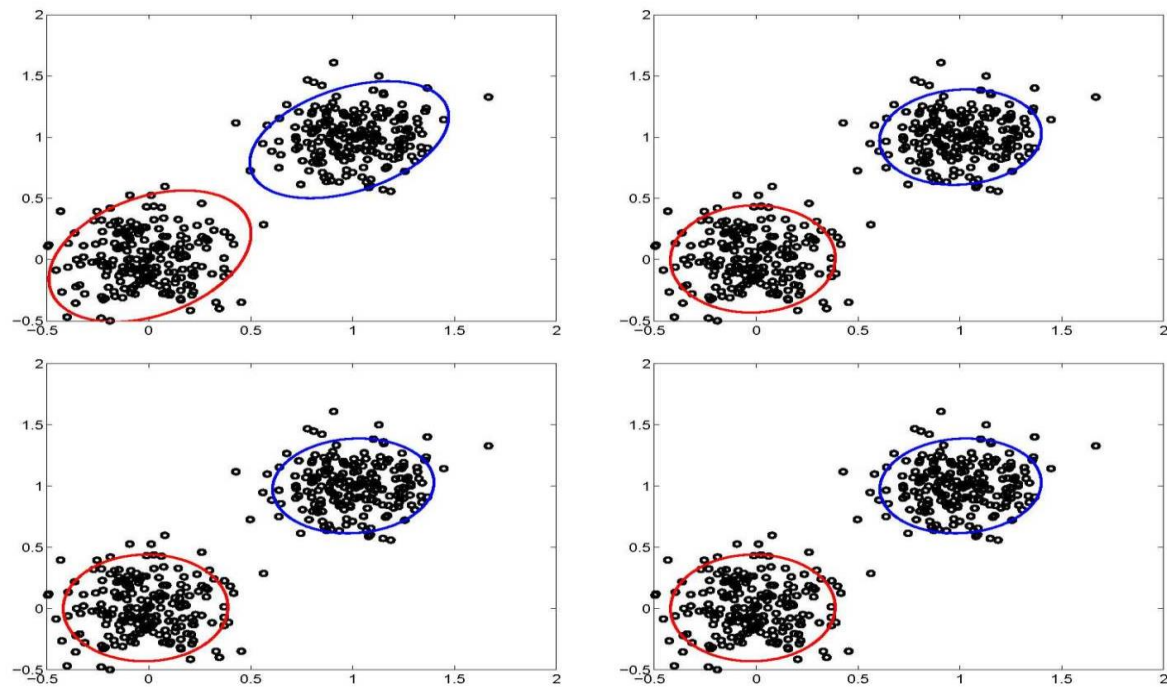
## Mixture density estimation



# Example

---

## Mixture density estimation



# Automatic Model Order Selection

---

- **Constructive algorithm** can be used to **automatically determine the number** of components in a mixture.
  - The available data set is partitioned into disjoint **training and validation sets**.
  - A mixture model is **initialized** with a small number of components, **typically one**.
  - The likelihood for the validation set is computed after every iteration.
  - The tasks of **selecting and splitting components** can be performed by utilizing the **likelihood for validation set**
-

# Automatic Model Order Selection

---

- For each component  $k$ , define a total responsibility  $r(k)$  as

$$r(k) = \sum_{m=1}^M p(\theta_k | x_m) = \sum_{m=1}^M \frac{p(x_m | \theta_k) p(\theta_k)}{\sum_{j=1}^K p(x_m | \theta_j) p(\theta_j)}$$

- The component with the lowest total responsibility for the validation set is selected for splitting.
- Covariance matrices equal to  $C_k$
- Means given by:

$$\mu = \mu_k \pm \frac{\sqrt{\lambda_1}}{2} u_1$$

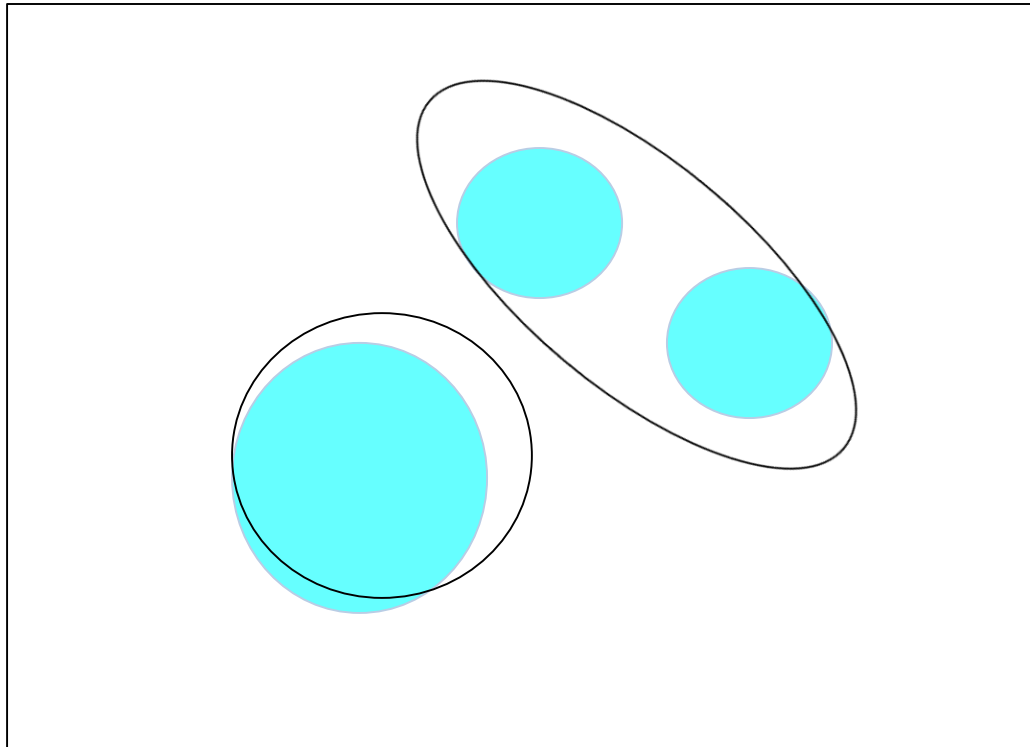
where  $\lambda_1$  is the largest eigenvalue of  $C_k$  and  $u_1$  is the corresponding eigenvector.

---

# Automatic Model Order Selection

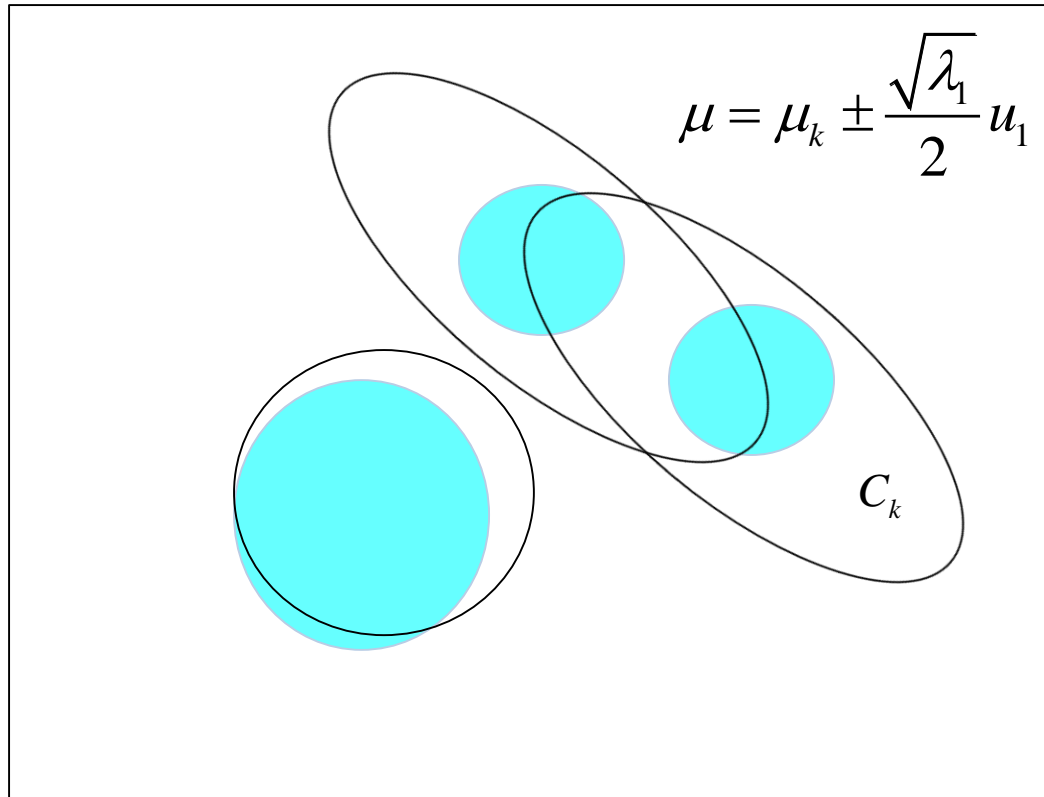
---

$$r(k) = \sum_{m=1}^M p(\theta_k | x_m) = \sum_{m=1}^M \frac{p(x_m | \theta_k) P(\theta_k)}{\sum_{j=1}^K p(x_m | \theta_j) P(\theta_j)}$$



# Automatic Model Order Selection

---



# Automatic Model Order Selection

---

- Prior probabilities for the new components are set to  $\frac{1}{2} p(\theta_k)$
  - $K_i$  denote the number of components in a model after iteration  $i$
  - $L_i$  be the likelihood of the validation set given the model
    1. Apply EM for model with  $K_i$  components.
    2. Compute  $L_i$  for validation set
    3. If  $(L_i - L_{i-1} \leq \varepsilon)$  STOP.
    4. Split component  $k$  with the lowest total responsibility
    5. Set  $K_{i+1} = K_i + 1$  and  $i = i + 1$
    6. Go to 1.
-

# Adaptive EM for Non-stationary Distributions

---

- In the context of dynamic vision, data are often sampled from non-stationary distributions.
- For example, the color of the object often changes gradually over time
- An algorithm for adaptively estimating such a mixture.
  - At each frame,  $t$ , a new set of data,  $X^{(t)}$  can be used to update the mixture model.
  - Let  $r^{(t)}$  denote the sum of the posterior probabilities

$$r^{(t)} = \sum_{x \in X^{(t)}} p(\theta^{(t-1)} | x)$$

---

# Adaptive EM for Non-stationary Distributions

---

- The parameters are first estimated for each mixture component,  $k$ , using only the new data,  $x \in X^{(t)}$

$$\mu^{(t)} = \frac{\sum p(\theta^{(t-1)} | x) x}{r^{(t)}},$$

$$C^t = \frac{\sum p(\theta^{(t-1)} | x) [x - \mu^{(t-1)}] [(x - \mu^{(t-1)})]^T}{r^{(t)}}$$

---

# Interim Summary

---

- $K_n$ -Nearest Neighbor Method
  - Nearest Neighbor Method
  - $k$  -Nearest Neighbor Method
  - $k$  -Nearest Neighbor Classifier
  - Bayes, the nearest neighbor, the k-NN classifiers
  - Gaussian Mixture Model
  - EM Algorithm
  - Automatic GMM selection
  - Adaptive GMM
-