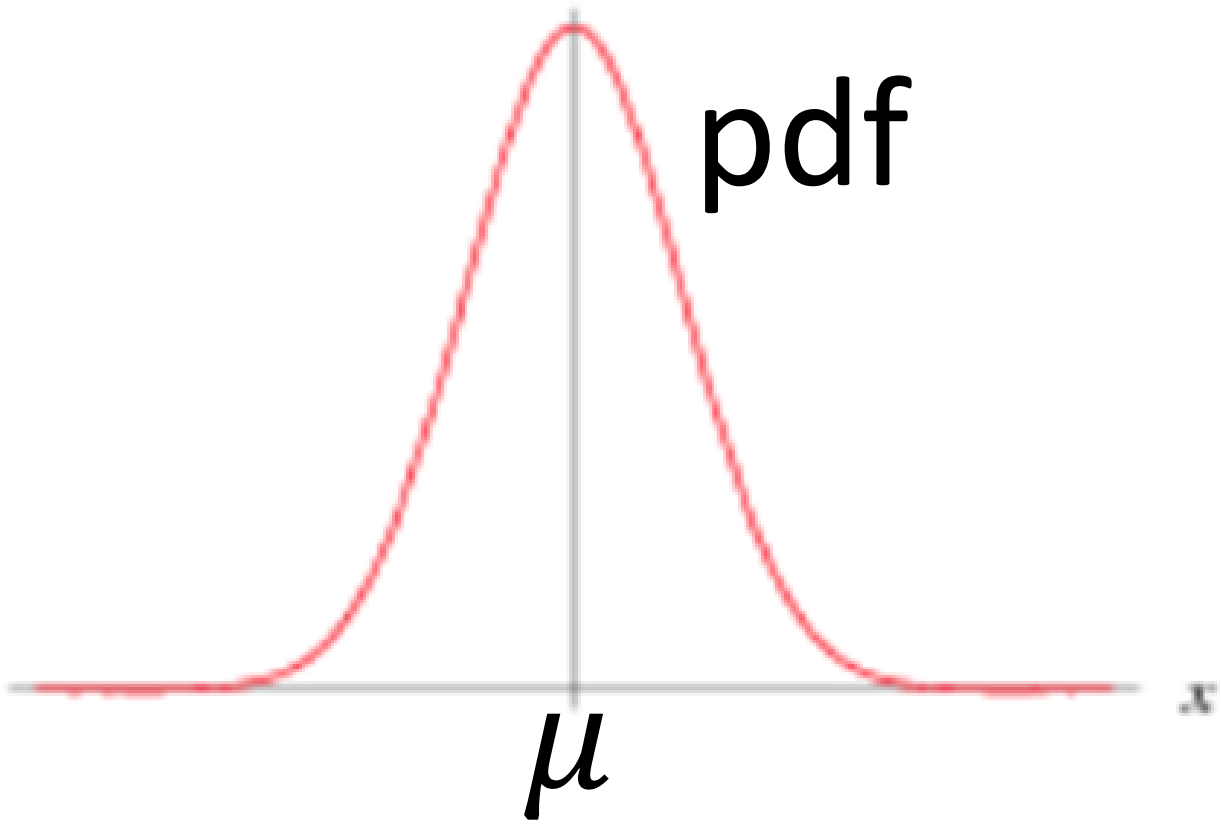
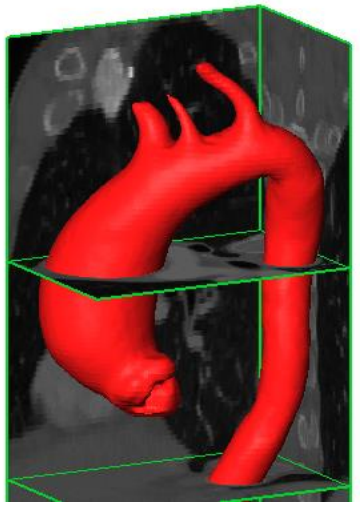
1-dimensional (1D) Gaussian Distribution

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normal Distribution



$\{x_1, x_2, x_3, \ldots, x_N\}$ is a set of data samples collected from a hospital
x_n is the aorta size of a patient



Patient Population
aorta size : 2 ~ 6 cm
aorta size distribution

x_1 is the aorta size of the patient #1
x_2 is the aorta size of the patient #2
x_3 is the aorta size of the patient #3
…

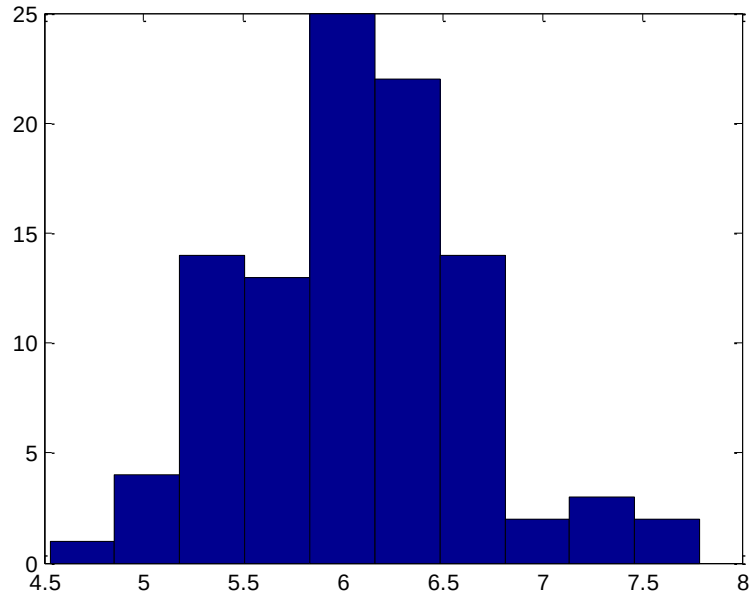
There could be nothing random about the aorta size and aneurysm of a patient
- the disease process could be deterministic

The data distribution can be model by a probability distribution (e.g. Gaussian)
In order to use the data distribution, we treat aorta size as random variable

disease group (ruptured)

subject id	diameter
1	5.5
2	4.6
3	3.2
4	4.2
5	6.0

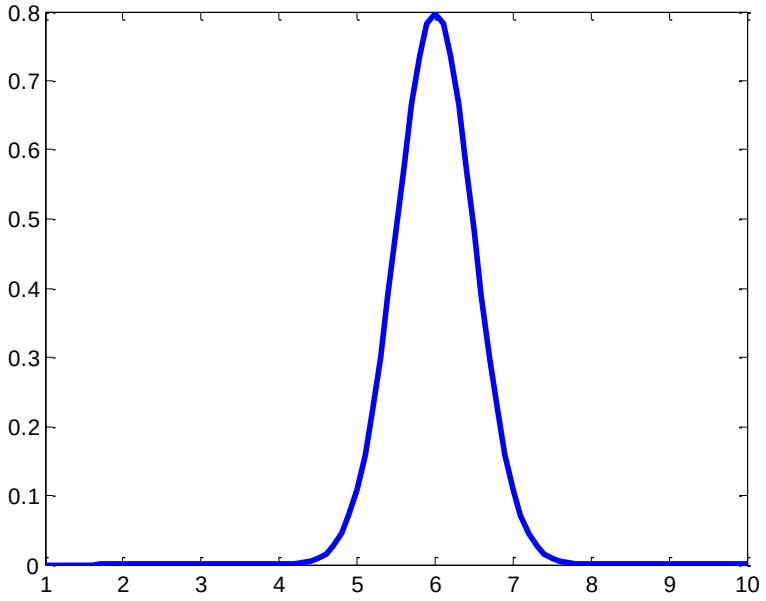
assume we have 100 subjects in this group



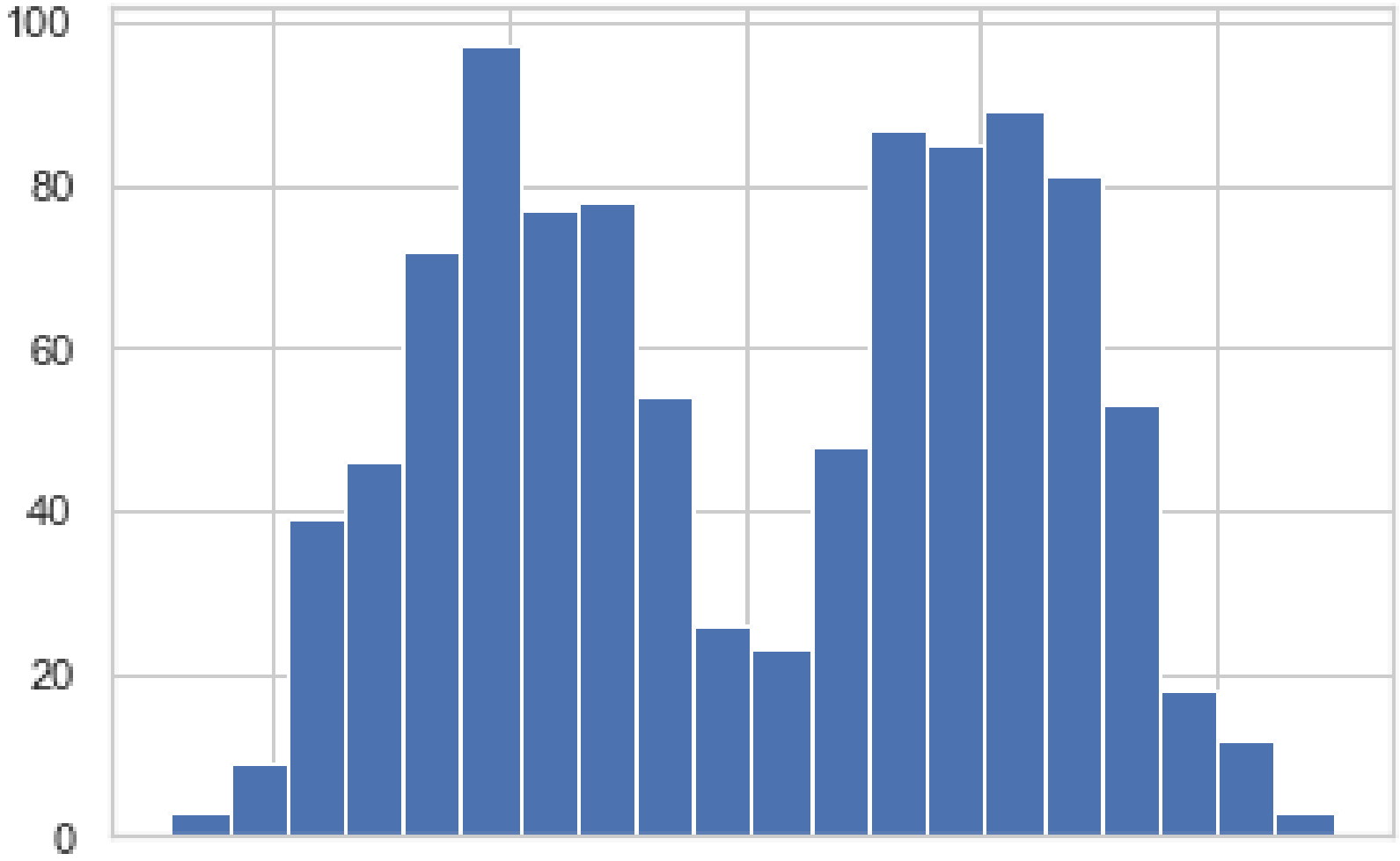
After we check the data, we guess that it may has a Gaussian distribution.

We use Gaussian PDF as the model of data distribution for machine learning.

Gaussian PDF



What if the distribution is not a simple Gaussian ?



Gaussian Mixture Model (GMM)

Liang Liang

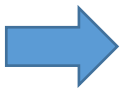
What does this mean?

$\{x_1, x_2, x_3, \ldots, x_N\}$ is a set of data samples, and $x_n \in \mathcal{R}^M$

x_n is an observed value /realization of a random variable X_n

random variables $\{X_1, X_2, X_3, \ldots, X_N\}$ are i.i.d. (**i**ndependently and **i**dentically **d**istributed)

Random Number/Vector Generator
PDF: $f_X(x)$



X

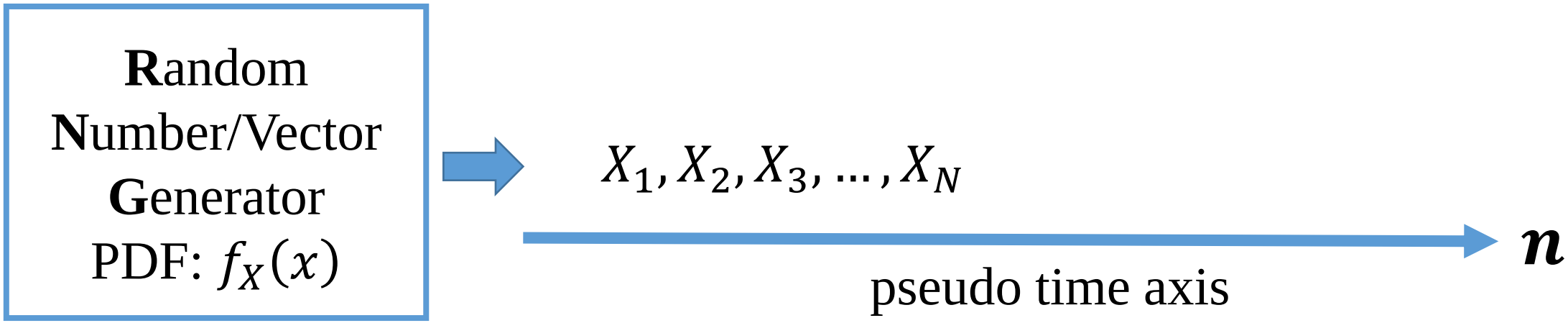
x1 is an observed value of X
x2 is an observed value of X
x3 is an observed value of X
we say x1, x2, x3,… are generated from $f_X(x)$

Where are those r.v. ?

$\{X_1, X_2, X_3, \ldots, X_N\}$

```
In [1]:
1 import numpy as np

In [2]:
1 rng = np.random.RandomState()

In [3]:
1 x1 = rng.randn()
2 x1

Out[3]: 0.554514177029439 6

In [4]:
1 x2 = rng.randn()
2 x2

Out[4]: -0.7736950982665748

In [5]:
1 x3 = rng.randn()
2 x3

Out[5]: 0.5774013368938337
```

Re-think the process from a different perspective



Task: Use the **rng** to generate a sequence of random numbers

After running Python code, we see a sequence of numbers coming out of **rng**

x1 is an observed value of X_1
x2 is an observed value of X_2
x3 is an observed value of X_3

$\{X_1, X_2, X_3\}$ are **i.i.d.** because they come from the same PDF
and they are generated independently

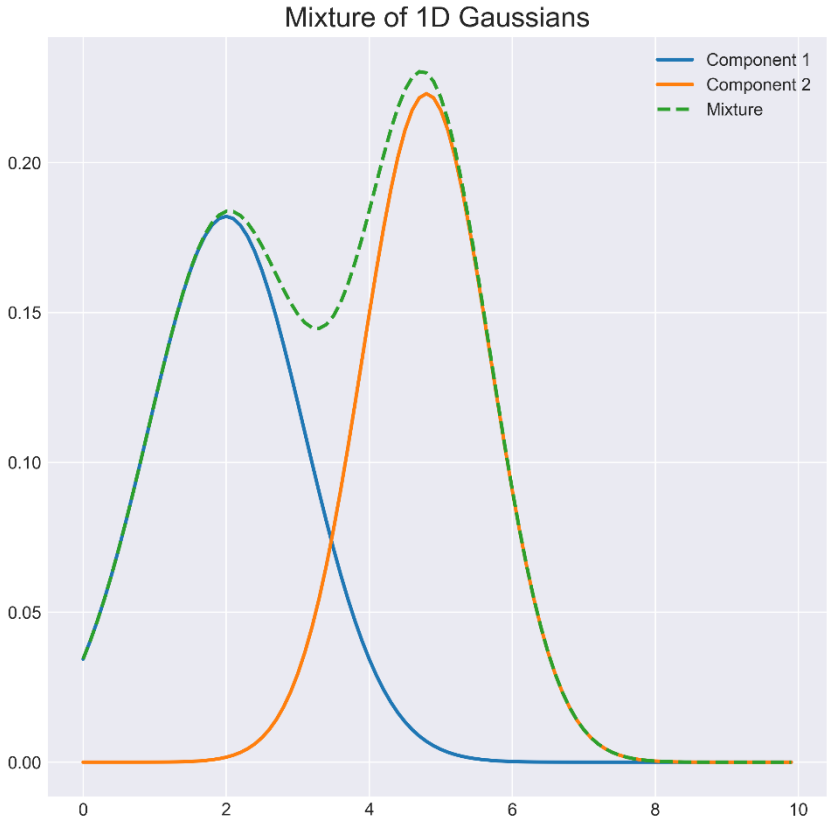
Generative Model

- Assumption in machine learning:
  data samples are generated from a probability distribution
- A model that can generate data samples, is a generative model
- GMM is a probability distribution and a generative model

1D Gaussian Mixture Model

- Mixture of two Gaussians in 1D

$$f_X(x) = \pi_1 \frac{1}{\sqrt{2\pi\sigma_1}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \pi_2 \frac{1}{\sqrt{2\pi\sigma_2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$



There are two components/clusters

a r.v. X may belong to cluster_1 or cluster_2

$\pi_1 = P(\{X \in cluster_1\})$ prior probability

$\pi_2 = P(\{X \in cluster_2\})$ prior probability

$\pi_1 + \pi_2 = 1$

$f_X(x)$ is the PDF of the random variable X

cluster-1: $\mathcal{N}(x|\mu_1, \sigma_1^2) = p_X(x|X \in cluster_1)$

cluster-2: $\mathcal{N}(x|\mu_2, \sigma_2^2) = p_X(x|X \in cluster_2)$

https://angusturner.github.io/generative_models/2017/11/03/pytorch-gaussian-mixture-model.html

See the demo: 1D_GMM.ipynb

Estimate the parameters of 1D GMM

- GMM with two 1D Gaussian components, the PDF is

$$f_X(x) = \pi_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \pi_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

- The GMM (PDF) has six **parameters**: $\pi_1, \mu_1, \sigma_1, \pi_2, \mu_2, \sigma_2$
- Given a set of data samples, we fit the GMM model to the data.
- **Model fitting/training** is to estimate/recover the model parameters

Maximum Likelihood (ML) Estimation

- $\{x_1, x_2, x_3, \ldots, x_N\}$ is a set of data samples, and $x_n \in \mathcal{R}^M$
- Assume x_n is an observed value of a random variable X_n, and the r.v. $\{X_1, X_2, X_3, \ldots, X_N\}$ are i.i.d. (independently and identically distributed)
- $f_{X_n}(x)$ is the PDF of X_n i.i.d. => $f(x) = f_{X_1}(x) = \cdots = f_{X_N}(x)$
- $f(x)$ has some parameters but the parameter values are unknown
- The goal is to estimate the parameters of $f(x)$ from the data samples

Maximum Likelihood (ML) Estimation

- The joint PDF is

$$f_{X_1,X_2,\ldots,X_N}(x_1, x_2, \ldots, x_N) = f_{X_1}(x_1) f_{X_2}(x_2) \ldots f_{x_N}(x_N) = \prod_{n=1}^{N} f(x_n)$$

- The **NLL loss function** of the parameters is

$$loss = -log f_{X_1,\ldots,X_N}(x_1, \ldots, x_N) = -\sum_{n=1}^{N} log f(x_n)$$

Maximum Likelihood (ML) Estimation

- The loss function of the parameters is

$$loss = -logf_{X_1,\dots,X_N}(x_1, \dots, x_N) = -\sum_{n=1}^{N} logf(x_n)$$

- $x_1, \dots, x_N$ are data points
- $-logf_{X_1,\dots,X_N}(x_1, \dots, x_N)$ is the **Negative Log Likelihood (NLL)**
- The goal is to maximize the log Likelihood (to minimize NLL) to obtain the optimal parameters of $f(x)$
- Why Max Likelihood ?

We want to find the best parameters such that the data samples are highly likely coming/generated from the PDF $f(x)$

The observed data samples are most probable under the assumed PDF

Maximum Likelihood (ML) Estimation of 1D Gaussian

- Let's assume the PDF is a 1D Gaussian

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The NLL loss function is

$$loss = L(\mu, \sigma) = -\sum_{n=1}^{N} log f(x_n) = \sum_{n=1}^{N} \frac{(x_n - \mu)^2}{2\sigma^2} + N log\left(\sqrt{2\pi}\sigma\right)$$

The loss reaches the minimum when $\frac{\partial L}{\partial \mu} = 0$ and $\frac{\partial L}{\partial \sigma} = 0$ (necessary condition)

From $\frac{\partial L}{\partial \mu} = 0$, we get $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$

From $\frac{\partial L}{\partial \sigma} = 0$, we get $\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$

GMM for Clustering 1D data

- GMM with two Gaussian components, the PDF is

$$f_X(x) = \pi_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \pi_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

- Each component corresponds to a cluster
- After model fitting, we do **clustering**: assign each data point to a cluster
- For the data point $x = x_n$,

if $\pi_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} > \pi_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$, then x_n is assigned to cluster-1

if $\pi_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} < \pi_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$, then x_n is assigned to cluster-2

M-dimensional (M-D) Gaussian Distribution

- $f_X(x) = \frac{1}{(2\pi)^{M/2}|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$

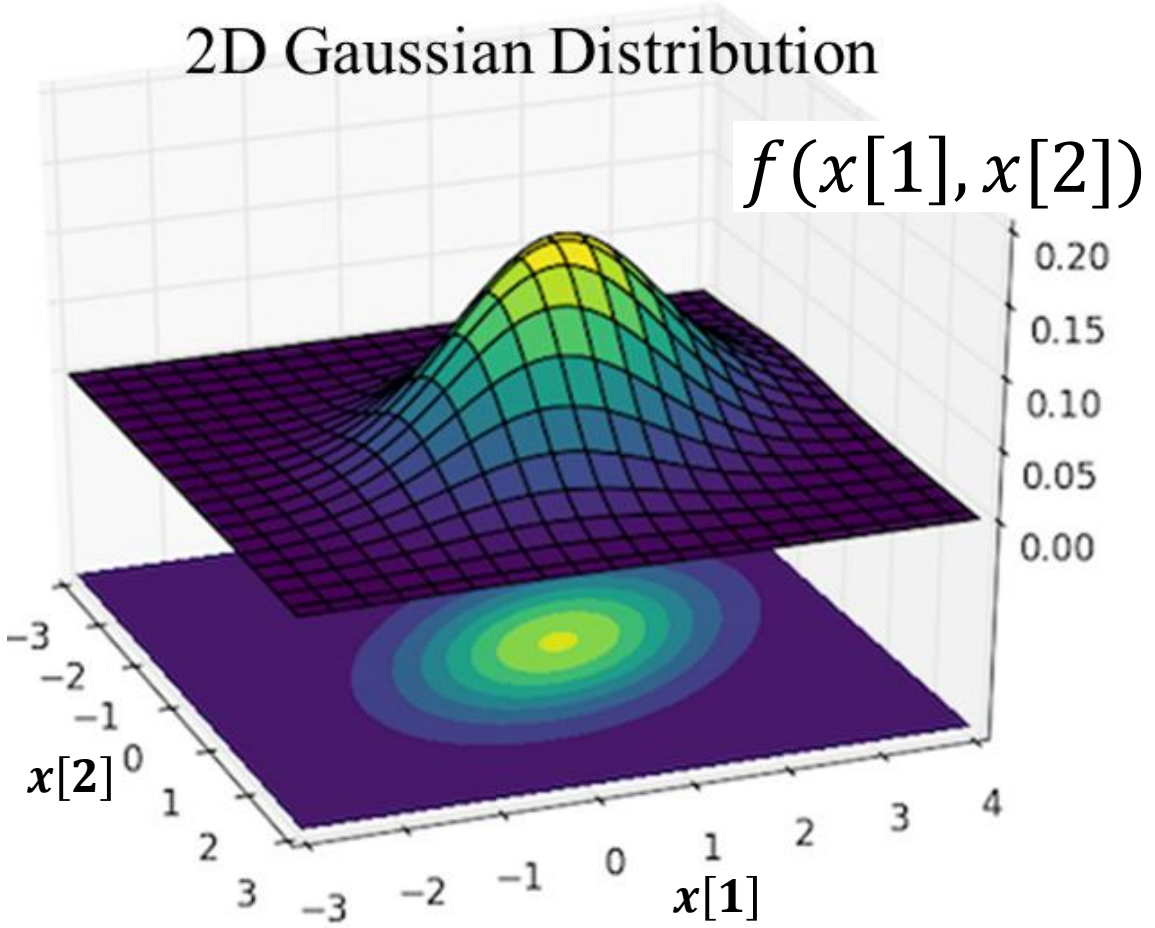
$x, \mu \in \mathcal{R}^M, \Sigma \in \mathcal{R}^{M\times M}$

$f_X(x)$

$= \mathcal{N}(\mu, \Sigma)$

$= \mathcal{N}(x; \mu, \Sigma)$

$= \mathcal{N}(x|\mu, \Sigma)$



M-D Gaussian Mixture Model

- Mixture of K Gaussians in M-dim: $x, \mu \in \mathcal{R}^M$, $\Sigma \in \mathcal{R}^{M \times M}$

$$f_X(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

$$\pi_k = P(\{X \in \ cluster_k\}), \text{and } \sum_{k=1}^{K} \pi_k = 1$$

- the GMM has K Gaussian components/clusters.
- the GMM is the PDF of a random variable X
- a data sample x is an observed value of the random variable X

Maximum Likelihood Estimation of GMM Parameters

- The loss function of the parameters is

$$loss = -logf_{X_1,\dots,X_N}(x_1, \dots, x_N) = -\sum_{n=1}^{N} logf(x_n)$$

- $x_1, \dots, x_N$ are data points
- $-logf_{X_1,\dots,X_N}(x_1, \dots, x_N)$ is **Negative Log Likelihood (NLL)**
- The goal is to maximize the log Likelihood (to minimize NLL) to obtain the optimal parameters of the PDF $f(x)$
- GMM parameters: π_k, μ_k, Σ_k to be estimated from data samples

Maximum Likelihood Estimation of GMM Parameters

- **GMM**: $f(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ where $\sum_{k=1}^{K} \pi_k = 1$
- **Goal**: estimate μ_k, Σ_k, π_k for k=1 to K, from data samples
- **Define:** $\gamma_{(n,k)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)}$ and $N_k = \sum_{n=1}^{N} \gamma_{(n,k)}$

$$\gamma_{(n,k)} = P(\{X_n \in cluster_k\}|X_n = x_n)$$

$\gamma_{(n,k)}$ is called the posterior probability

It is the probability of X_n belonging to $cluster_k$ after we observe its value x_n. In layman's term, it is the probability of x_n belonging to the $cluster_k$.

bishop- pattern recognition and machine learning

Maximum Likelihood Estimation of GMM Parameters

- $\gamma_{(n,k)} = P(\{X_n \in cluster_k\}|X_n = x_n)$
  - $\gamma_{(n,k)}$ is the probability of sample x_n belonging to $cluster_k$

    $\gamma_{(n,k)}$ is the posterior probability, soft assignment/label

- Define $z_{(n,k)}$ to be the indicator of the even $\{X_n \in cluster_k\}$

  If $z_{(n,k)} = 1$, then it is true that $X_n \in cluster_k$

  If $z_{(n,k)} = 0$, then it is false that $X_n \in cluster_k$

  $z_{(n,k)}$ is the (hard) cluster label/indicator of x_n

bishop- pattern recognition and machine learning

Maximum Likelihood Estimation of GMM Parameters

$$\gamma_{(n,k)} = P(\{X_n \in cluster_k\}|X_n = x_n)$$

$z_{(n,k)}$ is the binary indicator of the even $\{X_n \in cluster_k\}$

$$\gamma_{(n,k)} = p\left(z_{(n,k)} = 1\middle|X_n = x_n\right)$$

Let's simplify the notations:

$$\gamma_{(n,k)} = p(z_{(n,k)}|x_n)$$

It is the probability of x_n belonging to cluster-k

bishop- pattern recognition and machine learning

Maximum Likelihood Estimation of GMM Parameters

- GMM: $f(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ where $\sum_{k=1}^{K} \pi_k = 1$

$$\gamma_{(n,k)} = p\left(z_{(n,k)}|x_n\right) = \frac{p(z_{(n,k)}, x_n)}{p(x_n)} = \frac{p(z_{(n,k)}, x_n)}{f(x_n)}$$

assume $f(x)$ is the true PDF

$$p\left(z_{(n,k)}, x_n\right) = p\left(z_{(n,k)}\right)p\left(x_n|z_{(n,k)}\right)$$

$$p\left(z_{(n,k)}\right) \triangleq \pi_k, \text{ and } p\left(x_n|z_{(n,k)}\right) \triangleq \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

$$\gamma_{(n,k)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

bishop- pattern recognition and machine learning

Maximum Likelihood (ML) Estimation of GMM Parameters

- The NLL loss function of the parameters is

$$L = -\sum_{n=1}^{N} log f(x_n)$$

- GMM: $f(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ where $\sum_{k=1}^{K} \pi_k = 1$
- To obtain μ_k: $\frac{\partial L}{\partial \mu_k} = 0$, then we have

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{(n,k)} x_n$$

where $N_k = \sum_{n=1}^{N} \gamma_{(n,k)}$

bishop- pattern recognition and machine learning

Maximum Likelihood Estimation of GMM Parameters

- The ML loss (NLL) function of the parameters is

$$L = -\sum_{n=1}^{N} log f(x_n)$$

- GMM: $f(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ where $\sum_{k=1}^{K} \pi_k = 1$
- To obtain Σ_k: $\frac{\partial L}{\partial \Sigma_k} = 0$, then we have

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{(n,k)} (x_n - \mu_k)(x_n - \mu_k)^T$$

Σ_k is called covariance matrix

where $N_k = \sum_{n=1}^{N} \gamma_{(n,k)}$

bishop- pattern recognition and machine learning

Maximum Likelihood Estimation of GMM Parameters

- The ML loss (NLL) function of the parameters is

$$L = -\sum_{n=1}^{N} log f(x_n)$$

- GMM: $f(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$ where $\sum_{k=1}^{K} \pi_k = 1$
- To obtain π_k: we add a new term to L *because* $\sum_{k=1}^{K} \pi_k = 1$

$$L = -\sum_{n=1}^{N} log f(x_n) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)$$

$\frac{\partial L}{\partial \pi_k} = 0$, then we have $\pi_k = \frac{N_k}{N}$

where $N_k = \sum_{n=1}^{N} \gamma_{(n,k)}$

Algorithm for Model Fitting to Estimate Parameters

- **Input**: $\{x_1, x_2, x_3, \ldots, x_N\}$ is a set of data samples, and $x_n \in \mathcal{R}^M$,
  K is the number of components
- **Initialization**: initial values of μ_k, Σ_k, π_k for k=1 to K
- **E-step**: compute the soft assignment $\gamma_{(n,k)}$, the probability of sample x_n belonging to $cluster_k$

$$\gamma_{(n,k)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

soft assignment matrix $[\gamma_{(n,k)}]$

- **M-step**: update the parameters μ_k, Σ_k, π_k for k=1 to K

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{(n,k)} x_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{(n,k)} (x_n - \mu_k)(x_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}, \text{ where } N_k = \sum_{n=1}^{N} \gamma_{(n,k)}$$

- Iterate the two steps until convergence: loss almost does not change

Clustering is Performed After Model Fitting

- **Input**: $\{x_1, x_2, x_3, \ldots, x_N\}$ is a set of data samples, and $x_n \in \mathcal{R}^M$,

  K is the number of components

- Run the Parameter Estimation Algorithm and we obtain the final soft assignment $\gamma_{(n,k)}$, the probability of sample x_n belonging to $cluster_k$

$$\gamma_{(n,k)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)} \text{ where } \sum_{k=1}^{K} \gamma_{(n,k)} = 1$$

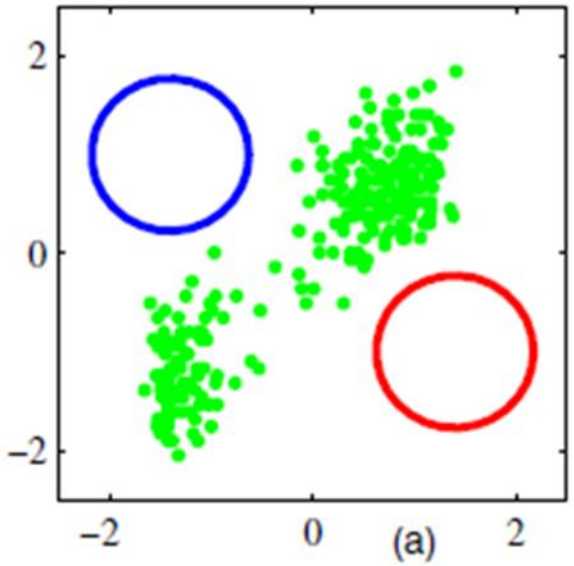
soft assignment matrix $[\gamma_{(n,k)}]$

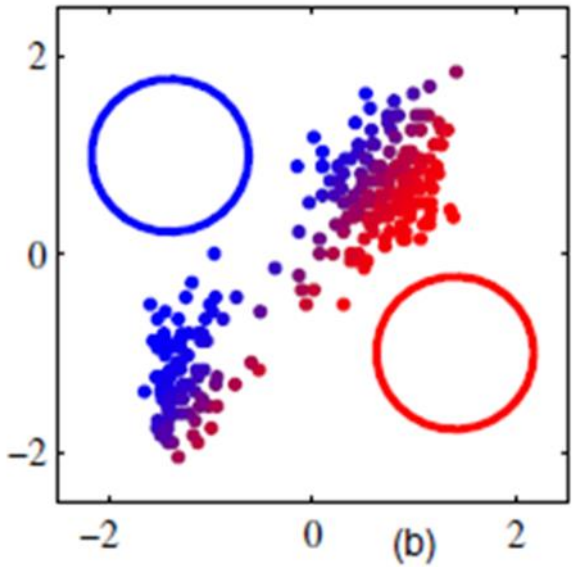
- **Clustering:**

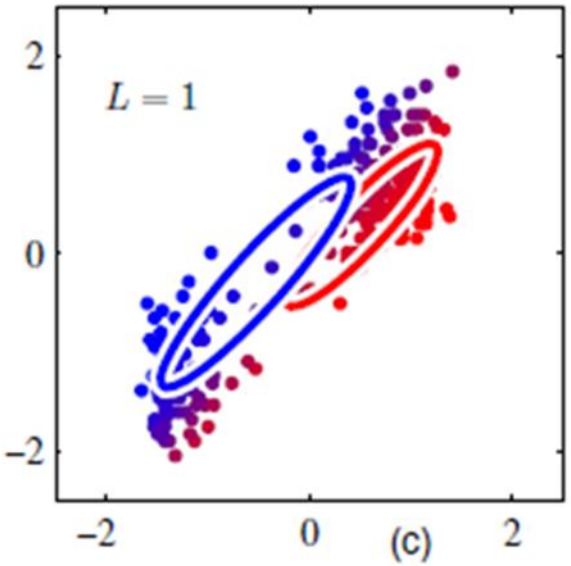
  cluster label of x_n is $argmax[\gamma_{(n,1)}, \gamma_{(n,2)}, \gamma_{(n,3)}, \ldots, \gamma_{(n,K)}]$

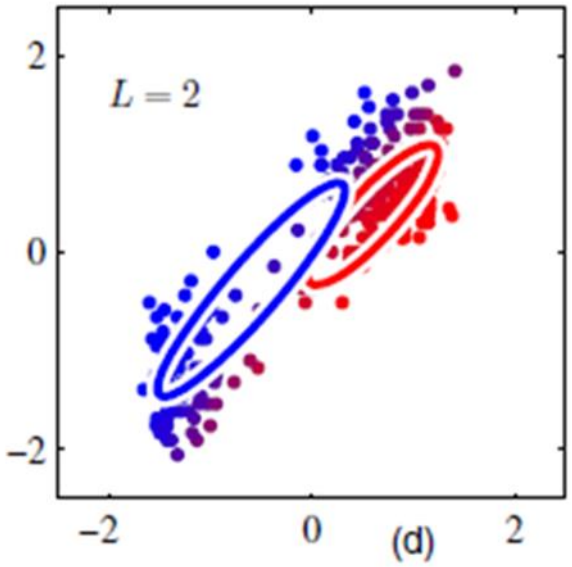
  example: x_n is assigned to cluster-1 if the row-n is [0.9, 0.1, 0.01, …]

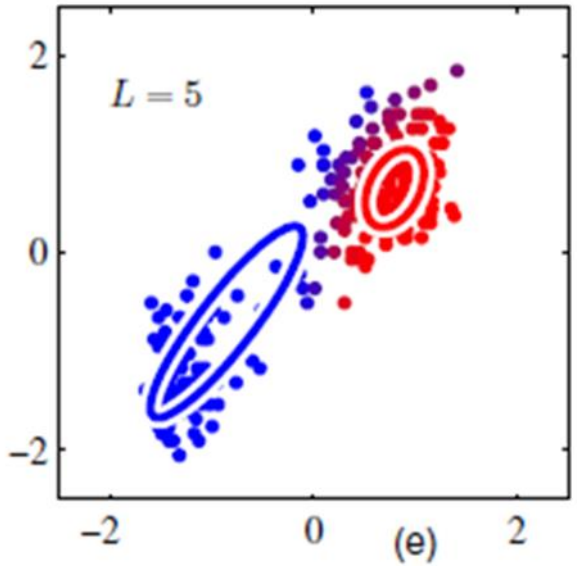
Initialization

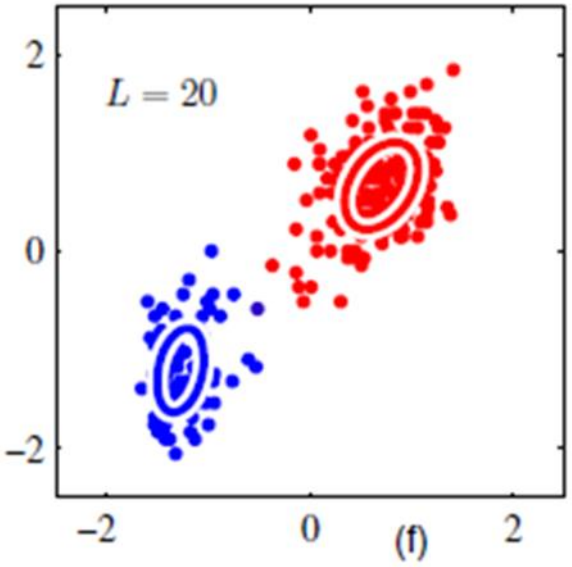












bishop- pattern recognition and machine learning

EM Algorithm

- The algorithm is derived by setting $\frac{\partial L}{\partial \mu_k} = 0$, $\frac{\partial L}{\partial \Sigma_k} = 0$ and $\frac{\partial L}{\partial \pi_k} = 0$

- The algorithm can also be derived from a method: **E**xpectation-**M**aximization
  - E-step: calculate $f^t(\mathbf{J}) \triangleq P(\mathbf{J}|\mathbf{U}, \mathbf{\Theta}^t)$
  - M-step: $\mathbf{\Theta}^{t+1} = \text{argmax}_{\mathbf{\Theta}} [Q^t(\mathbf{\Theta}) + \log P(\mathbf{\Theta})]$

It is very useful if parameter estimation is a 'chicken-egg' problem. A model has two sets of parameters, set-a, set-b.
(1) to get set-a, we need to know set-b
(2) to get set-b, we need to know set-a

The Expectation Maximization Algorithm

Frank Dellaert

College of Computing, Georgia Institute of Technology
Technical Report number GIT-GVU-02-20
February 2002

Abstract

This note represents my attempt at explaining the EM algorithm (Hartley, 1958; Dempster et al., 1977; McLachlan and Krishnan, 1997). This is just a slight variation on Tom Minka's tutorial (Minka, 1998), perhaps a little easier (or perhaps not). It includes a graphical example to provide some intuition.

How can we determine the optimal value of *K* ?

- Knowledge of the application
- Bayesian information criterion (BIC)
- Akaike information criterion (AIC)

- The basic idea is to reduce the number of components while keeping a high log likelihood. For example, you may define your own criterion:

$$\text{loss}(K) = -\,\text{Log-likelihood}/N + g(K)$$

GMM => modified k-means for clustering

- **E-step (GMM)**: compute the soft assignment $\gamma_{(n,k)}$, the probability of sample x_n belonging to $cluster_k$

$$\gamma_{(n,k)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)} \text{ where } \sum_{k=1}^{K} \gamma_{(n,k)} = 1$$

- **E-step (modified k-means)**: compute the soft assignment $\gamma_{(n,k)}$, and convert it to hard assignment.

  e.g. K=2 soft $\gamma_{(n,1)} = 0.1, \gamma_{(n,2)} = 0.9$ => hard $\gamma_{(n,1)} = 0, \gamma_{(n,2)} = 1$

- **M-step**: update the parameters μ_k, Σ_k, π_k for k=1 to K

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{(n,k)} x_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{(n,k)} (x_n - \mu_k)(x_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}, \text{ where } N_k = \sum_{n=1}^{N} \gamma_{(n,k)}$$

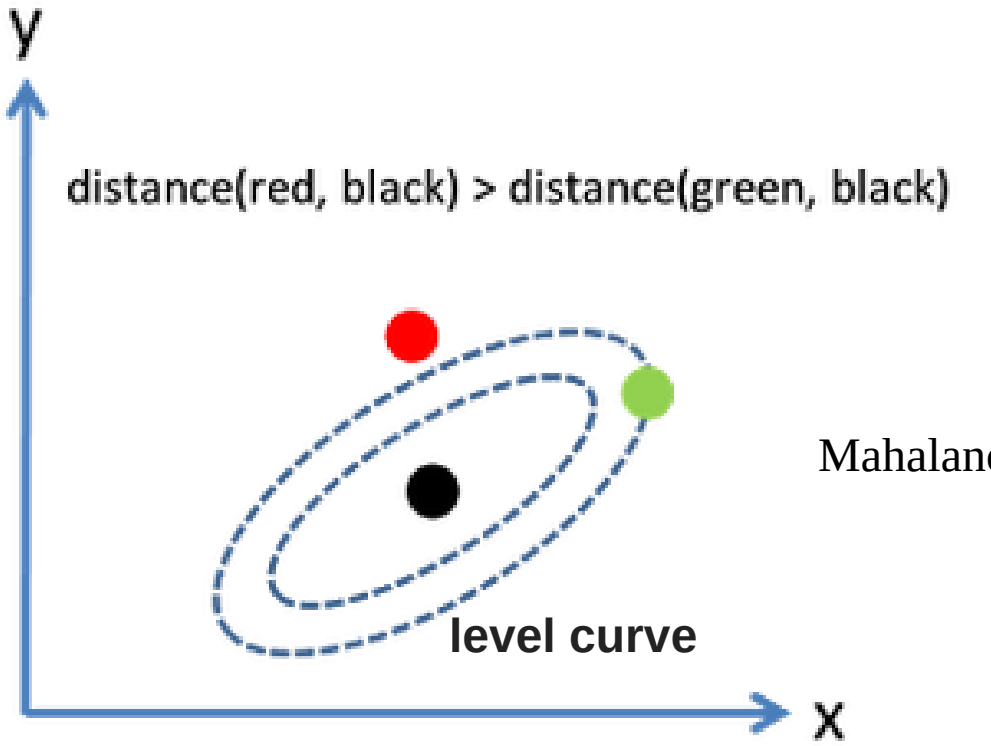
- Iterate the two steps until convergence: loss almost does not change

The distance function of the modified k-means

- **The distance function is**

$$d_k(x_n, \mu_k) = \sqrt{(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)}$$

a.k.a. Mahalanobis distance



Mahalanobis distance in 2D space

https://lylelin317.wordpress.com/2014/03/11/distance-and-similarity-measuring-methods-in-machine-learning/

The distance function of the modified k-means

- **E-step (modified k-means)**: compute the soft assignment $\gamma_{(n,k)}$, and convert it to hard assignment.

$$\gamma_{(n,k)} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)} \text{ where } \sum_{k=1}^{K} \gamma_{(n,k)} = 1$$

set K=2:

$$\gamma_{(n,1)} = \frac{\pi_1 \mathcal{N}(x_n|\mu_1,\Sigma_1)}{\sum_{j=1}^{2} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)},\ \gamma_{(n,2)} = \frac{\pi_2 \mathcal{N}(x_n|\mu_2,\Sigma_2)}{\sum_{j=1}^{2} \pi_j \mathcal{N}(x_n|\mu_j,\Sigma_j)} \text{ then } \gamma_{(n,1)} + \gamma_{(n,2)} = 1$$

if $\gamma_{(n,1)} = 0$, then x_n belongs to cluster-2

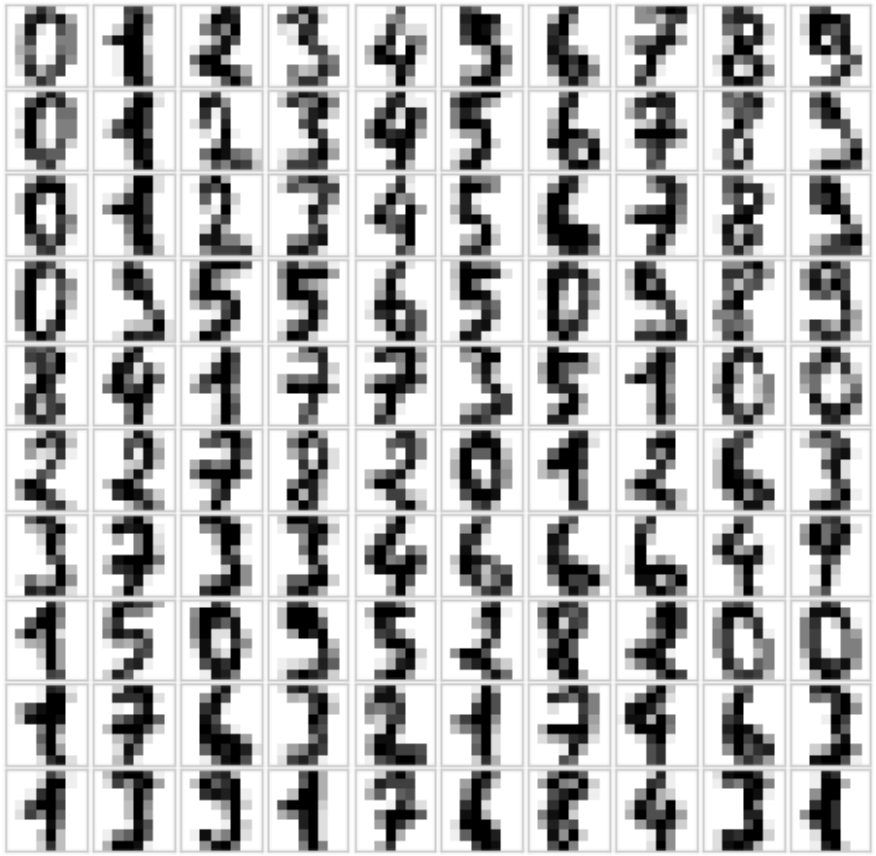
else: ratio $= \frac{\gamma_{(n,2)}}{\gamma_{(n,1)}} = \frac{\pi_2 \mathcal{N}(x_n|\mu_2,\Sigma_2)}{\pi_1 \mathcal{N}(x_n|\mu_1,\Sigma_1)}$, if ratio > 1, then $x_n \in cluster_2$

$$\log \frac{\gamma_{(n,2)}}{\gamma_{(n,1)}} \approx (x_n - \mu_1)^T \Sigma_1^{-1} (x_n - \mu_1) - (x_n - \mu_2)^T \Sigma_2^{-1} (x_n - \mu_2)$$

$$= \left(d_1(x_n, \mu_1)\right)^2 - \left(d_2(x_n, \mu_2)\right)^2$$

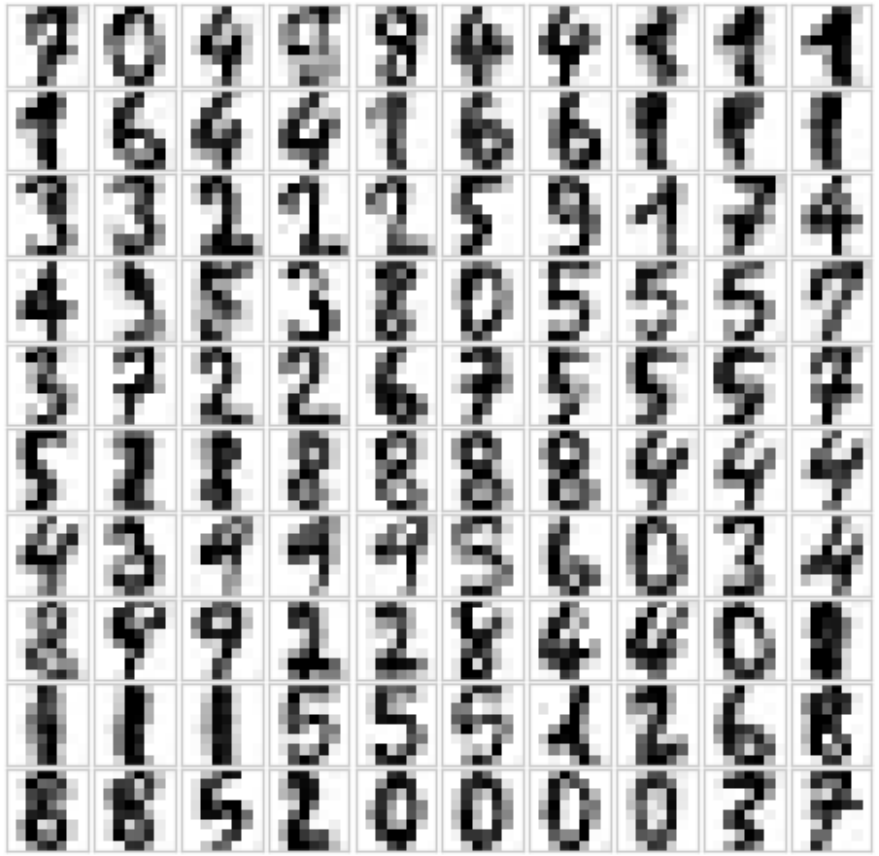
GMM is a Generative Model because it is a PDF

Training Data



GMM

Generated New Data



see demos:
2D_GMM.ipynb
GMM_DE_Generative_Model.ipynb