



# HUST

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.



ĐẠI HỌC  
BÁCH KHOA HÀ NỘI  
HANOI UNIVERSITY  
OF SCIENCE AND TECHNOLOGY

# Introduction to Machine Learning and Data Mining

IT3190

Lecture: Probabilistic models – EM algorithm

ONE LOVE. ONE FUTURE.

# Contents

- Lecture 1: Introduction to Machine Learning & Data Mining
- Lecture 2: Data crawling and pre-processing
- Lecture 3: Linear regression
- Lecture 4+5: Clustering
- Lecture 6: Decision tree and Random forest
- Lecture 7: Neural networks
- Lecture 8: Support vector machines
- Lecture 9: Performance evaluation
- **Lecture 10: Probabilistic models**
- Lecture 11: Basics of data mining
- Lecture 12: Association rule mining
- Lecture 13: Regularization and advanced topics

# Difficult situations

- No closed-form solution for the learning/inference problem?  
(không tìm được ngay công thức nghiệm)
  - The examples before are easy cases, as we can find solutions in a closed form by using gradient.
  - Many models (e.g., GMM) do not admit a closed-form solution
- No explicit expression of the density/mass function?  
(không có công thức tường minh để tính toán)
- Intractable inference (bài toán không khả thi)
  - Inference in many probabilistic models is NP-hard  
[Sontag & Roy, 2011; Tosh & Dasgupta, 2019]

# Expectation maximization

- The EM algorithm

# GMM revisit

- Consider learning GMM, with  $K$  Gaussian distributions, from the training data  $\mathbf{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ .
- The density function is  $p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi}) = \sum_{k=1}^K \phi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ 
  - $\boldsymbol{\phi} = (\phi_1, \dots, \phi_K)$  represents the weights of the Gaussians,  $P(z = k | \boldsymbol{\phi}) = \phi_k$ .
  - Each multivariate Gaussian has density
$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma}_k)}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$
- MLE tries to maximize the following log-likelihood function

$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi}) = \sum_{i=1}^M \log \sum_{k=1}^K \phi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- **We cannot find a closed-form solution!**
- **Naïve gradient decent:** repeat until convergence
  - Optimize  $L(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$  w.r.t  $\boldsymbol{\phi}$ , when fixing  $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ .
  - Optimize  $L(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$  w.r.t  $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , when fixing  $\boldsymbol{\phi}$ .

Still hard

# GMM revisit: K-means

## □ GMM: we need to know

- Among  $K$  gaussian components, which generates an instance  $\mathbf{x}$ ?  
the index  $z$  of the gaussian component
- The parameters of individual gaussian components:  $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \phi_k)$

## □ K-means:

- Among  $K$  clusters, to which an instance  $\mathbf{x}$  belongs?  
the cluster index  $z$
- The parameters of individual clusters: the mean

## □ Idea for GMM?

- $P(z|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$ ?  
(note  $\sum_{k=1}^K P(z = k|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi}) = 1$ )  
(soft assignment)
- Update the parameters of individual gaussians:  $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \phi_k)$

## □ K-means training:

- Step 1: assign each instance  $\mathbf{x}$  to the nearest cluster  
(the cluster index  $z$  for each  $\mathbf{x}$ )  
(hard assignment)
- Step 2: recompute the means of the clusters

# GMM: lower bound

## □ Idea for GMM?

- Step 1: compute  $P(z|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$ ? (note  $\sum_{k=1}^K P(z = k|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi}) = 1$ )
- Step 2: Update the parameters of the gaussian components:  $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$

- Consider the log-likelihood function

$$L(\boldsymbol{\theta}) = \log P(\mathbf{D}|\boldsymbol{\theta}) = \sum_{i=1}^M \log \sum_{k=1}^K \phi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Too complex if directly using gradient descent
- Note that  $\log P(\mathbf{x}|\boldsymbol{\theta}) = \log P(\mathbf{x}, z|\boldsymbol{\theta}) - \log P(z|\mathbf{x}, \boldsymbol{\theta})$ . Therefore

$$\log P(\mathbf{x}|\boldsymbol{\theta}) = \mathbb{E}_{z|\mathbf{x}, \boldsymbol{\theta}} \log P(\mathbf{x}, z|\boldsymbol{\theta}) - \mathbb{E}_{z|\mathbf{x}, \boldsymbol{\theta}} \log P(z|\mathbf{x}, \boldsymbol{\theta}) \geq \mathbb{E}_{z|\mathbf{x}, \boldsymbol{\theta}} \log P(\mathbf{x}, z|\boldsymbol{\theta})$$

- Maximizing  $L(\boldsymbol{\theta})$  can be done by *maximizing the lower bound*

$$LB(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \mathbf{D}} \mathbb{E}_{z|\mathbf{x}, \boldsymbol{\theta}} \log P(\mathbf{x}, z|\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \mathbf{D}} \sum_z P(z|\mathbf{x}, \boldsymbol{\theta}) \log P(\mathbf{x}, z|\boldsymbol{\theta})$$



# GMM: maximize the lower bound

- Step 1: compute  $P(z|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$ ? (note  $\sum_{k=1}^K P(z = k|\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi}) = 1$ )
- Step 2: Update the parameters of the gaussian components:  $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$

- Bayes' rule:  $P(z|\mathbf{x}, \boldsymbol{\theta}) = P(\mathbf{x}|z, \boldsymbol{\theta})P(z|\boldsymbol{\phi})/P(\mathbf{x}) = \phi_z \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)/C$ , where  $C$  is the normalizing constant.

- Meaning that one can compute  $P(z|\mathbf{x}, \boldsymbol{\theta})$  if  $\boldsymbol{\theta}$  is known
- Denoting  $T_{ki} = P(z = k|\mathbf{x}_i, \boldsymbol{\theta})$  for any index  $k = \overline{1, K}, i = \overline{1, M}$

- How about  $\boldsymbol{\phi}$ ?

- $\phi_z = P(z|\boldsymbol{\phi}) = P(z|\boldsymbol{\theta}) = \int P(z, \mathbf{x}|\boldsymbol{\theta})d\mathbf{x} = \int P(z|\mathbf{x}, \boldsymbol{\theta})P(\mathbf{x}|\boldsymbol{\theta})d\mathbf{x} = \mathbb{E}_{\mathbf{x}}(P(z|\mathbf{x}, \boldsymbol{\theta})) \approx \frac{1}{M} \sum_{\mathbf{x} \in D} P(z|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^M T_{zi}$

- Then the lower bound can be maximized w.r.t individual  $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ :

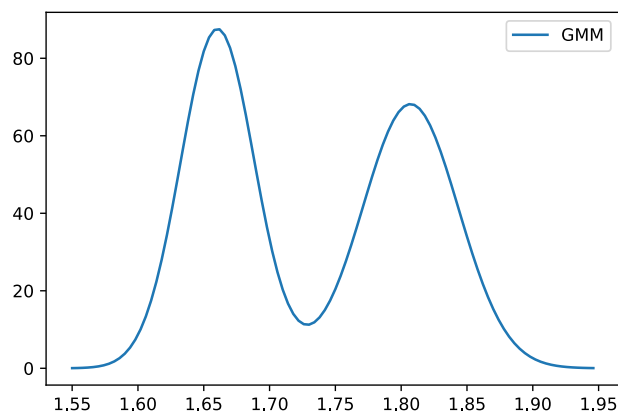
$$\begin{aligned} LB(\boldsymbol{\theta}) &= \sum_{\mathbf{x} \in D} \sum_z P(z|\mathbf{x}, \boldsymbol{\theta}) \log[P(\mathbf{x}|z, \boldsymbol{\theta})P(z|\boldsymbol{\theta})] \\ &= \sum_{i=1}^M \sum_{k=1}^K T_{ki} \left[ -\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) - \log \sqrt{\det(2\pi \boldsymbol{\Sigma}_k)} \right] + constant \end{aligned}$$

# GMM: EM algorithm

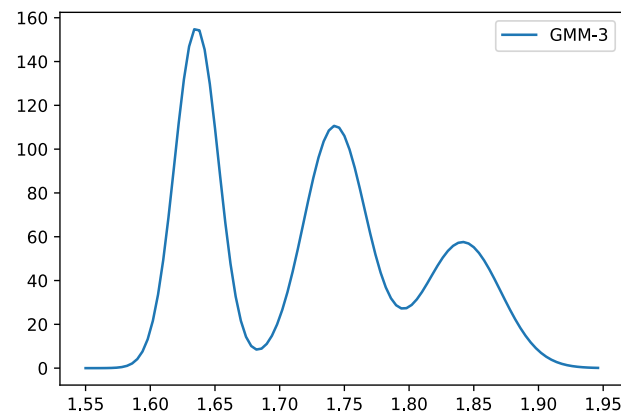
- **Input:** training data  $\mathbf{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ ,  $K > 0$
- **Output:** model parameter  $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\phi})$
- Initialize  $(\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}, \boldsymbol{\phi}^{(0)})$  randomly
  - $\boldsymbol{\phi}^{(0)}$  must be non-negative and sum to 1.
- At iteration  $t$ :
  - **E step:** compute  $T_{ki} = P(z = k | \mathbf{x}_i, \boldsymbol{\theta}^{(t)}) = \phi_k^{(t)} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) / C$  for any index  $k = \overline{1, K}, i = \overline{1, M}$
  - **M step:** update for any  $k$ ,
$$\phi_k^{(t+1)} = \frac{1}{M} \sum_{i=1}^M T_{ki}; \boldsymbol{\mu}_k^{(t+1)} = \frac{1}{M \phi_k} \sum_{i=1}^M T_{ki} \mathbf{x}_i;$$
$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{M \phi_k} \sum_{i=1}^M T_{ki} (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)}) (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})^T$$
- If not convergence, go to iteration  $t + 1$ .

# GMM: example 1

- We wish to model the height of a person
  - We had collected a dataset from 10 people in Hanoi + 10 people in Sydney  
 $\mathbf{D} = \{1.6, 1.7, 1.65, 1.63, 1.75, 1.71, 1.68, 1.72, 1.77, 1.62, 1.75, 1.80, 1.85, 1.65, 1.91, 1.78, 1.88, 1.79, 1.82, 1.81\}$



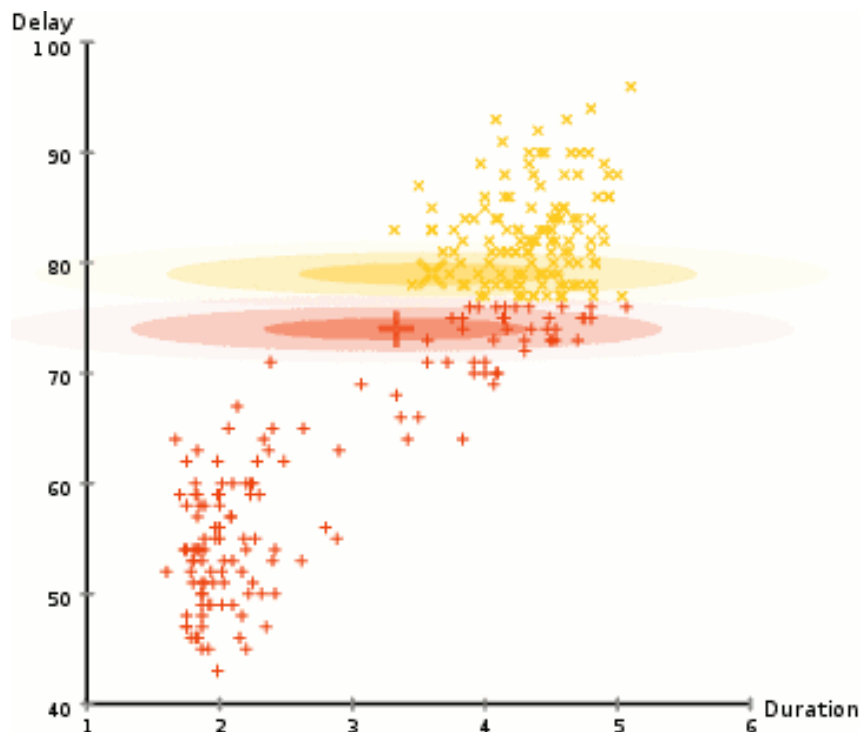
GMM with  
2 components



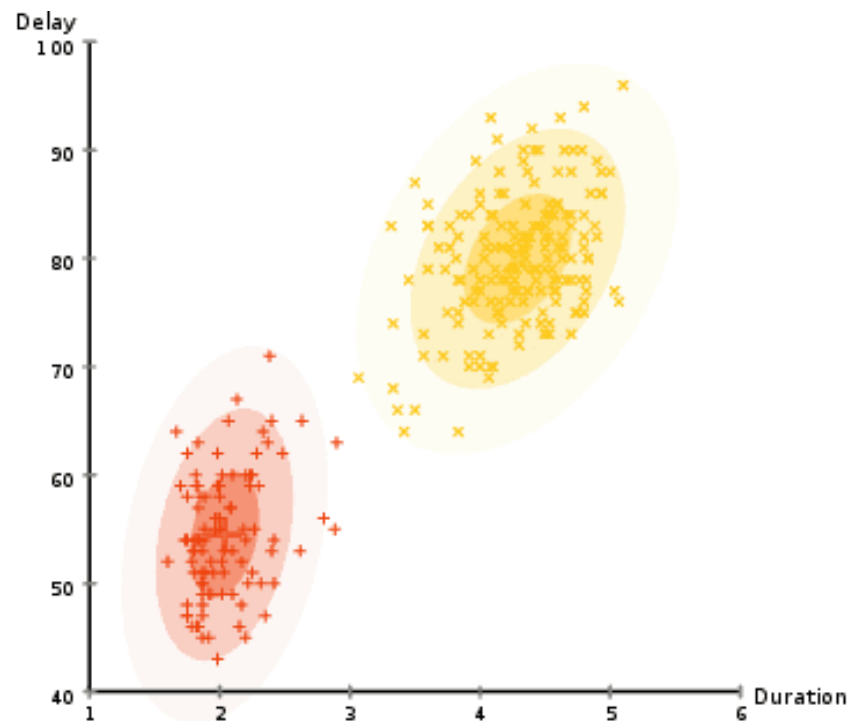
GMM with  
3 components

# GMM: example 2

- A GMM is fitted in a 2-dimensional dataset to do clustering.



From initialization



To convergence

[https://en.wikipedia.org/wiki/Expectation-maximization\\_algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm)

# GMM: comparison with K-means

## □ K-means:

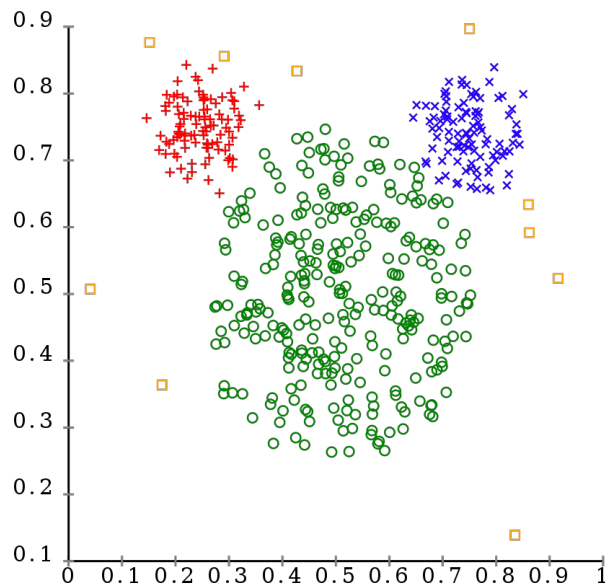
- Step 1: hard assignment
- Step 2: the means  
→ similar shape for the clusters?

## □ GMM clustering

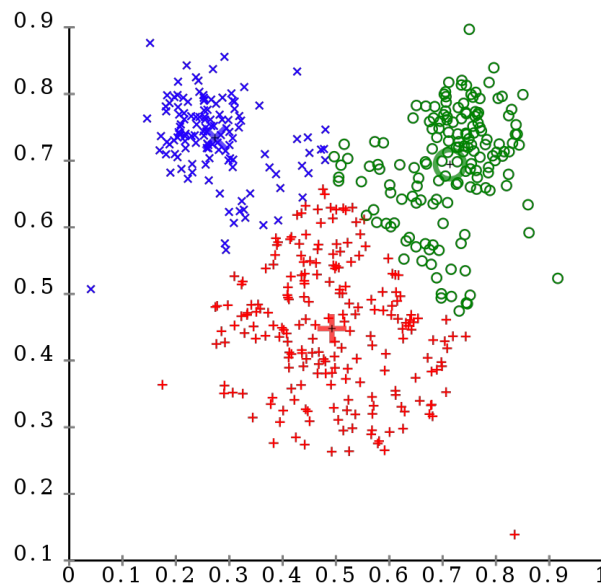
- Soft assignment of data to the clusters
- Parameters ( $\mu_k, \Sigma_k, \phi_k$ )  
→ different shapes for the clusters

Different cluster analysis results on "mouse" data set:

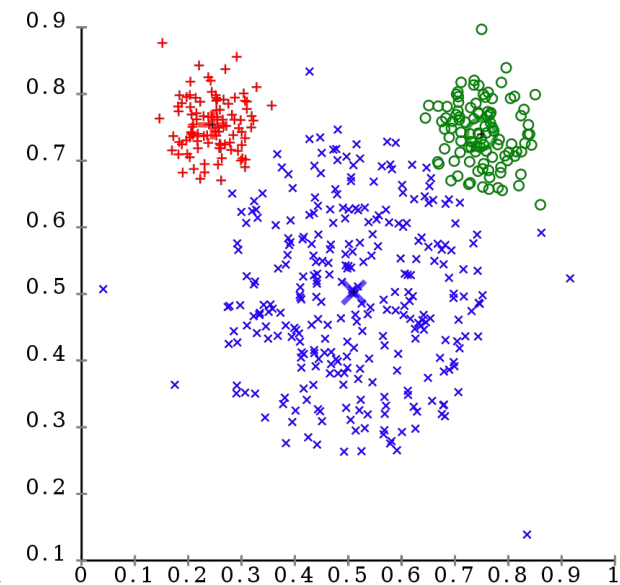
Original Data



k-Means Clustering



EM Clustering



[https://en.wikipedia.org/wiki/Expectation-maximization\\_algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm)

# General models

- We can make the EM algorithm in more general cases.
- Consider a model  $B(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})$  with observed variable  $\mathbf{x}$ , hidden variable  $\mathbf{z}$ , and parameterized by  $\boldsymbol{\theta}$   
(mô hình có một biến  $\mathbf{x}$  quan sát được, biến ẩn  $\mathbf{z}$ , và tham số  $\boldsymbol{\theta}$ )
  - $\mathbf{x}$  depends on  $\mathbf{z}$  and  $\boldsymbol{\theta}$ , while  $\mathbf{z}$  may depend on  $\boldsymbol{\theta}$
  - Mixture models: each observed data point has a corresponding latent variable, specifying the mixture component which generated the data point
- The learning task is to find a specific model, from the model family parameterized by  $\boldsymbol{\theta}$ , that maximizes the log-likelihood of training data  $\mathbf{D}$ :
$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \log P(\mathbf{D}|\boldsymbol{\theta})$$
- We assume  $\mathbf{D}$  consists of i.i.d samples of  $\mathbf{x}$ , the the log-likelihood function can be expressed analytically,  $\mathbb{E}_{\mathbf{z}|\mathbf{D},\boldsymbol{\theta}} \log P(\mathbf{D}|\boldsymbol{\theta})$  can be computed easily (hàm log-likelihood có thể viết một cách tường minh)
  - Since there is a latent variable, MLE may not have a close form solution

# The Expectation Maximization algorithm

- The Expectation maximization (EM) algorithm was introduced in 1977 by Arthur Dempster, Nan Laird, and Donald Rubin.
- The EM algorithm maximizes the lower bound of the log-likelihood

$$L(\boldsymbol{\theta}; \mathbf{D}) = \log P(\mathbf{D}|\boldsymbol{\theta}) \geq LB(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \mathbf{D}} \mathbb{E}_{z|\mathbf{x}, \boldsymbol{\theta}} \log P(\mathbf{x}, z|\boldsymbol{\theta})$$

- *Initialization:  $\boldsymbol{\theta}^{(0)}, t = 0$*
- *At iteration  $t$ :*
  - **E step:** *compute the expectation  $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = LB(\boldsymbol{\theta}^{(t-1)})$*   
(tính hàm kỳ vọng Q khi cố định giá trị  $\boldsymbol{\theta}^{(t)}$  đã biết ở bước trước)
  - **M step:** *find  $\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$*   
(tìm điểm  $\boldsymbol{\theta}^{(t+1)}$  mà làm cho hàm Q đạt cực đại)
- *If not convergence, go to iteration  $t + 1$ .*

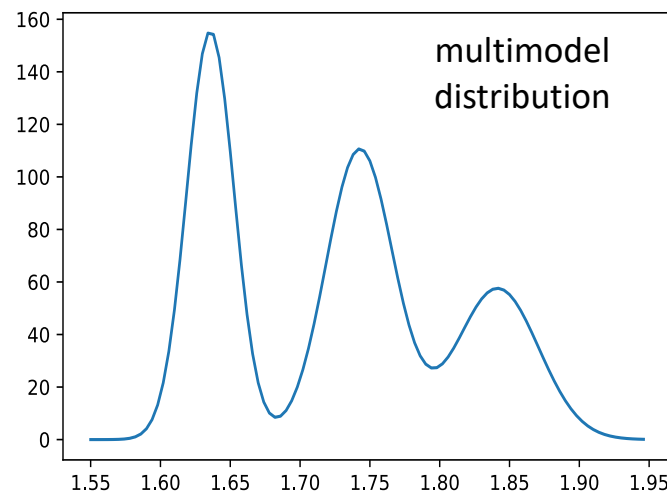
# EM: convergence condition

- Different conditions can be used to check convergence
  - $LB(\theta)$  does not change much between two consecutive iterations
  - $\theta$  does not change much between two consecutive iterations
- In practice, we sometimes need to limit the maximum number of iterations



# EM: some properties

- The EM algorithm is guaranteed to return a stationary point of the lower bound  $LB(\theta)$   
(thuật toán EM đảm bảo sẽ hội tụ về một điểm dừng của hàm cận dưới)
  - It may be the local maximum
- Due to maximizing the lower bound, EM does not necessarily return the maximizer of the log-likelihood function  
(EM chưa chắc trả về điểm cực đại của hàm log-likelihood)
  - No guarantee exists
  - It can be seen in cases of multimodel, where the log-likelihood function is non-concave
- The Baum-Welch algorithm is the a special case of EM for hidden Markov models

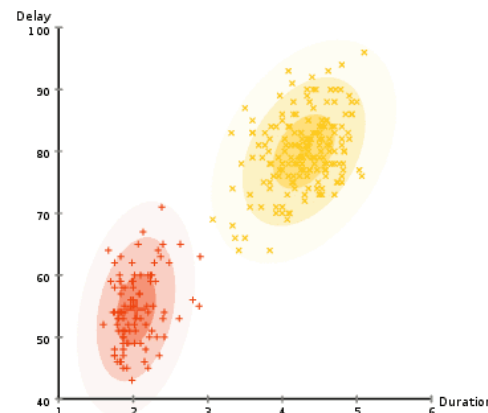


# EM, mixture model, and clustering

- *Mixture model*: we assume the data population is composed of  $K$  different components (distributions), and each data point is generated from one of those components
  - E.g., Gaussian mixture model, categorical mixture model, Bernoulli mixture model,...
  - The mixture density function can be written as

$$f(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{k=1}^K \phi_k f_k(\mathbf{x} | \boldsymbol{\theta}_k)$$

where  $f_k(\mathbf{x} | \boldsymbol{\theta}_k)$  is the density of the  $k$ -th component



- We can interpret that a mixture distribution partitions the data space into different regions, each associates with a component  
(Một phân bố hỗn hợp tạo ra một cách chia không gian dữ liệu ra thành các vùng khác nhau, mà mỗi vùng tương ứng với 1 thành phần trong hỗn hợp đó)
- Hence, mixture models provide solutions for clustering
- The EM algorithm provides a natural way to learn mixture models

- When the lower bound  $LB(\theta)$  does not admit easy computation of the expectation or maximization steps
  - Admixture models, Bayesian mixture models
  - Hierarchical probabilistic models
  - Nonparametric models
- EM finds a point estimate, hence easily gets stuck at a local maximum
- In practice, EM is sensitive with initialization
  - Is it good to use the idea of K-means++ for initialization?
- Sometimes EM converges slowly in practice

# Further?

- Variational inference
  - Inference for more general models
- Deep generative models
  - Neural networks + probability theory
- Bayesian neural networks
  - Neural networks + Bayesian inference
- Amortized inference
  - Neural networks for doing Bayesian inference
  - Learning to do inference

# Reference

- Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "Variational inference: A review for statisticians." *Journal of the American Statistical Association* 112, no. 518 (2017): 859-877.
- Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight Uncertainty in Neural Network." In *International Conference on Machine Learning (ICML)*, pp. 1613-1622. 2015.
- Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B*. 39 (1): 1-38.
- Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In *ICML*, pp. 1050-1059. 2016.
- Ghahramani, Zoubin. "Probabilistic machine learning and artificial intelligence." *Nature* 521, no. 7553 (2015): 452-459.
- Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." In *International Conference on Learning Representations (ICLR)*, 2014.
- Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." *Science* 349, no. 6245 (2015): 255-260.
- Tosh, Christopher, and Sanjoy Dasgupta. "The Relative Complexity of Maximum Likelihood Estimation, MAP Estimation, and Sampling." In *COLT, PMLR* 99:2993-3035, 2019.
- Sontag, David, and Daniel Roy, "Complexity of inference in latent dirichlet allocation" in: *Advances in Neural Information Processing System*, 2011.

A decorative graphic on the left side of the slide. It features a dark blue background with a large, stylized circular pattern composed of many small red dots. The dots are arranged in concentric, slightly irregular rings, creating a sense of depth and movement. The word "HUST" is centered within this pattern.

# HUST

# THANK YOU !