Generative Models for Clustering, GMM, and Intro to EM

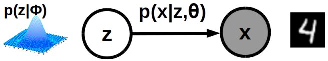
Piyush Rai

Machine Learning (CS771A)

Sept 26, 2016

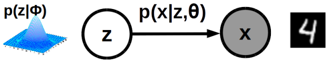
Generative Models

- A probabilistic way to think about the data generation process



Generative Models

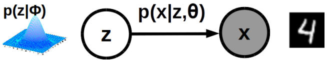
- A probabilistic way to think about the data generation process



- Idea: First generate a random latent variable $\boldsymbol{z}$ from a prior distr. $p(\boldsymbol{z}|\phi)$ and then generate $\boldsymbol{x}$ conditioned on $\boldsymbol{z}$ from the data distr. $p(\boldsymbol{x}|\boldsymbol{z},\theta)$

Generative Models

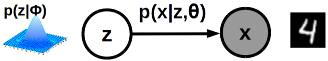
- A probabilistic way to think about the data generation process



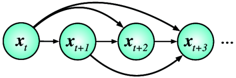
- Idea: First generate a random latent variable $\boldsymbol{z}$ from a prior distr. $p(\boldsymbol{z}|\phi)$ and then generate $\boldsymbol{x}$ conditioned on $\boldsymbol{z}$ from the data distr. $p(\boldsymbol{x}|\boldsymbol{z},\theta)$
- Some exceptions to this general view/definition of a generative model:

Generative Models

- A probabilistic way to think about the data generation process

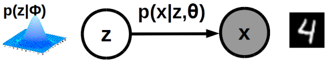


- Idea: First generate a random latent variable $\boldsymbol{z}$ from a prior distr. $p(\boldsymbol{z}|\phi)$ and then generate $\boldsymbol{x}$ conditioned on $\boldsymbol{z}$ from the data distr. $p(\boldsymbol{x}|\boldsymbol{z},\theta)$
- Some exceptions to this general view/definition of a generative model:
  - Not all generative models have latent variables (e.g., for each observation, the other observations may play the role of latent variables)

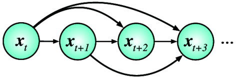


Generative Models

- A probabilistic way to think about the data generation process



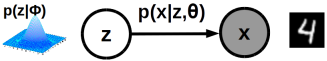
- Idea: First generate a random latent variable $\boldsymbol{z}$ from a prior distr. $p(\boldsymbol{z}|\phi)$ and then generate $\boldsymbol{x}$ conditioned on $\boldsymbol{z}$ from the data distr. $p(\boldsymbol{x}|\boldsymbol{z},\theta)$
- Some exceptions to this general view/definition of a generative model:
  - Not all generative models have latent variables (e.g., for each observation, the other observations may play the role of latent variables)



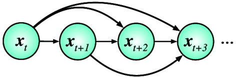
  - The $\boldsymbol{z}$ to $\boldsymbol{x}$ map may be a deterministic fn. (e.g., a neural or deep neural net)

Generative Models

- A probabilistic way to think about the data generation process



- Idea: First generate a random latent variable $\boldsymbol{z}$ from a prior distr. $p(\boldsymbol{z}|\phi)$ and then generate $\boldsymbol{x}$ conditioned on $\boldsymbol{z}$ from the data distr. $p(\boldsymbol{x}|\boldsymbol{z},\theta)$
- Some exceptions to this general view/definition of a generative model:
  - Not all generative models have latent variables (e.g., for each observation, the other observations may play the role of latent variables)

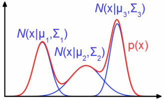


  - The $\boldsymbol{z}$ to $\boldsymbol{x}$ map may be a deterministic fn. (e.g., a neural or deep neural net)
- We will focus on probabilistic generative models with latent variables

Generative Models for Clustering

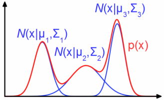
Gaussian Mixture Model (GMM)

- A generative model for data clustering
- Data assumed generated from a mixture of K Gaussians



Gaussian Mixture Model (GMM)

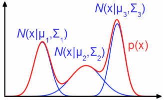
- A generative model for data clustering
- Data assumed generated from a mixture of K Gaussians



- Let $0 \leq \pi_k \leq 1$ denote the "mixing weight" of the k-th Gaussian. It means:
  - π_k is the fraction of points generated from the k-th Gaussian

Gaussian Mixture Model (GMM)

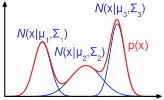
- A generative model for data clustering
- Data assumed generated from a mixture of K Gaussians



- Let $0 \leq \pi_k \leq 1$ denote the "mixing weight" of the k-th Gaussian. It means:
  - π_k is the fraction of points generated from the k-th Gaussian
  - $\pi_k = p(\boldsymbol{z}_n = k)$ is the prior prob. of $\boldsymbol{x}_n$ belonging to the k-th Gaussian

Gaussian Mixture Model (GMM)

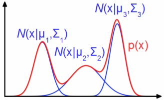
- A generative model for data clustering
- Data assumed generated from a mixture of K Gaussians



- Let $0 \leq \pi_k \leq 1$ denote the "mixing weight" of the k-th Gaussian. It means:
  - π_k is the fraction of points generated from the k-th Gaussian
  - $\pi_k = p(\boldsymbol{z}_n = k)$ is the prior prob. of $\boldsymbol{x}_n$ belonging to the k-th Gaussian
- Let $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$ denote the vector of mixing wts of K Gaussians. This is a probability vector and sums to 1, i.e., $\sum_{k=1}^{K} \pi_k = 1$

Gaussian Mixture Model (GMM)

- A generative model for data clustering
- Data assumed generated from a mixture of K Gaussians

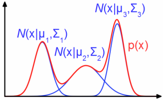


- Let $0 \leq \pi_k \leq 1$ denote the "mixing weight" of the k-th Gaussian. It means:
  - π_k is the fraction of points generated from the k-th Gaussian
  - $\pi_k = p(\boldsymbol{z}_n = k)$ is the prior prob. of $\boldsymbol{x}_n$ belonging to the k-th Gaussian
- Let $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$ denote the vector of mixing wts of K Gaussians. This is a probability vector and sums to 1, i.e., $\sum_{k=1}^{K} \pi_k = 1$
- Notation $\boldsymbol{z}_n = k$ is equivalent to a size K one-hot vector $\boldsymbol{z}_n$

$$\boldsymbol{z}_n \quad = \quad \underbrace{[0 \;\; 0 \; \ldots \; 1 \;\; 0 \;\; 0]}_{\text{all zeros except the k-th bit, i.e., } z_{nk} = 1}$$

Gaussian Mixture Model (GMM)

- A generative model for data clustering
- Data assumed generated from a mixture of K Gaussians



- Let $0 \leq \pi_k \leq 1$ denote the "mixing weight" of the k-th Gaussian. It means:
  - π_k is the fraction of points generated from the k-th Gaussian
  - $\pi_k = p(\boldsymbol{z}_n = k)$ is the prior prob. of $\boldsymbol{x}_n$ belonging to the k-th Gaussian
- Let $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$ denote the vector of mixing wts of K Gaussians. This is a probability vector and sums to 1, i.e., $\sum_{k=1}^{K} \pi_k = 1$
- Notation $\boldsymbol{z}_n = k$ is equivalent to a size K one-hot vector $\boldsymbol{z}_n$

$$\boldsymbol{z}_n = \underbrace{[0 \;\; 0 \;\; \ldots \;\; 1 \;\; 0 \;\; 0]}_{\text{all zeros except the k-th bit, i.e., } z_{nk} = 1}$$

- Note: The prior $p(\boldsymbol{z}|\pi)$ on each $\boldsymbol{z}_n$ is a multinomial, i.e., $p(\boldsymbol{z}_n|\pi) = \prod_{k=1}^{K} \pi_k^{z_{nk}}$

GMM: The Generative Story

- The generative story for each $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$

GMM: The Generative Story

- The generative story for each $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$
  - First choose one of the K mixture components as

$$\boldsymbol{z}_n \sim \text{Multinomial}(\boldsymbol{z}_n|\pi) \qquad \text{(from the prior } p(\boldsymbol{z}) \text{ over } \boldsymbol{z}\text{)}$$

GMM: The Generative Story

- The generative story for each $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$
  - First choose one of the K mixture components as

$$\boldsymbol{z}_n \sim \text{Multinomial}(\boldsymbol{z}_n|\pi) \qquad \text{(from the prior } p(\boldsymbol{z}) \text{ over } \boldsymbol{z})$$

  - Suppose $\boldsymbol{z}_n = k$. Now generate $\boldsymbol{x}_n$ from the k-th Gaussian as

$$\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(from the data distr. } p(\boldsymbol{x}|\boldsymbol{z}))$$

GMM: The Generative Story

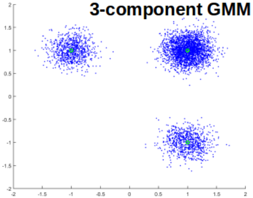
- The generative story for each $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$
  - First choose one of the K mixture components as

$$\boldsymbol{z}_n \sim \text{Multinomial}(\boldsymbol{z}_n|\pi) \qquad \text{(from the prior } p(\boldsymbol{z}) \text{ over } \boldsymbol{z})$$

  - Suppose $\boldsymbol{z}_n = k$. Now generate $\boldsymbol{x}_n$ from the k-th Gaussian as

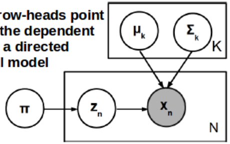
$$\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(from the data distr. } p(\boldsymbol{x}|\boldsymbol{z}))$$

Some simulated data from a 3-component GMM



Directed Graphical Model for a K-component GMM

Note: Arrow-heads point towards the dependent nodes in a directed graphical model



Shaded nodes: Observed

White nodes: Unknowns

Multivariate Gaussian Distribution

- Multivariate Gaussian in D dimensions

$$p(\boldsymbol{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^\top\Sigma^{-1}(\boldsymbol{x}-\mu)\right)$$

Multivariate Gaussian Distribution

- Multivariate Gaussian in D dimensions

$$p(\boldsymbol{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^\top \Sigma^{-1}(\boldsymbol{x}-\mu)\right)$$

- Given N i.i.d. observations $\{\boldsymbol{x}_n\}_{n=1}^N$ from this Gaussian

Multivariate Gaussian Distribution

- Multivariate Gaussian in D dimensions

$$p(\boldsymbol{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^\top \Sigma^{-1}(\boldsymbol{x}-\mu)\right)$$

- Given N i.i.d. observations $\{\boldsymbol{x}_n\}_{n=1}^N$ from this Gaussian
  - MLE for the $D \times 1$ mean $\mu \in \mathbb{R}^D$

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

Multivariate Gaussian Distribution

- Multivariate Gaussian in D dimensions

$$p(\boldsymbol{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^\top \Sigma^{-1}(\boldsymbol{x}-\mu)\right)$$

- Given N i.i.d. observations $\{\boldsymbol{x}_n\}_{n=1}^N$ from this Gaussian
  - MLE for the $D \times 1$ mean $\mu \in \mathbb{R}^D$

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n$$

  - MLE for the $D \times D$ p.s.d. covariance matrix Σ

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^N (\boldsymbol{x}_n - \hat{\mu})(\boldsymbol{x}_n - \hat{\mu})^\top$$

Multivariate Gaussian Distribution

- Multivariate Gaussian in D dimensions

$$p(\boldsymbol{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^\top \Sigma^{-1}(\boldsymbol{x}-\mu)\right)$$

- Given N i.i.d. observations $\{\boldsymbol{x}_n\}_{n=1}^N$ from this Gaussian
  - MLE for the $D \times 1$ mean $\mu \in \mathbb{R}^D$

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

  - MLE for the $D \times D$ p.s.d. covariance matrix Σ

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \hat{\mu})(\boldsymbol{x}_n - \hat{\mu})^\top$$

- Note: The "trace trick" simplifies the derivative calculations

$$\underbrace{\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}}_{\text{a scalar}} = \text{trace}(\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}) = \text{trace}(\Sigma^{-1} \boldsymbol{x}\boldsymbol{x}^\top)$$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- We basically have to estimate K multivariate Gaussians with wts $\pi_1, \ldots, \pi_K$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
- We basically have to estimate K multivariate Gaussians with wts $\pi_1, \ldots, \pi_K$
- The conditional p.d.f. of a data point $\boldsymbol{x}$ if it comes from Gaussian k

$$p(\boldsymbol{x}|\boldsymbol{z} = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- We basically have to estimate K multivariate Gaussians with wts $\pi_1, \ldots, \pi_K$
- The conditional p.d.f. of a data point $\boldsymbol{x}$ if it comes from Gaussian k

$$p(\boldsymbol{x}|\boldsymbol{z} = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Since $\boldsymbol{z}$ is NOT known, we need to look at the marginal p.d.f. of $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}) = \sum_{k=1}^K p(\boldsymbol{x}, \boldsymbol{z} = k) = \sum_{k=1}^K p(\boldsymbol{z} = k) p(\boldsymbol{x}|\boldsymbol{z} = k) = \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- We basically have to estimate K multivariate Gaussians with wts $\pi_1, \ldots, \pi_K$
- The conditional p.d.f. of a data point $\boldsymbol{x}$ if it comes from Gaussian k

$$p(\boldsymbol{x}|\boldsymbol{z} = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Since $\boldsymbol{z}$ is NOT known, we need to look at the marginal p.d.f. of $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}) = \sum_{k=1}^K p(\boldsymbol{x}, \boldsymbol{z} = k) = \sum_{k=1}^K p(\boldsymbol{z} = k) p(\boldsymbol{x}|\boldsymbol{z} = k) = \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Note: Here $p(\boldsymbol{x})$ means $p(\boldsymbol{x}|\Theta)$ where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- We basically have to estimate K multivariate Gaussians with wts $\pi_1, \ldots, \pi_K$
- The conditional p.d.f. of a data point $\boldsymbol{x}$ if it comes from Gaussian k

$$p(\boldsymbol{x}|\boldsymbol{z} = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Since $\boldsymbol{z}$ is NOT known, we need to look at the marginal p.d.f. of $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}) = \sum_{k=1}^K p(\boldsymbol{x}, \boldsymbol{z} = k) = \sum_{k=1}^K p(\boldsymbol{z} = k) p(\boldsymbol{x}|\boldsymbol{z} = k) = \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Note: Here $p(\boldsymbol{x})$ means $p(\boldsymbol{x}|\Theta)$ where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- To learn the GMM parameters Θ, we have to do MLE on $p(\boldsymbol{x})$

Learning GMM

- Let's do MLE for estimating GMM parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- We basically have to estimate K multivariate Gaussians with wts $\pi_1, \ldots, \pi_K$
- The conditional p.d.f. of a data point $\boldsymbol{x}$ if it comes from Gaussian k

$$p(\boldsymbol{x}|\boldsymbol{z}=k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Since $\boldsymbol{z}$ is NOT known, we need to look at the marginal p.d.f. of $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}) = \sum_{k=1}^K p(\boldsymbol{x}, \boldsymbol{z}=k) = \sum_{k=1}^K p(\boldsymbol{z}=k) p(\boldsymbol{x}|\boldsymbol{z}=k) = \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Note: Here $p(\boldsymbol{x})$ means $p(\boldsymbol{x}|\Theta)$ where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
- To learn the GMM parameters Θ, we have to do MLE on $p(\boldsymbol{x})$
- In general, it is not an easy problem due to the difficult form of $p(\boldsymbol{x})$ (for "why", see the next slide)

Learning GMM: The Difficulty

- Given N observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-likelihood will be

$$\mathcal{L} = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{params get coupled!}}$$

Learning GMM: The Difficulty

- Given N observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-likelihood will be

$$\mathcal{L} = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{params get coupled!}}$$

- Due to the coupling of parameters, MLE by simply taking derivatives of $\mathcal{L}$ and setting to zero won't give a closed form solution of $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$

Learning GMM: The Difficulty

- Given N observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-likelihood will be

$$\mathcal{L} = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{params get coupled!}}$$

- Due to the coupling of parameters, MLE by simply taking derivatives of $\mathcal{L}$ and setting to zero won't give a closed form solution of $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
  - Gradient based iterative methods can be used

Learning GMM: The Difficulty

- Given N observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-likelihood will be

$$\mathcal{L} = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{params get coupled!}}$$

- Due to the coupling of parameters, MLE by simply taking derivatives of $\mathcal{L}$ and setting to zero won't give a closed form solution of $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
  - Gradient based iterative methods can be used
  - However, we will use Expectation Maximization (EM) - a more general way of solving such problems (i.e., parameter estimation with latent variables)

Learning GMM: The Difficulty

- Given N observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-likelihood will be

$$\mathcal{L} = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{params get coupled!}}$$

- Due to the coupling of parameters, MLE by simply taking derivatives of $\mathcal{L}$ and setting to zero won't give a closed form solution of $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
  - Gradient based iterative methods can be used
  - However, we will use Expectation Maximization (EM) - a more general way of solving such problems (i.e., parameter estimation with latent variables)

- Note: For problems where $\boldsymbol{z}_n$ is continuous or comes from a combinatorially large space (e.g., it's a binary vector), doing MLE will be even harder!

$$\mathcal{L} = \sum_{n=1}^{N} \log \underbrace{\int_{\boldsymbol{z}_n} p(\boldsymbol{x}_n | \boldsymbol{z}_n) p(\boldsymbol{z}_n) d\boldsymbol{z}_n}_{\text{Ouch! Intractable integral!!!}}$$

Learning GMM: The Difficulty

- Given N observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the log-likelihood will be

$$\mathcal{L} = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \underbrace{\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{params get coupled!}}$$

- Due to the coupling of parameters, MLE by simply taking derivatives of $\mathcal{L}$ and setting to zero won't give a closed form solution of $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$
    - Gradient based iterative methods can be used
    - However, we will use Expectation Maximization (EM) - a more general way of solving such problems (i.e., parameter estimation with latent variables)

- Note: For problems where $\boldsymbol{z}_n$ is continuous or comes from a combinatorially large space (e.g., it's a binary vector), doing MLE will be even harder!

$$\mathcal{L} = \sum_{n=1}^{N} \log \underbrace{\int_{\boldsymbol{z}_n} p(\boldsymbol{x}_n | \boldsymbol{z}_n) p(\boldsymbol{z}_n) d\boldsymbol{z}_n}_{\text{Ouch! Intractable integral!!!}}$$

In such cases, something like EM becomes even more important

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$
  - $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is known as "complete data likelihood" ($\boldsymbol{z}_n$ makes $\boldsymbol{x}_n$ "complete")

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$
  - $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is known as "complete data likelihood" ($\boldsymbol{z}_n$ makes $\boldsymbol{x}_n$ "complete")
  - $p(\boldsymbol{x}_n)$ is known as "incomplete data likelihood"

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$
  - $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is known as "complete data likelihood" ($\boldsymbol{z}_n$ makes $\boldsymbol{x}_n$ "complete")
  - $p(\boldsymbol{x}_n)$ is known as "incomplete data likelihood"
- Since we don't know the "true" $\boldsymbol{z}_n$, we will have to rely on a "guess" for $\boldsymbol{z}_n$

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$
  - $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is known as "complete data likelihood" ($\boldsymbol{z}_n$ makes $\boldsymbol{x}_n$ "complete")
  - $p(\boldsymbol{x}_n)$ is known as "incomplete data likelihood"
- Since we don't know the "true" $\boldsymbol{z}_n$, we will have to rely on a "guess" for $\boldsymbol{z}_n$
  - This guess for $\boldsymbol{z}_n$ will be based on the current values of params Θ

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$
  - $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is known as "complete data likelihood" ($\boldsymbol{z}_n$ makes $\boldsymbol{x}_n$ "complete")
  - $p(\boldsymbol{x}_n)$ is known as "incomplete data likelihood"
- Since we don't know the "true" $\boldsymbol{z}_n$, we will have to rely on a "guess" for $\boldsymbol{z}_n$
  - This guess for $\boldsymbol{z}_n$ will be based on the current values of params Θ
  - Can do MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ to re-estimate Θ using these guesses, and repeat

Learning GMM by Making Guesses!

- MLE for GMM will be simplified if we "knew" the $\boldsymbol{z}_n$ for each $\boldsymbol{x}_n$
- Reason: If $\boldsymbol{z}_n$ is known, the summation over $\boldsymbol{z}_n$ isn't required

$$\sum_{n=1}^{N} \log \cancel{\sum_{k=1}^{K}} \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- With $\boldsymbol{z}_n$ "known", we can try doing MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$, instead of on $p(\boldsymbol{x}_n)$
  - $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is known as "complete data likelihood" ($\boldsymbol{z}_n$ makes $\boldsymbol{x}_n$ "complete")
  - $p(\boldsymbol{x}_n)$ is known as "incomplete data likelihood"
- Since we don't know the "true" $\boldsymbol{z}_n$, we will have to rely on a "guess" for $\boldsymbol{z}_n$
  - This guess for $\boldsymbol{z}_n$ will be based on the current values of params Θ
  - Can do MLE on $p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ to re-estimate Θ using these guesses, and repeat
  - A more formal view of this iterative procedure is given by the Expectation Maximization (EM) algorithm (next lecture)

Learning GMM

- The complete data log-likelihood over the N obs.

$$\sum_{n=1}^{N} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{z}_n)p(\boldsymbol{z}_n) = \sum_{n=1}^{N} \log \underbrace{\prod_{k=1}^{K} [p(\boldsymbol{x}_n|\boldsymbol{z}_n = k)p(\boldsymbol{z}_n = k)]^{z_{nk}}}_{\text{(note that, for each } n\text{, only one } z_{nk} \text{ will be 1)}}$$

Learning GMM

- The complete data log-likelihood over the N obs.

$$\sum_{n=1}^{N} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{z}_n)p(\boldsymbol{z}_n) = \sum_{n=1}^{N} \log \underbrace{\prod_{k=1}^{K} [p(\boldsymbol{x}_n|\boldsymbol{z}_n = k)p(\boldsymbol{z}_n = k)]^{z_{nk}}}_{\text{(note that, for each } n\text{, only one } z_{nk} \text{ will be 1)}}$$

- The above gets further simplified to

$$\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \log p(\boldsymbol{z}_n = k)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

Learning GMM

- The complete data log-likelihood over the N obs.

$$\sum_{n=1}^{N} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{z}_n)p(\boldsymbol{z}_n) = \sum_{n=1}^{N} \log \underbrace{\prod_{k=1}^{K}[p(\boldsymbol{x}_n|\boldsymbol{z}_n = k)p(\boldsymbol{z}_n = k)]^{z_{nk}}}_{\text{(note that, for each } n\text{, only one } z_{nk} \text{ will be 1)}}$$

- The above gets further simplified to

$$\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \log p(\boldsymbol{z}_n = k)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- If we know value of each z_{nk} deterministically, we can plug these in and do MLE on the above objective (which has a simple and separable structure)

Learning GMM

- The complete data log-likelihood over the N obs.

$$\sum_{n=1}^{N} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{z}_n)p(\boldsymbol{z}_n) = \sum_{n=1}^{N} \log \underbrace{\prod_{k=1}^{K} [p(\boldsymbol{x}_n|\boldsymbol{z}_n = k)p(\boldsymbol{z}_n = k)]^{z_{nk}}}_{\text{(note that, for each } n\text{, only one } z_{nk} \text{ will be 1)}}$$

- The above gets further simplified to

$$\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \log p(\boldsymbol{z}_n = k)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- If we know value of each z_{nk} deterministically, we can plug these in and do MLE on the above objective (which has a simple and separable structure)
- What if we don't have a *deterministic* guess for z_{nk}?

Learning GMM

- The complete data log-likelihood over the N obs.

$$\sum_{n=1}^{N} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{z}_n)p(\boldsymbol{z}_n) = \sum_{n=1}^{N} \log \underbrace{\prod_{k=1}^{K}[p(\boldsymbol{x}_n|\boldsymbol{z}_n = k)p(\boldsymbol{z}_n = k)]^{z_{nk}}}_{\text{(note that, for each } n\text{, only one } z_{nk} \text{ will be 1)}}$$

- The above gets further simplified to

$$\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \log p(\boldsymbol{z}_n = k)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- If we know value of each z_{nk} deterministically, we can plug these in and do MLE on the above objective (which has a simple and separable structure)
- What if we don't have a *deterministic* guess for z_{nk}?
  - In such cases, we can use the posterior expectations of the z_{nk}'s (which are basically posterior probabilities of cluster assignments of points to clusters)

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$\mathbb{E}[z_{nk}] = 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n)$$

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$
\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n) \\
&= p(z_{nk} = 1|\boldsymbol{x}_n)
\end{aligned}
$$

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$
\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n) \\
&= p(z_{nk} = 1|\boldsymbol{x}_n) \\
&\propto p(z_{nk} = 1)p(\boldsymbol{x}_n|z_{nk} = 1) \qquad \text{(Bayes Rule)}
\end{aligned}
$$

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$
\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n) \\
&= p(z_{nk} = 1|\boldsymbol{x}_n) \\
&\propto p(z_{nk} = 1)p(\boldsymbol{x}_n|z_{nk} = 1) \qquad \text{(Bayes Rule)} \\
\text{Thus} \quad \mathbb{E}[z_{nk}] &\propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(Posterior prob. of } \boldsymbol{x}_n \text{ belonging to cluster } k\text{)}
\end{aligned}
$$

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$
\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n) \\
&= p(z_{nk} = 1|\boldsymbol{x}_n) \\
&\propto p(z_{nk} = 1)p(\boldsymbol{x}_n|z_{nk} = 1) \qquad \text{(Bayes Rule)} \\
\text{Thus} \quad \mathbb{E}[z_{nk}] &\propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(Posterior prob. of } \boldsymbol{x}_n \text{ belonging to cluster } k\text{)}
\end{aligned}
$$

- The final expression of $\mathbb{E}[z_{nk}]$ makes intuitive sense

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$
\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n) \\
&= p(z_{nk} = 1|\boldsymbol{x}_n) \\
&\propto p(z_{nk} = 1)p(\boldsymbol{x}_n|z_{nk} = 1) \qquad \text{(Bayes Rule)} \\
\text{Thus} \quad \mathbb{E}[z_{nk}] &\propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(Posterior prob. of } \boldsymbol{x}_n \text{ belonging to cluster } k\text{)}
\end{aligned}
$$

- The final expression of $\mathbb{E}[z_{nk}]$ makes intuitive sense
- Note: We can finally normalize $\mathbb{E}[z_{nk}]$ as $\mathbb{E}[z_{nk}] = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$ since $\sum_{k=1}^{K} \mathbb{E}[z_{nk}] = 1$

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk} = 0|\boldsymbol{x}_n) + 1 \times p(z_{nk} = 1|\boldsymbol{x}_n) \\
&= p(z_{nk} = 1|\boldsymbol{x}_n) \\
&\propto p(z_{nk} = 1)p(\boldsymbol{x}_n|z_{nk} = 1) \qquad \text{(Bayes Rule)} \\
\text{Thus} \quad \mathbb{E}[z_{nk}] &\propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(Posterior prob. of } \boldsymbol{x}_n \text{ belonging to cluster } k\text{)}
\end{aligned}$$

- The final expression of $\mathbb{E}[z_{nk}]$ makes intuitive sense
- Note: We can finally normalize $\mathbb{E}[z_{nk}]$ as $\mathbb{E}[z_{nk}] = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$ since $\sum_{k=1}^{K} \mathbb{E}[z_{nk}] = 1$
- Given $\mathbb{E}[z_{nk}]$, we can now define expected complete data log-lik.

Learning GMM: Cluster Assignment Probabilities

- Posterior expectations $\mathbb{E}[z_{nk}]$ can be computed using current estimates of Θ

$$
\begin{aligned}
\mathbb{E}[z_{nk}] &= 0 \times p(z_{nk}=0|\boldsymbol{x}_n) + 1 \times p(z_{nk}=1|\boldsymbol{x}_n) \\
&= p(z_{nk}=1|\boldsymbol{x}_n) \\
&\propto p(z_{nk}=1)p(\boldsymbol{x}_n|z_{nk}=1) \qquad \text{(Bayes Rule)} \\
\text{Thus} \quad \mathbb{E}[z_{nk}] &\propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(Posterior prob. of } \boldsymbol{x}_n \text{ belonging to cluster } k\text{)}
\end{aligned}
$$

- The final expression of $\mathbb{E}[z_{nk}]$ makes intuitive sense
- Note: We can finally normalize $\mathbb{E}[z_{nk}]$ as $\mathbb{E}[z_{nk}] = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$ since $\sum_{k=1}^{K} \mathbb{E}[z_{nk}] = 1$
- Given $\mathbb{E}[z_{nk}]$, we can now define expected complete data log-lik.

$$
\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}[z_{nk}][\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]
$$

.. and do MLE for the parameters Θ using this as the objective function

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^K \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} [\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} [\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- Taking derivatives w.r.t. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $\forall k = 1, \ldots, K$

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} [\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- Taking derivatives w.r.t. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $\forall k = 1, \ldots, K$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \mu_k \text{ can be ignored)}$$

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} [\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- Taking derivatives w.r.t. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $\forall k = 1, \ldots, K$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \mu_k \text{ can be ignored)}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \boldsymbol{\Sigma}_k \text{ can be ignored)}$$

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- Taking derivatives w.r.t. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $\forall k = 1, \ldots, K$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \mu_k \text{ can be ignored)}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \boldsymbol{\Sigma}_k \text{ can be ignored)}$$

- For each k, it's a "weighted" version of the MLE problem for the multivariate Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, given observations $\{\boldsymbol{x}_n\}_{n=1}^{N}$ with weights $\{\gamma_{nk}\}_{n=1}^{N}$

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} [\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- Taking derivatives w.r.t. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $\forall k = 1, \dots, K$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \mu_k \text{ can be ignored)}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \sum_{n=1}^{N} \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \boldsymbol{\Sigma}_k \text{ can be ignored)}$$

- For each k, it's a "weighted" version of the MLE problem for the multivariate Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, given observations $\{\boldsymbol{x}_n\}_{n=1}^{N}$ with weights $\{\gamma_{nk}\}_{n=1}^{N}$

- Can also solve for π_k likewise (subject to contraint $\sum_{k=1}^{K} \pi_k = 1$)

Learning GMM: Component Parameters

- Given $\mathbb{E}[z_{nk}] = \gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{\ell=1}^K \pi_\ell \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell)}$, the expected complete data log-lik.

$$\mathcal{L} = \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

- Taking derivatives w.r.t. $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $\forall k = 1, \ldots, K$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{n=1}^N \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \mu_k \text{ can be ignored)}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \sum_{n=1}^N \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = 0 \qquad \text{(note: constants w.r.t. } \boldsymbol{\Sigma}_k \text{ can be ignored)}$$

- For each k, it's a "weighted" version of the MLE problem for the multivariate Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, given observations $\{\boldsymbol{x}_n\}_{n=1}^N$ with weights $\{\gamma_{nk}\}_{n=1}^N$
- Can also solve for π_k likewise (subject to contraint $\sum_{k=1}^K \pi_k = 1$)
- Derivations are a bit tedious (but straightforward). I will provide a note.

GMM Parameter Update Equations

- The final expressions for updates of $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

$$
\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}\boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \\
\pi_k &= \frac{N_k}{N}
\end{aligned}
$$

GMM Parameter Update Equations

- The final expressions for updates of $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}\boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_{n=1}^N \gamma_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \\
\pi_k &= \frac{N_k}{N}
\end{aligned}$$

- Note: $N_k = \sum_{n=1}^N \gamma_{nk}$ is the effective num. of pts. assigned to Gaussian k
- Update equations make intuitive sense

GMM Parameter Update Equations

- The final expressions for updates of $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

$$
\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}\boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \\
\pi_k &= \frac{N_k}{N}
\end{aligned}
$$

- Note: $N_k = \sum_{n=1}^{N} \gamma_{nk}$ is the effective num. of pts. assigned to Gaussian k
- Update equations make intuitive sense
- Also note that each point $\boldsymbol{x}_n$ contributes to each $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ but fractionally (based on the values of γ_{nk})

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ randomly, or using K-means

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ randomly, or using K-means
- Iterate until convergence (e.g., when $\log p(\boldsymbol{x}|\Theta)$ ceases to increase)

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ randomly, or using K-means
- Iterate until convergence (e.g., when $\log p(\boldsymbol{x}|\Theta)$ ceases to increase)
  - Given Θ, compute each expectation z_{nk} (post. prob. of $z_{nk} = 1$), $\forall n, k$

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ randomly, or using K-means
- Iterate until convergence (e.g., when $\log p(\boldsymbol{x}|\Theta)$ ceases to increase)
  - Given Θ, compute each expectation z_{nk} (post. prob. of $z_{nk} = 1$), $\forall n, k$

$$\gamma_{nk} = \mathbb{E}[z_{nk}] \propto \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(and re-normalize s.t. } \sum_{k=1}^K \gamma_{nk} = 1\text{)}$$

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ randomly, or using K-means
- Iterate until convergence (e.g., when $\log p(\boldsymbol{x}|\Theta)$ ceases to increase)
  - Given Θ, compute each expectation z_{nk} (post. prob. of $z_{nk} = 1$), $\forall n, k$

$$\gamma_{nk} = \mathbb{E}[z_{nk}] \propto \pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(and re-normalize s.t. } \sum_{k=1}^{K} \gamma_{nk} = 1)$$

  - Given $\gamma_{nk} = \mathbb{E}[z_{nk}]$ and $N_k = \sum_{n=1}^N \gamma_{nk}$, update $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ as

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ randomly, or using K-means
- Iterate until convergence (e.g., when $\log p(\boldsymbol{x}|\Theta)$ ceases to increase)
  - Given Θ, compute each expectation z_{nk} (post. prob. of $z_{nk} = 1$), $\forall n, k$

$$\gamma_{nk} = \mathbb{E}[z_{nk}] \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(and re-normalize s.t. } \sum_{k=1}^K \gamma_{nk} = 1\text{)}$$

  - Given $\gamma_{nk} = \mathbb{E}[z_{nk}]$ and $N_k = \sum_{n=1}^N \gamma_{nk}$, update $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ as

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} \boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \\
\pi_k &= \frac{N_k}{N}
\end{aligned}$$

The Full Algorithm for Learning GMM

- Initialize the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ randomly, or using K-means
- Iterate until convergence (e.g., when $\log p(\boldsymbol{x}|\Theta)$ ceases to increase)
  - Given Θ, compute each expectation z_{nk} (post. prob. of $z_{nk} = 1$), $\forall n, k$

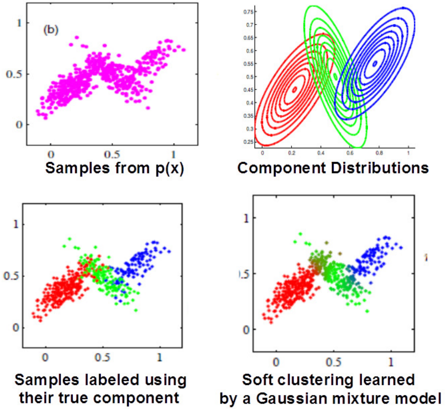
$$\gamma_{nk} = \mathbb{E}[z_{nk}] \propto \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{(and re-normalize s.t. } \sum_{k=1}^K \gamma_{nk} = 1\text{)}$$

  - Given $\gamma_{nk} = \mathbb{E}[z_{nk}]$ and $N_k = \sum_{n=1}^N \gamma_{nk}$, update $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ as

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} \boldsymbol{x}_n \\ \boldsymbol{\Sigma}_k &= \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \\ \pi_k &= \frac{N_k}{N} \end{aligned}$$

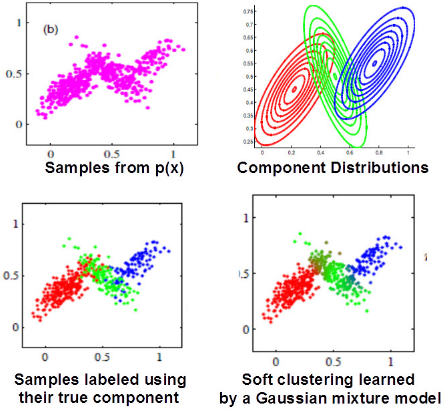
(It's basically an Expectation Maximization (EM) algorithm for learning GMM. We will look at EM more formally in the next class.)

Illustration of GMM Clustering



Notice the "mixed" colored points in the overlapping regions in the final clustering

Illustration of GMM Clustering

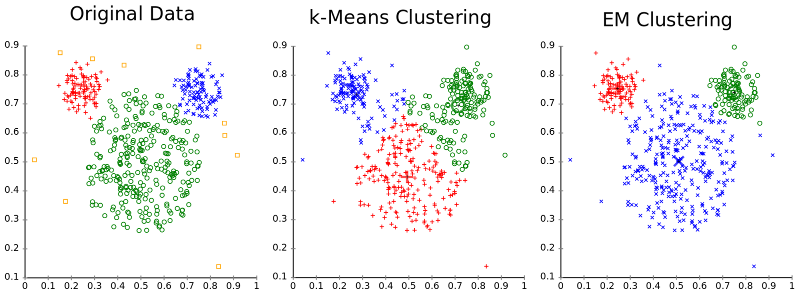


Notice the “mixed” colored points in the overlapping regions in the final clustering

Also check out this demo of GMM: `https://www.youtube.com/watch?v=B36fzChfyGU`

GMM vs K-means

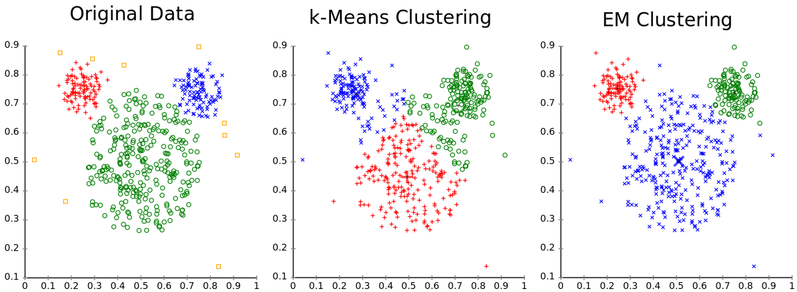
For the GMM clustering (rightmost figure), the most probable cluster for each point has been labeled



Pic courtesy: `https://en.wikipedia.org/wiki/Expectation-maximization_algorithm/`

GMM vs K-means

For the GMM clustering (rightmost figure), the most probable cluster for each point has been labeled

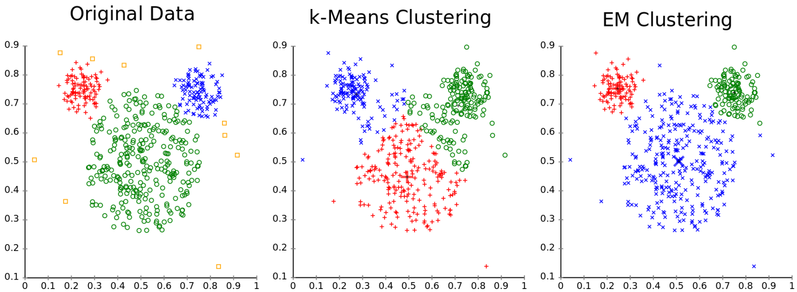


Note that K-means, unlike GMM, tends to learn equi-sized clusters.

Pic courtesy: `https://en.wikipedia.org/wiki/Expectation-maximization_algorithm/`

GMM vs K-means

For the GMM clustering (rightmost figure), the most probable cluster for each point has been labeled



Note that K-means, unlike GMM, tends to learn equi-sized clusters.

GMM with $\Sigma_k = \mathbf{I}$ and $\pi_k = 1/K$, and soft assignments converted to hard assign. (setting the largest prob. to 1, rest to 0), is equivalent to K-means.

Pic courtesy: `https://en.wikipedia.org/wiki/Expectation-maximization_algorithm/`

Mixture Models: Some Extensions

- Other types of component distributions can be used

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)
  - For data $\boldsymbol{x}_n$, first choose one of K experts, and then use that expert's predictive model for $p(y_n|\boldsymbol{x}_n)$

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)
  - For data $\boldsymbol{x}_n$, first choose one of K experts, and then use that expert's predictive model for $p(y_n|\boldsymbol{x}_n)$
  - Experts and points-to-experts assignments can be learned iteratively as in GMM

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)
  - For data $\boldsymbol{x}_n$, first choose one of K experts, and then use that expert's predictive model for $p(y_n|\boldsymbol{x}_n)$
  - Experts and points-to-experts assignments can be learned iteratively as in GMM
- Can be used for performing generative classification (e.g., naïve Bayes)

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)
  - For data $\boldsymbol{x}_n$, first choose one of K experts, and then use that expert's predictive model for $p(y_n|\boldsymbol{x}_n)$
  - Experts and points-to-experts assignments can be learned iteratively as in GMM
- Can be used for performing generative classification (e.g., naïve Bayes)

$$p(y_n = k|\boldsymbol{x}_n) \propto p(y_n = k)p(\boldsymbol{x}_n|y_n = k) \quad \text{(cluster ids are the known classes)}$$

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)
  - For data $\boldsymbol{x}_n$, first choose one of K experts, and then use that expert's predictive model for $p(y_n|\boldsymbol{x}_n)$
  - Experts and points-to-experts assignments can be learned iteratively as in GMM
- Can be used for performing generative classification (e.g., naïve Bayes)

$$p(y_n = k|\boldsymbol{x}_n) \propto p(y_n = k)p(\boldsymbol{x}_n|y_n = k) \quad \text{(cluster ids are the known classes)}$$

$p(y = k)$ and $p(\boldsymbol{x}|\boldsymbol{z} = k)$ can be efficiently estimated using training data

Mixture Models: Some Extensions

- Other types of component distributions can be used
- Sequence data models, such as HMM, can also be seen as mixture models

$$p(\boldsymbol{z}_n|\phi) = p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(Current cluster/state depends on previous)}$$

- Also used in supervised learning problems (mixture of experts)
  - For data $\boldsymbol{x}_n$, first choose one of K experts, and then use that expert's predictive model for $p(y_n|\boldsymbol{x}_n)$
  - Experts and points-to-experts assignments can be learned iteratively as in GMM
- Can be used for performing generative classification (e.g., naïve Bayes)

$$p(y_n = k|\boldsymbol{x}_n) \propto p(y_n = k)p(\boldsymbol{x}_n|y_n = k) \quad \text{(cluster ids are the known classes)}$$

$p(y = k)$ and $p(\boldsymbol{x}|\boldsymbol{z} = k)$ can be efficiently estimated using training data

- Number of clusters (K) in a mixture model can be learned from data using nonparametric Bayesian methods (e.g., "infinite" mixture models)

Next Class

- The general Expectation Maximization (EM) algorithm
- Generative models for dimensionality reduction
    - Factor Analysis and Probabilistic PCA (and extensions)
    - EM based parameter estimation for these models