

CSCI567 Machine Learning (Fall 2025)

Haipeng Luo

University of Southern California

Oct 24, 2025

Administration

Exam 1 grading still ongoing.

HW3 will be available after today's lecture.

Outline

- 1 Clustering
- 2 Gaussian mixture models

Outline

- 1 Clustering
 - Problem setup
 - K-means algorithm
 - Initialization and Convergence
- 2 Gaussian mixture models

Unsupervised learning

Recall there are different types of machine learning problems

- **supervised learning** (what we have discussed so far)
Aim to **predict** accurately, e.g., classification and regression
- **unsupervised learning** (this and next few lectures)
Aim to **discover hidden/latent patterns and explore data**
- **decision making** (last two lectures)
Aim to **act optimally under uncertainty**

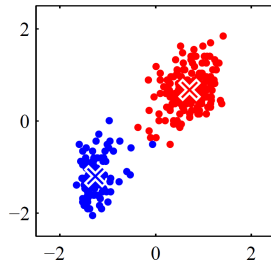
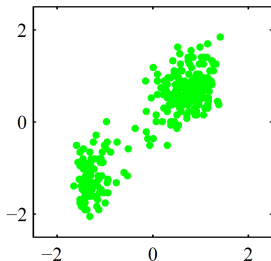
Today's focus: **clustering**, a canonical unsupervised learning problem

Clustering: informal definition

Given: a set of data points (feature vectors), *without labels*

Output: group the data into some clusters, which means

- **assign** each point to a specific cluster
- find the **center** (representative/prototype/...) of each cluster

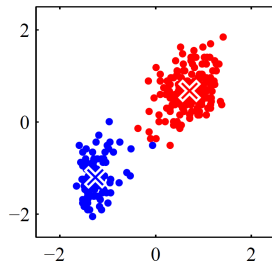
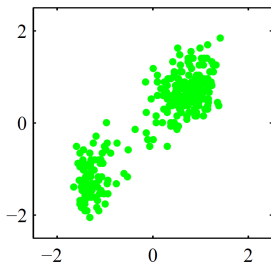


Clustering: formal definition

Given: data points $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^D$ and #clusters K we want

Output: group the data into K clusters, which means

- find **assignment** $\gamma_{nk} \in \{0, 1\}$ for each data point $n \in [N]$ and $k \in [K]$
s.t. $\sum_{k \in [K]} \gamma_{nk} = 1$ for any fixed n
- find the cluster **centers** $\mu_1, \dots, \mu_K \in \mathbb{R}^D$



Many applications

- recognize communities in a social network
- group similar customers in market research
- grouping documents into different topics
- image segmentation
- identifying groups of genes with similar expression patterns
- accelerate other algorithms (e.g. NNC as in programing projects)
- ...

One example

image compression:

- each pixel is a point
- perform clustering over these points
- **replace each point by the center** of the cluster it belongs to



Original image

Large $K \rightarrow$ Small K

Formal Objective

Key difference from supervised learning problems: no labels given, which means *no ground-truth to even measure the quality of your answer!*

Still, we can turn it into an optimization problem, e.g. through the popular **“K-means” objective**: find γ_{nk} and μ_k to minimize

$$F(\{\gamma_{nk}\}, \{\mu_k\}) = \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} \|\mathbf{x}_n - \mu_k\|_2^2$$

i.e. the **sum of squared distances of each point to its center**.

Unfortunately, finding the exact minimizer is *NP-hard!*

Alternating minimization

Instead, use a heuristic that **alternatingly minimizes over $\{\gamma_{nk}\}$ and $\{\mu_k\}$** :

Initialize $\{\mu_k^{(1)}\}$

For $t = 1, 2, \dots$

- find

$$\{\gamma_{nk}^{(t+1)}\} = \operatorname{argmin}_{\{\gamma_{nk}\}} F\left(\{\gamma_{nk}\}, \{\mu_k^{(t)}\}\right)$$

- find

$$\{\mu_k^{(t+1)}\} = \operatorname{argmin}_{\{\mu_k\}} F\left(\{\gamma_{nk}^{(t+1)}\}, \{\mu_k\}\right)$$

A closer look

The first step

$$\begin{aligned}\min_{\{\gamma_{nk}\}} F(\{\gamma_{nk}\}, \{\boldsymbol{\mu}_k\}) &= \min_{\{\gamma_{nk}\}} \sum_n \sum_k \gamma_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \\ &= \sum_n \min_{\{\gamma_{nk}\}} \sum_k \gamma_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2\end{aligned}$$

is simply to **assign each $\mathbf{x}_n$ to the closest $\boldsymbol{\mu}_k$** , i.e.

$$\gamma_{nk} = \begin{cases} 1, & \text{if } k = \operatorname{argmin}_c \|\mathbf{x}_n - \boldsymbol{\mu}_c\|_2^2 \\ 0, & \text{else} \end{cases}$$

for all $k \in [K]$ and $n \in [N]$.

A closer look

The second step

$$\begin{aligned}\min_{\{\boldsymbol{\mu}_k\}} F(\{\gamma_{nk}\}, \{\boldsymbol{\mu}_k\}) &= \min_{\{\boldsymbol{\mu}_k\}} \sum_n \sum_k \gamma_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \\ &= \sum_k \min_{\boldsymbol{\mu}_k} \sum_{n:\gamma_{nk}=1} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2\end{aligned}$$

is simply **to average the points of each cluster** (hence the name)

$$\boldsymbol{\mu}_k = \frac{\sum_{n:\gamma_{nk}=1} \mathbf{x}_n}{|\{n : \gamma_{nk} = 1\}|} = \frac{\sum_n \gamma_{nk} \mathbf{x}_n}{\sum_n \gamma_{nk}}$$

for each $k \in [K]$.

The K-means algorithm

Step 0 Initialize $\mu_1, \dots, \mu_K$

Step 1 Fix the centers $\mu_1, \dots, \mu_K$, assign each point to the closest center:

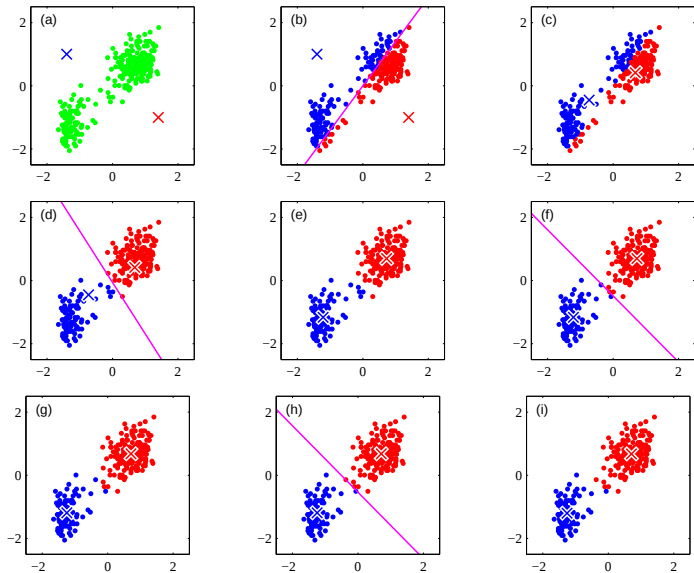
$$\gamma_{nk} = \begin{cases} 1, & \text{if } k = \operatorname{argmin}_c \|\mathbf{x}_n - \mu_c\|_2^2 \\ 0, & \text{else} \end{cases}$$

Step 2 Fix the assignment $\{\gamma_{nk}\}$, update the centers

$$\mu_k = \frac{\sum_n \gamma_{nk} \mathbf{x}_n}{\sum_n \gamma_{nk}}$$

Step 3 Return to Step 1 if not converged

An example



How to initialize?

There are **different ways to initialize**:

- randomly pick K points as initial centers
- or randomly assign each point to a cluster, then average
- or more sophisticated approaches (e.g. greedy or **K-means++**)

Initialization matters for **convergence**.

Convergence

K-means will **converge in a finite number of iterations**, why?

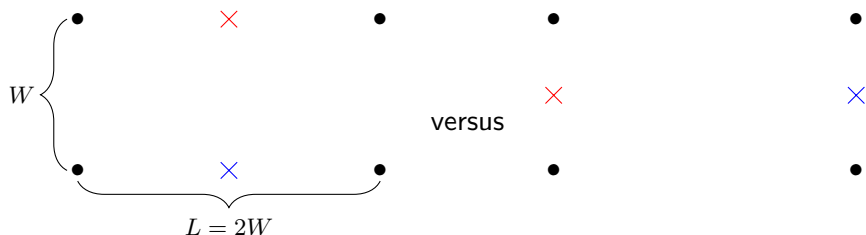
- objective decreases at each step
- objective is lower bounded by 0
- #possible_assignments is finite (K^N , exponentially large though)

However

- it could take *exponentially many iterations* to converge
- and it *might not converge to the global minimum* of the K-means objective

Local minimum v.s global minimum

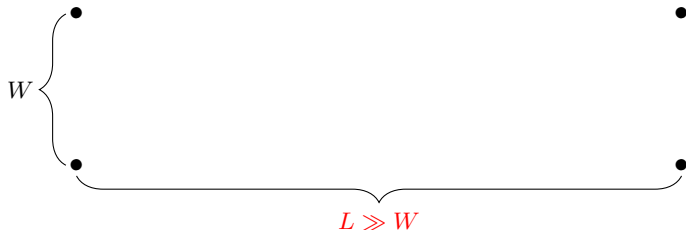
Simple example: 4 data points, 2 clusters, 2 different initializations



K-means converges immediately in both cases, but

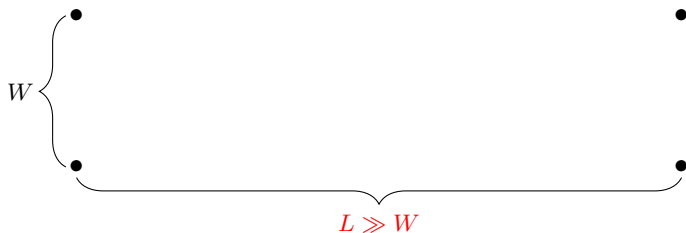
- left has K-means objective $L^2 = 4W^2$
- right has K-means objective W^2 , *4 times better than left!*
- in fact, left is **local minimum**, and right is **global minimum**.

Local minimum v.s global minimum



- moreover, local minimum can be *arbitrarily worse* if we increase L
- so *initialization matters a lot* for K-means

How common initialization methods perform?



- randomly pick K points as initial centers: fails with $1/3$ probability
- or randomly assign each point to a cluster, then average: similarly fail with a constant probability
- or more sophisticated approaches?

An Intuitive and Greedy Approach

Idea: *spread out the initial centers*

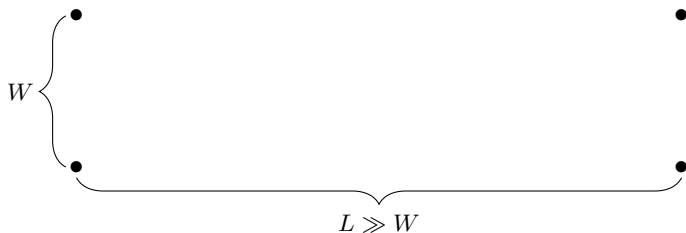
Start with a random data point as the first center μ_1

For $k = 2, \dots, K$

- let the k -th center μ_k be the farthest from the chosen centers:

$$\mu_k = \underset{x \in \{x_1, \dots, x_N\}}{\operatorname{argmax}} \min_{j=1, \dots, k-1} \|x - \mu_j\|_2^2$$

Greedy initialization on the same example



Suppose we pick top left as μ_1 , then μ_2 is the bottom right

- K-means converges to the **global minimum** after one iteration!

See demo at <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>.

Does greedy initialization always work?

Not really; *it is too sensitive to outliers!*



● outlier



In HW3, you will verify that greedy initialization can also lead to *arbitrarily bad* local minimums.

Solution: K-means++

K-means++: robustify the greedy approach via **randomness**

Start with a random data point as the first center μ_1

For $k = 2, \dots, K$

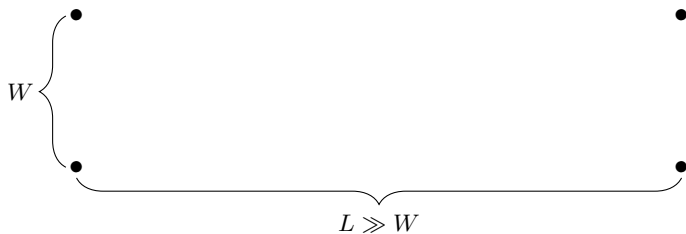
- randomly pick the k -th center μ_k such that

~~$$\mu_k = \operatorname{argmax}_{x \in \{x_1, \dots, x_N\}} \min_{j=1, \dots, k-1} \|x - \mu_j\|_2^2$$~~

$$\Pr[\mu_k = x_n] \propto \min_{j=1, \dots, k-1} \|x_n - \mu_j\|_2^2$$

K-means++ *guarantees* to find a solution that in expectation is at most $O(\log K)$ times of the global optimal.

K-means++ on the same example



Suppose we pick top left as μ_1 , then

- $\Pr[\mu_2 = \text{bottom left}] \propto W^2$, $\Pr[\mu_2 = \text{top right}] \propto L^2$
- $\Pr[\mu_2 = \text{bottom right}] \propto W^2 + L^2$

So the **expected K-means objective** is

$$\frac{W^2}{2(W^2 + L^2)} \cdot L^2 + \left(\frac{L^2}{2(W^2 + L^2)} + \frac{1}{2} \right) \cdot W^2 \leq \frac{3}{2}W^2,$$

that is, *at most 1.5 times of the optimal*.

Summary for K-means

K-means is alternating minimization for the K-means objective.

The initialization matters a lot for the convergence.

K-means++ uses a theoretically (and often empirically) better initialization.

Outline

- 1 Clustering
- 2 Gaussian mixture models
 - Motivation and Model
 - EM algorithm
 - EM applied to GMMs

Gaussian mixture models

Gaussian mixture models (GMM) is a probabilistic approach for clustering

- more explanatory than minimizing the K-means objective
- can be seen as a soft version of K-means

To solve GMM, we will introduce a powerful method for learning probabilistic model: **Expectation–Maximization (EM) algorithm**

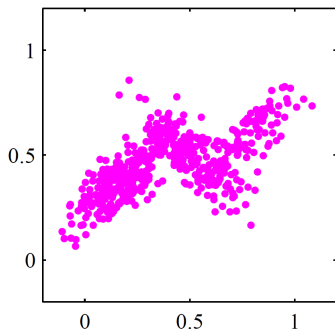
A generative model

For classification, we discussed the sigmoid model to “explain” how the labels are generated.

Similarly, for clustering, we want to come up with a probabilistic model p to **“explain” how the data is generated.**

That is, each point is **an independent sample** of $x \sim p$.

What probabilistic model generates data like this?

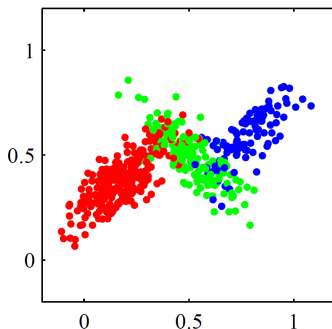


GMM: intuition

GMM is a natural model to explain such data

Assume there are 3 ground-truth Gaussian models. To generate a point, we

- first **randomly pick one of the Gaussian models**,
- then **draw a point according this Gaussian**.



Hence the name “**Gaussian mixture model**”.

GMM: formal definition

A GMM has the following density function:

$$p(\mathbf{x}) = \sum_{k=1}^K \omega_k N(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where

- K : the number of **Gaussian components** (same as #clusters we want)
- $\omega_1, \dots, \omega_K$: **mixture weights**, a distribution over K components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: **mean and covariance matrix** of the k -th Gaussian
- N : the density function for a Gaussian

Another view

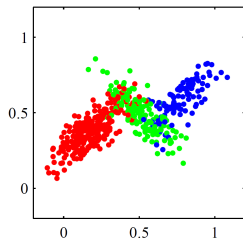
By introducing a **latent variable** $z \in [K]$, which indicates cluster membership, we can see p as a **marginal distribution**

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

$\mathbf{x}$ and z are both random variables drawn from the model

- $\mathbf{x}$ is **observed**
- z is **unobserved/latent**

An example

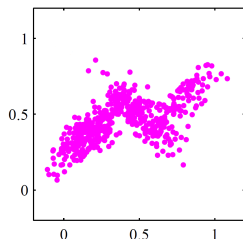


The conditional distributions are

$$p(\mathbf{x} \mid z = \text{red}) = N(\mathbf{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$

$$p(\mathbf{x} \mid z = \text{blue}) = N(\mathbf{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

$$p(\mathbf{x} \mid z = \text{green}) = N(\mathbf{x} \mid \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is

$$p(\mathbf{x}) = p(\text{red})N(\mathbf{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(\text{blue})N(\mathbf{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ + p(\text{green})N(\mathbf{x} \mid \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Learning GMMs

Learning a GMM means finding all the parameters $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$.

In the process, we will learn the latent variable z_n as well:

$$p(z_n = k \mid \mathbf{x}_n) \triangleq \gamma_{nk} \in [0, 1]$$

i.e. “soft assignment” of each point to each cluster, as opposed to “hard assignment” by K-means.

GMM is more explanatory than K-means

- both learn the cluster centers μ_k 's
- in addition, GMM learns cluster weight ω_k and covariance Σ_k , thus
 - we can *predict probability of seeing a new point*
 - we can *generate synthetic data*

How to learn these parameters?

An obvious attempt is **maximum-likelihood estimation (MLE)**: find

$$\operatorname{argmax}_{\boldsymbol{\theta}} \ln \prod_{n=1}^N p(\mathbf{x}_n ; \boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^N \ln p(\mathbf{x}_n ; \boldsymbol{\theta}) \triangleq \operatorname{argmax}_{\boldsymbol{\theta}} P(\boldsymbol{\theta})$$

This is called **incomplete log-likelihood** (since z_n 's are unobserved), and is *intractable in general* (non-concave problem).

One solution is to still apply GD/SGD, but a much more effective approach is the **Expectation–Maximization (EM) algorithm**.

Preview of EM for learning GMMs

Step 0 Initialize $\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for each $k \in [K]$

Step 1 (E-Step) **update the “soft assignment”** (fixing parameters)

$$\gamma_{nk} = p(z_n = k \mid \mathbf{x}_n) \propto \omega_k N(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Step 2 (M-Step) **update the model parameter** (fixing assignments)

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N} \quad \boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk} \mathbf{x}_n}{\sum_n \gamma_{nk}}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Step 3 return to Step 1 if not converged

We will see how this is **a special case of EM**.

Demo

Generate 50 data points from a mixture of 2 Gaussians with

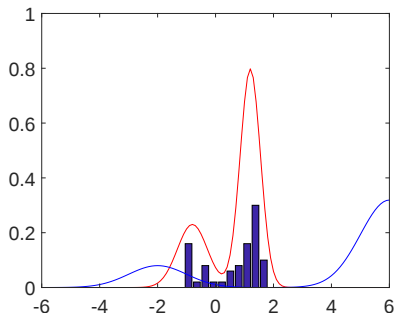
- $\omega_1 = 0.3, \mu_1 = -0.8, \Sigma_1 = 0.52$
- $\omega_2 = 0.7, \mu_2 = 1.2, \Sigma_2 = 0.35$

histogram represents the data

red curve represents the
ground-truth density

$$p(\mathbf{x}) = \sum_{k=1}^K \omega_k N(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

blue curve represents the learned
density for a specific round



EM_demo_1D.pdf shows how the blue curve moves towards red curve quickly via EM

EM algorithm

In general EM is **a heuristic to solve MLE with latent variables** (not just GMM), i.e. find the maximizer of

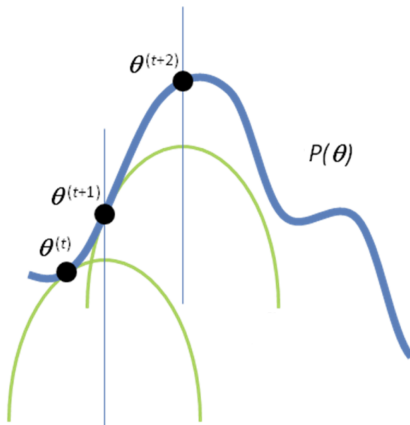
$$P(\boldsymbol{\theta}) = \sum_{n=1}^N \ln p(\mathbf{x}_n ; \boldsymbol{\theta}) = \sum_{n=1}^N \ln \int_{z_n} p(\mathbf{x}_n, z_n ; \boldsymbol{\theta}) dz_n$$

- $\boldsymbol{\theta}$ is the **parameters** for a general probabilistic model
- $\mathbf{x}_n$'s are **observed random variables**
- z_n 's are **latent variables**

Again, directly solving the objective is intractable.

High level idea

Keep maximizing **a lower bound of P that is more manageable**



Derivation of EM (not required)

Finding the lower bound of P :

$$\ln p(\mathbf{x} ; \boldsymbol{\theta}) = \ln \frac{p(\mathbf{x}, z ; \boldsymbol{\theta})}{p(z|\mathbf{x} ; \boldsymbol{\theta})} \quad (\text{true for any } z)$$

$$= \mathbb{E}_{z \sim q} \left[\ln \frac{p(\mathbf{x}, z ; \boldsymbol{\theta})}{p(z|\mathbf{x} ; \boldsymbol{\theta})} \right] \quad (\text{true for any dist. } q)$$

$$= \mathbb{E}_{z \sim q} [\ln p(\mathbf{x}, z ; \boldsymbol{\theta})] - \mathbb{E}_{z \sim q} [\ln q(z)] - \mathbb{E}_{z \sim q} \left[\ln \frac{p(z|\mathbf{x} ; \boldsymbol{\theta})}{q(z)} \right]$$

$$= \mathbb{E}_{z \sim q} [\ln p(\mathbf{x}, z ; \boldsymbol{\theta})] + H(q) - \mathbb{E}_{z \sim q} \left[\ln \frac{p(z|\mathbf{x} ; \boldsymbol{\theta})}{q(z)} \right] \quad (H \text{ is entropy})$$

$$\geq \mathbb{E}_{z \sim q} [\ln p(\mathbf{x}, z ; \boldsymbol{\theta})] + H(q) - \ln \mathbb{E}_{z \sim q} \left[\frac{p(z|\mathbf{x} ; \boldsymbol{\theta})}{q(z)} \right] \quad (\text{Jensen's inequality})$$

$$= \mathbb{E}_{z \sim q} [\ln p(\mathbf{x}, z ; \boldsymbol{\theta})] + H(q)$$

Alternatively maximize the lower bound

Therefore, we obtain a lower bound for the log-likelihood function

$$\begin{aligned} P(\boldsymbol{\theta}) &= \sum_{n=1}^N \ln p(\mathbf{x}_n ; \boldsymbol{\theta}) \\ &\geq \sum_{n=1}^N (\mathbb{E}_{z_n \sim q_n} [\ln p(\mathbf{x}_n, z_n ; \boldsymbol{\theta})] + H(q_n)) = F(\boldsymbol{\theta}, \{q_n\}) \end{aligned}$$

This holds for *any* $\{q_n\}$, so how do we choose? Naturally, *the one that maximizes the lower bound* (i.e. the tightest lower bound)!

Equivalently, this is the same as *alternatingly maximizing F over $\{q_n\}$ and $\boldsymbol{\theta}$* (similar to K-means).

Maximizing over $\{q_n\}$

Fix $\boldsymbol{\theta}^{(t)}$, the solution to

$$\operatorname{argmax}_{q_n} \mathbb{E}_{z_n \sim q_n} \left[\ln p(\mathbf{x}_n, z_n; \boldsymbol{\theta}^{(t)}) \right] + H(q_n)$$

turns out to be $q_n^{(t)}$ s.t.

$$q_n^{(t)}(z_n) = p(z_n \mid \mathbf{x}_n; \boldsymbol{\theta}^{(t)}) \propto p(\mathbf{x}_n, z_n; \boldsymbol{\theta}^{(t)})$$

i.e., the *posterior distribution* of z_n given $\mathbf{x}_n$ and $\boldsymbol{\theta}^{(t)}$.

So at $\boldsymbol{\theta}^{(t)}$, we found the tightest lower bound $F(\boldsymbol{\theta}, \{q_n^{(t)}\})$:

- $F(\boldsymbol{\theta}, \{q_n^{(t)}\}) \leq P(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$.
- $F(\boldsymbol{\theta}^{(t)}, \{q_n^{(t)}\}) = P(\boldsymbol{\theta}^{(t)})$ (can be verified through Slide 40)

Maximizing over θ

Fix $\{q_n^{(t)}\}$, maximize over θ :

$$\begin{aligned} & \operatorname{argmax}_{\theta} F\left(\theta, \{q_n^{(t)}\}\right) \\ &= \operatorname{argmax}_{\theta} \sum_{n=1}^N \mathbb{E}_{z_n \sim q_n^{(t)}} [\ln p(\mathbf{x}_n, z_n ; \theta)] \quad (H(q_n^{(t)}) \text{ is independent of } \theta) \\ &\triangleq \operatorname{argmax}_{\theta} Q(\theta ; \theta^{(t)}) \quad (\{q_n^{(t)}\} \text{ are computed via } \theta^{(t)}) \end{aligned}$$

Q is the (expected) **complete likelihood** and is usually more tractable.

- versus the incomplete likelihood: $P(\theta) = \sum_{n=1}^N \ln p(\mathbf{x}_n ; \theta)$

General EM algorithm

Step 0 Initialize $\theta^{(1)}$, $t = 1$

Step 1 (E-Step) **update the posterior of latent variables**

$$q_n^{(t)}(\cdot) = p(\cdot \mid \mathbf{x}_n ; \theta^{(t)})$$

and obtain **Expectation** of complete likelihood

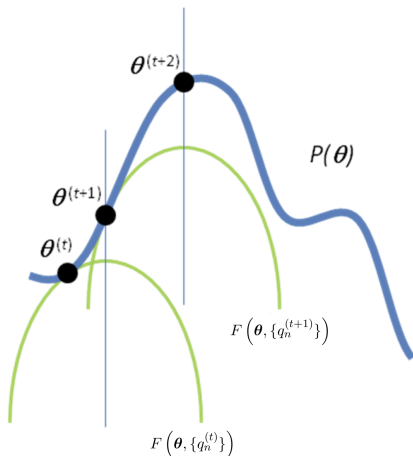
$$Q(\theta ; \theta^{(t)}) = \sum_{n=1}^N \mathbb{E}_{z_n \sim q_n^{(t)}} [\ln p(\mathbf{x}_n, z_n ; \theta)]$$

Step 2 (M-Step) **update the model parameter** via **Maximization**

$$\theta^{(t+1)} \leftarrow \operatorname{argmax}_{\theta} Q(\theta ; \theta^{(t)})$$

Step 3 $t \leftarrow t + 1$ and return to Step 1 if not converged

Pictorial explanation



$P(\theta)$ is non-concave, but $Q(\theta; \theta^{(t)})$ often is concave and easy to maximize.

$$\begin{aligned}
 P(\theta^{(t+1)}) &\geq F\left(\theta^{(t+1)}; \{q_n^{(t)}\}\right) \\
 &\geq F\left(\theta^{(t)}; \{q_n^{(t)}\}\right) \\
 &= P(\theta^{(t)})
 \end{aligned}$$

So **EM always increases the objective value** and will **converge to some local maximum** (similar to K-means).

Apply EM to learn GMMs

E-Step:

$$\begin{aligned} q_n^{(t)}(z_n = k) &= p(z_n = k \mid \mathbf{x}_n; \boldsymbol{\theta}^{(t)}) \\ &\propto p(\mathbf{x}_n, z_n = k; \boldsymbol{\theta}^{(t)}) \\ &= p(z_n = k; \boldsymbol{\theta}^{(t)}) p(\mathbf{x}_n \mid z_n = k; \boldsymbol{\theta}^{(t)}) \\ &= \omega_k^{(t)} N(\mathbf{x}_n \mid \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) \end{aligned}$$

This computes the “soft assignment” $\gamma_{nk} = q_n^{(t)}(z_n = k)$, i.e. conditional probability of $\mathbf{x}_n$ belonging to cluster k .

Apply EM to learn GMMs

M-Step:

$$\begin{aligned}
 \operatorname{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^N \mathbb{E}_{z_n \sim q_n^{(t)}} [\ln p(\mathbf{x}_n, z_n; \boldsymbol{\theta})] \\
 &= \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^N \mathbb{E}_{z_n \sim q_n^{(t)}} [\ln p(z_n; \boldsymbol{\theta}) + \ln p(\mathbf{x}_n | z_n; \boldsymbol{\theta})] \\
 &= \operatorname{argmax}_{\{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}} \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} (\ln \omega_k + \ln N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))
 \end{aligned}$$

To find $\omega_1, \dots, \omega_K$, solve

$$\operatorname{argmax}_{\boldsymbol{\omega}} \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} \ln \omega_k$$

To find each $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, solve

$$\operatorname{argmax}_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \sum_{n=1}^N \gamma_{nk} \ln N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

M-Step (continued)

Solutions to previous two problems are very natural, for each k

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N}$$

i.e. (weighted) fraction of examples belonging to cluster k

$$\boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk} \mathbf{x}_n}{\sum_n \gamma_{nk}}$$

i.e. (weighted) average of examples belonging to cluster k

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

i.e (weighted) covariance of examples belonging to cluster k

(You can try to verify these for the 1D case.)

Putting it together

EM for learning GMMs: (see 2D demo)

Step 0 Initialize $\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for each $k \in [K]$

Step 1 (E-Step) **update the “soft assignment”** (fixing parameters)

$$\gamma_{nk} = p(z_n = k \mid \mathbf{x}_n) \propto \omega_k N(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Step 2 (M-Step) **update the model parameter** (fixing assignments)

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N} \quad \boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk} \mathbf{x}_n}{\sum_n \gamma_{nk}}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Step 3 return to Step 1 if not converged

Connection to K-means

K-means is in fact a special case of EM for (a simplified) GMM:

- assume $\Sigma_k = \sigma^2 \mathbf{I}$ for some fixed σ so only ω_k and μ_k are parameters
- when $\sigma \rightarrow 0$, EM becomes K-means

GMM is a soft version of K-means and it provides a probabilistic interpretation of the data, which means we can predict and generate data after learning.