Lecture 10

Clustering

Informal definition

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

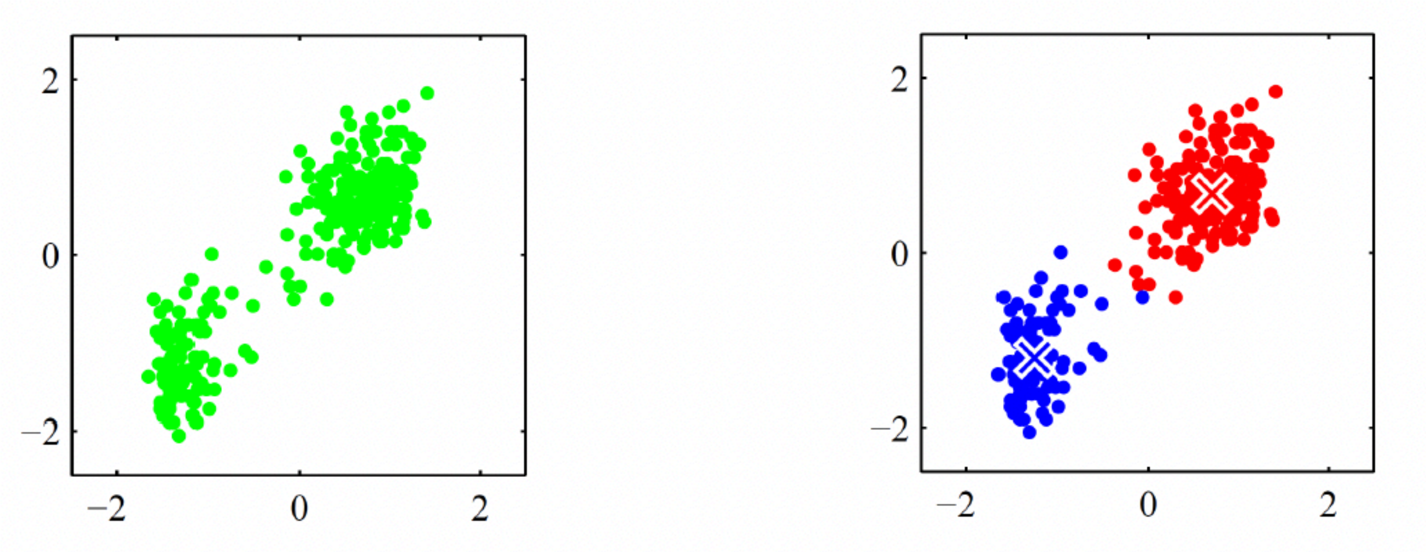
Output: group the data into some clusters, which means

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

Given: data points $x_1, \cdots, x_n \in \mathbb{R}^d$ and clusters k we want.

Output: group the data into k clusters, which means,

- find assignment $\gamma_{ij} \in \{0, 1\}$ for each data point $i \in [n]$ and $j \in [k]$ s.t. $\sum_{j\in[k]} \gamma_{ij} = 1$ for any fixed i . each datapoint is assigned to exactly 1 cluster.
- find the cluster centers $\mu_1, \cdots, \mu_k \in \mathbb{R}^d$.



Clustering is one of the most fundamental ML tasks, with many applications:

- recognize communities in a social network
- group similar customers in market research
- image segmentation
- accelerate other algorithms (e.g. nearest neighbor classification)

Formal Objective

As with PCA, no ground-truth to even measure the quality of the answer (no labels given).

What is the high-level goal here?

We want to partition the points into k clusters, such that points within each cluster are close to their cluster center.

We can turn this into an optimization problem, find γ_{ij} and μ_j to minimize

$$F(\gamma_{ij}, \mu_j) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \mu_j||_2^2$$

i.e. the sum of squared distances of each point to its center. This is the "k-means" objective.

Alternating minimization

Unfortunately, finding the exact minimizer of the k-means objective is NP-hard (we don't expect the problem to be exactly solvable efficiently and polynomial time)!

Therefore, we use a heuristic (alternating minimization) that alternatively minimizes over γ_{ij} and μ_j :

Initialize: $\mu_j^{(1)} : j \in [k]$

For $t = 1, 2, \cdots$

- find

  $$\gamma_{ij}^{(t+1)} = \arg\min_{\gamma_{ij}} F(\gamma_{ij}, \mu_j^{(t)})$$

  this means fix μ_{ij} , find γ_j .
- find

  $$\mu_j^{(t+1)} = \arg\min_{\mu_j} F(\gamma_{ij}^{(t+1)}, \mu_j)$$

  this means fix γ_{ij} , find μ_j .

Alternating minimization: Closer look

The first step:

$$\min_{\gamma_{ij}} F(\gamma_{ij}, \mu_j) = \min_{\gamma_{ij}} \sum_i \sum_j \gamma_{ij} ||x_i - \mu_j||_2^2$$
$$= \sum_i \min_{\gamma_{ij}} \sum_j \gamma_{ij} ||x_i - \mu_j||_2^2$$

is simply to assign each x_i to the closest μ_j , i.e.

$$\gamma_{ij} = \mathbb{I}[j == \arg\min_{c \in [k]} ||x_i - \mu_c||_2^2]$$

for all $j \in [k]$ and $i \in [n]$. This means $\begin{cases} 1, & if\ j\ is\ minimizes \\ 0, & otherwise \end{cases}$.

The second step

$$\min_{\mu j} F(\gamma_{ij}, \mu_j) = \min_{\gamma_{ij}} \sum_i \sum_j \gamma_{ij} ||x_i - \mu_j||_2^2$$
$$= \sum_j \min_{\mu_j} \sum_{i:\gamma_{ij}=1} ||x_i - \mu_j||_2^2$$

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

$$\mu_j = \frac{\sum_{i:\gamma_{ij}=1} x_i}{|i : \gamma_{ij} = 1|} = \frac{\sum_i \gamma_{ij} x_i}{\sum_i \gamma_{ij}}$$

for each $j \in [k]$. This is vectorized equation. verify: take gradients!

k-means algorithm

step 0: Initialize $\mu_1, \cdots, \mu_k$

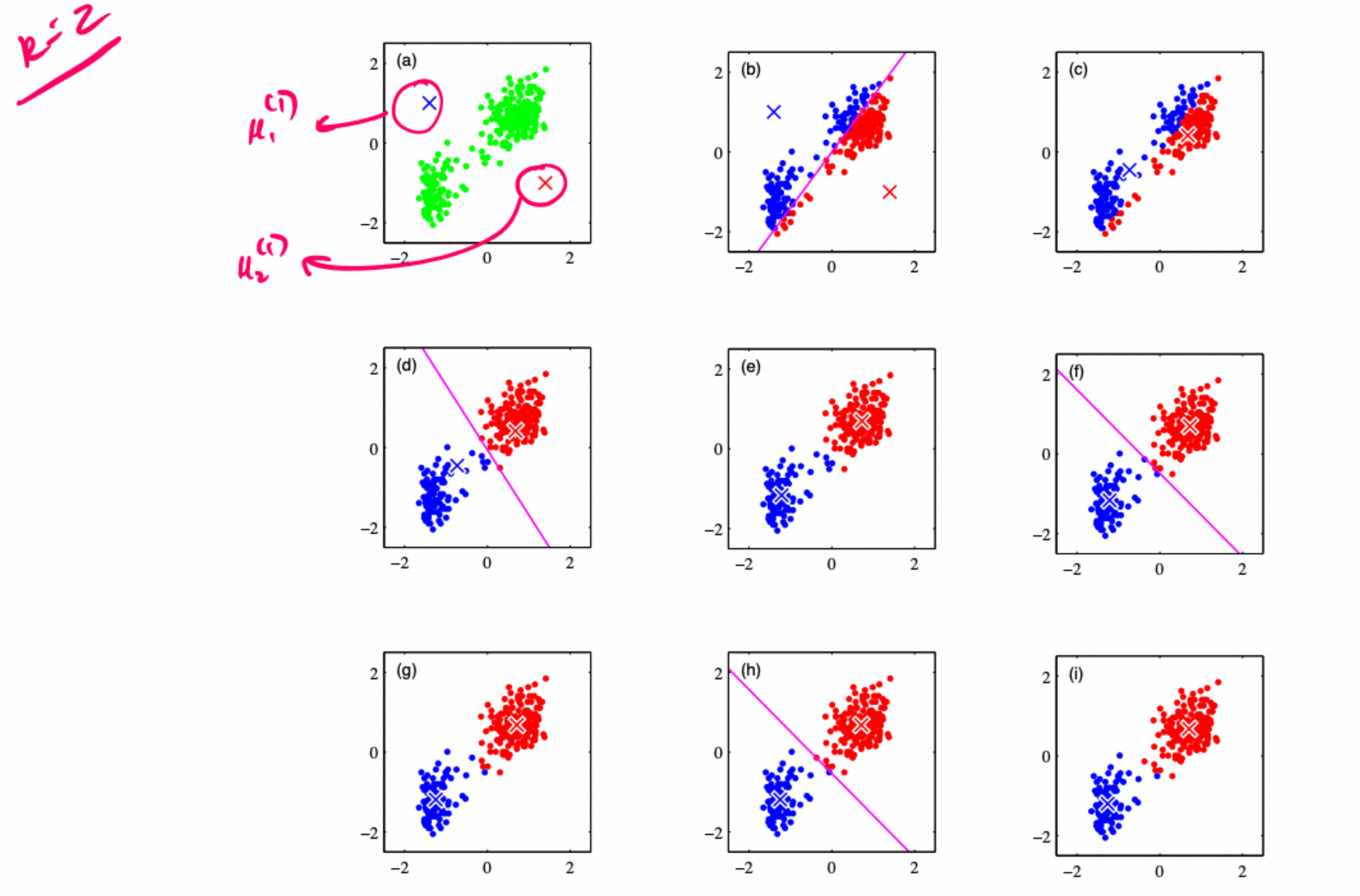
step 1: For the centers $\mu_1, \cdots, \mu_k$ being fixed, assign each point to the closest center:

$$\gamma_{ij} = \mathbb{I}[j == \arg\min_{c \in [k]} ||x_i - \mu_c||_2^2]$$

step 2: For the assignments γ_{ij} being fixed, update the centers

$$\mu_j = \frac{\sum_i \gamma_{ij} x_i}{\sum_i \gamma_{ij}}$$

step 3: Return to Step 1 if not converged (convergence means that all the assignments γ_{ij} are unchanged in Step 1).



k-means algorithm: convergence

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

1.objective strictly decreases at each step if the algorithm has not converged.

Why? For $t = 1, 2, \cdots$

- find

$$\gamma_{ij}^{(t+1)} = \arg\min_{\gamma_{ij}} F(\gamma_{ij}, \mu_j^{(t)})$$
$$= \mathbb{I}[j == \arg\min_{c \in [k]} ||x_i - \mu_c||_2^2]$$

  this step will never increase objective function value. (as long as there are no ties, then it decreases function value)

- find

$$\mu_j^{(t+1)} = \arg\min_{\mu_j} F(\gamma_{ij}^{(t+1)}, \mu_j)$$

  this step means if the assignments changed in the previous step, then this reduces its function value (mean is unique minimizer of sum of squares objective)

2.#possible assignments are finite (k^n, exponentially large though, k is the number of possible assignments to each point)

Therefore, the algorithm must converge in at most k^n steps.

Why? More specifically, why can't the algorithm cycle between different clusterings?

- Suppose the algorithm finds the same clustering at time steps t_1 and t_2 .
- Since the objective function value decreases at every step, this means the same clustering (at time steps t_1 and t_2) has two different costs, which is not possible.

- Therefore, by contradiction, the algorithm cannot cycle between clusterings.

However,

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

k-means algorithm: how to initialize?

There are different ways to initialize:

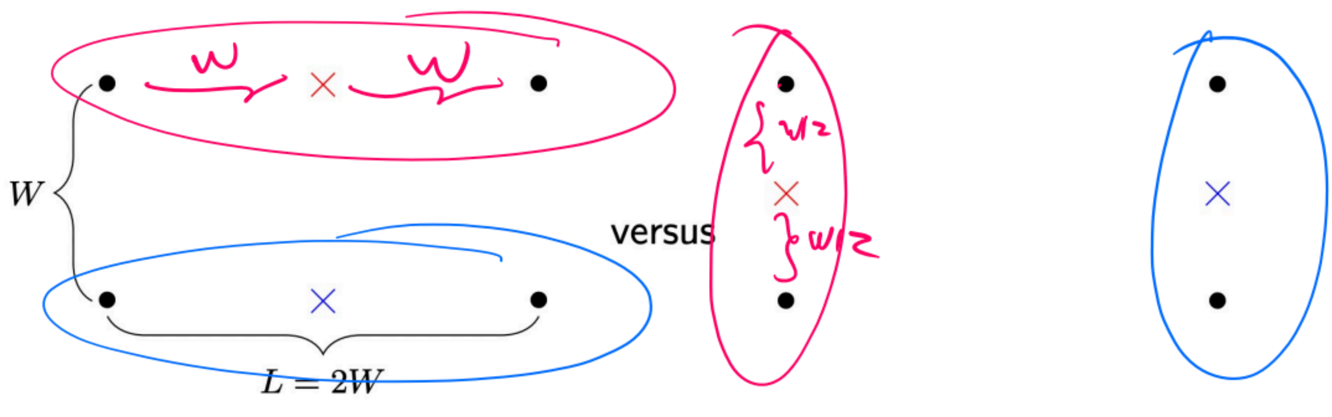
- randomly pick k points as initial centers $\mu_1, \cdots, \mu_k$
- or randomly assign each point to a cluster, then average to find centers
- or more sophisticated approaches (e.g. k-means++)

Initialization matters for convergence.

k-means++ have different initialization: Assuming that n initial cluster centers have been selected, when selecting the $n+1$-th cluster center: we caculate the distance from every point to the n cluster center, and normalize it to probability. The more distant points from the current n cluster centers will have a higher probability of being selected as the $n+1$ cluster center.

Local vs Global minima

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.

As we increase L , we can make the local minima arbitrarily bad. $\therefore$ Initialization matters a lot to convergence.

Summary

- Clustering is a fundamental unsupervised learning task.
- k-means is a alternating minimization algorithm for the k-means objective.
- The algorithm always converges, but it can converge to a local minimum.
- Initialization matters a lot for the convergence. There are principled initialization schemes, which have guarantees on the solution they find (e.g. k-means++).

Gaussian Mixture Model and EM Algorithm

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 models: the Expectation Maximization (EM) algorithm.

For classification, we discussed the sigmoid model to “explain” how the labels are generated($\ln[y|x, w] = \sigma(yw^T x)$).

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

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

Why do generative modeling?

- can generate data from p
- can estimate probability of seeing any datapoint (useful for many tasks, such as for finding outliers/anomalies in data)

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:

$$p(x) = \sum_{j=1}^{k} \pi_j N(x|\mu_j, \Sigma_j)$$

where

- k : the number of Gaussian components (same as #clusters we want)
- $\pi_1, \cdots, \pi_k$: mixture weights, a distribution over k components. It means the probability of picking Gaussian j . and π_j need to meet $\sum_j \pi_j = 1$.
- μ_j and Σ_j : mean and covariance matrix of the k-th Gaussian
- N: the density function for a Gaussian, means then we sample datapoints from Gaussian.

Another view:

by introducing a latent(unobserved) variable $z \in [k]$, which indicates cluster membership, we can see p as a marginal distribution

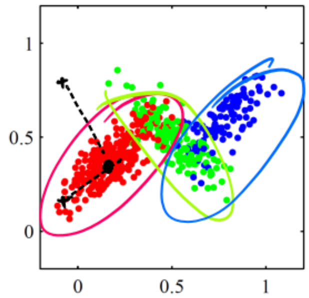
$$p(x) = \sum_{j=1}^{k} p(x, z=j) = \sum_{j=1}^{k} p(z=j)p(x|z=j) = \sum_{j=1}^{k} \pi_j N(x|\mu_j, \Sigma_j)$$

x and z are both random variables drawn from the model: x is observed; z is unobserved/latent.

An example:

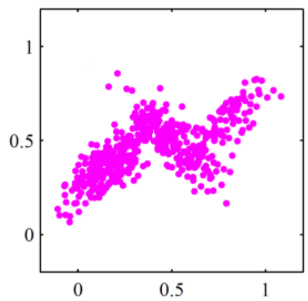
GMMs are better at handling scaling

An example



The conditional distributions are

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



The marginal distribution is

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

Learning GMMs

Learning a GMM means finding all the parameters $\theta = \{\pi_j, \mu_j, \Sigma_j\}_{j=1}^k$.

In the process, we will learn the distribution of the latent variable z_i as well:

$$p(z_i = j|x_i) := \gamma_{ij} \in [0, 1]$$

i.e. "soft assignment" of each point to each cluster, as opposed to “hard assignment” by k-means (all $\gamma_{ij} = \{0, 1\}$).

GMM is more explanatory than k-means

- both learn the cluster centers μ_j ’s.
- in addition, GMM learns cluster weight π_j and covariance Σ_j , thus
  - we can predict probability of seeing a new point
  - we can generate synthetic data

As always, we want to do maximum-likelihood estimation (MLE): use log-likelihood of data, to find

$$\arg\max_\theta \ln \prod_{i=1}^n p(x_i; \theta) = \arg\max_\theta \sum_{i=1}^n \ln p(x_i; \theta) := \arg\max_\theta P(\theta)$$

This is called incomplete log-likelihood (since z_i ’s are unobserved). We can still write it down as an optimization problem by marginalizing out the z_i ’s.

$$\begin{aligned} P(\theta) &= \sum_{i=1}^n \ln p(x_i; \theta) = \sum_{i=1}^n \ln \left(\sum_{j=1}^k p(x_i, z_i = j; \theta) \right) \\ &= \sum_{i=1}^n \ln \left(\sum_{j=1}^k p(z_i = j; \theta) p(x_i | z_i = j; \theta) \right) = \sum_{i=1}^n \ln \left(\sum_{j=1}^k \pi_j N(x_i | \mu_j, \Sigma_j) \right) \end{aligned}$$

This is a non-concave problem, and does not have a closed-form solution.

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 π_j, μ_j, Σ_j for each $j \in [k]$.

step 1: (E-Step) update the “soft assignment” (fixing parameters), priors $\times$ likelihood:

$$\gamma_{ij} = p(z_i = j | x_i) \propto \pi_j N(x_i | \mu_j, \Sigma_j)$$

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

$$\pi_j = \frac{\sum_i \gamma_{ij}}{n} \qquad \mu_j = \frac{\sum_i \gamma_{ij} x_i}{\sum_i \gamma_{ij}}$$
$$\Sigma_j = \frac{1}{\sum_i \gamma_{ij}} \sum_i \gamma_{ij} (x_i - \mu_i)(x_i - \mu_j)^T$$

step 3: return to Step 1 if not converged.

EM Algorithm

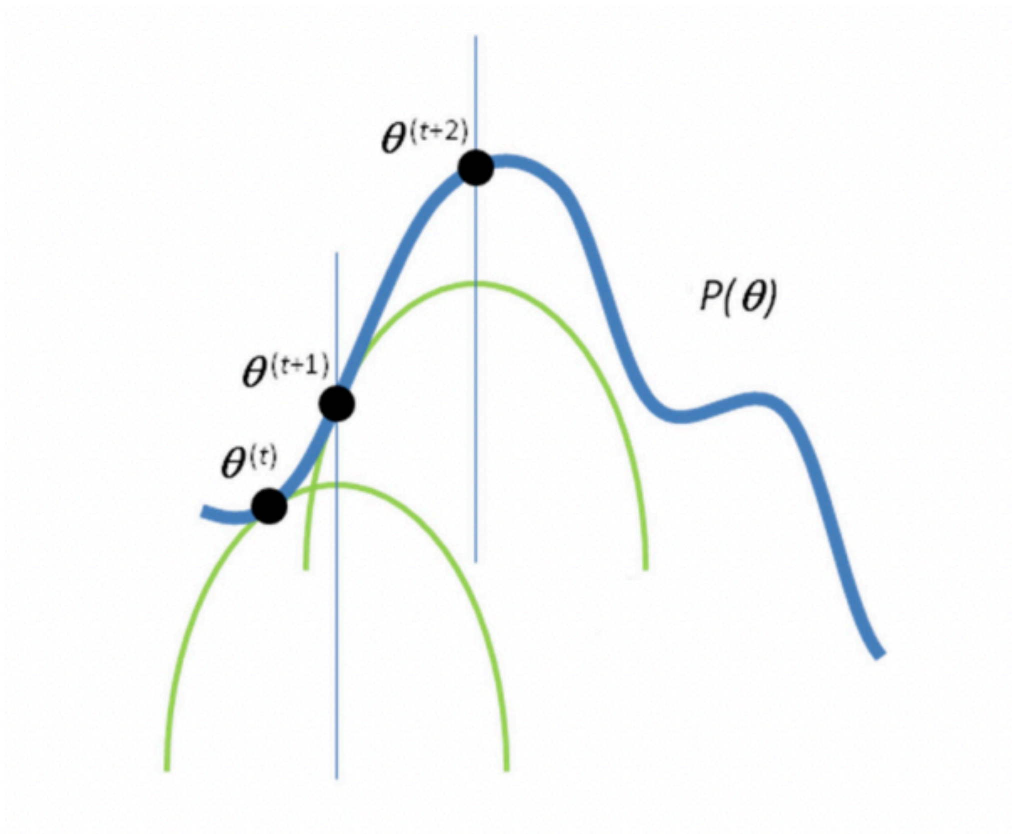
In general, EM is a heuristic to solve MLE with latent variables (not just GMM), i.e. find the maximizer of

$$P(\theta) = \sum_{i=1}^{n} \ln p(x_i; \theta) = \sum_{i=1}^{n} \ln \int_{z_i} p(x_i, z_i; \theta) \mathrm{d} z_i$$

- $\theta = \{\mu_j, \Sigma_j, \pi_j\}$ is the parameters for a general probabilistic model.
- x_i ’s are observed random variables.
- z_i ’s are latent variables. If continuous, integral z_i , otherwise sum z_i .

Again, directly solving the objective is usually complicated and does not have a closed form solution.

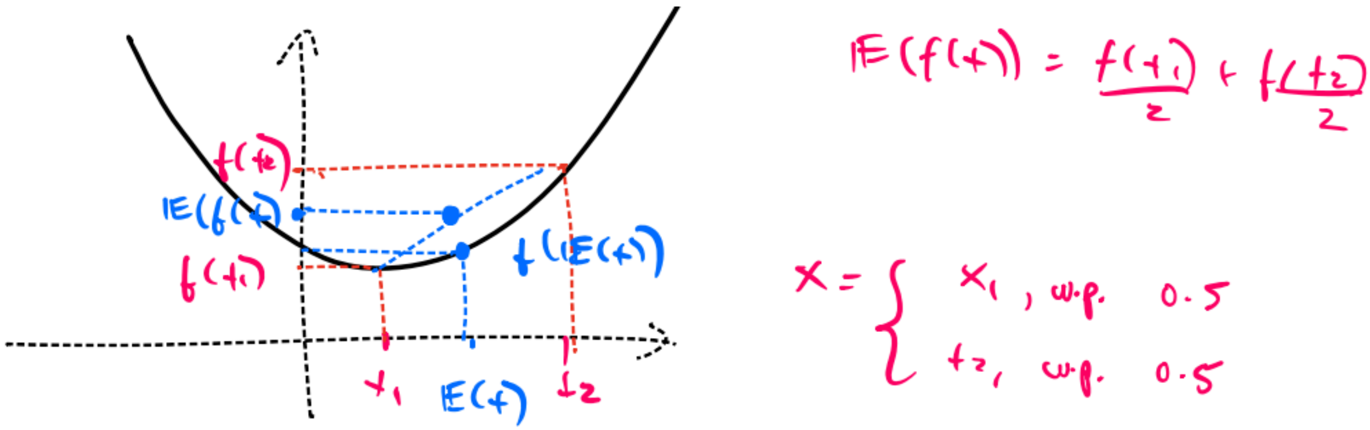
High-level idea: Keep maximizing a lower bound of P that is more manageable.



Jensen's inequality

For any x and convex function $f(x)$, $f(E(x)) \leqslant E(f(x))$

e.g. $f(x) = x^2$, then $(E(x))^2 \leqslant E(x^2)$. This is correct since $Var(x) = E(x^2) - (E(x))^2 \geqslant 0$.



Equal Condition: function f if $f(x)$ is strictly convex ($f''(x) > 0, \forall x$), then $f(E(x)) = E(f(x)) \Rightarrow$

x is a constant($x = c$ for some c , x always $= \{x_1, x_2\}$ for the exp above)

A lower bound on the log likelihood

Introducing P , and finding the lower bound of P

$$P(\theta) = \sum_{i=1}^{n} \ln p(x_i; \theta)$$

$$\begin{aligned}
\ln p(x; \theta) &= \ln \left(\sum_{z=1}^{k} p(x, z = j; \theta) \right) \\
&= \ln \left(\sum_{z=1}^{k} q(z) \frac{p(x, z; \theta)}{q(z)} \right) \qquad true\ for\ any\ q(z) \neq 0\ (we\ also\ impose\ \sum_{z=1}^{k} q(z) = 1) \\
&= \ln \left(\sum_{z=1}^{k} \mathbb{E}_{z \sim q(z)} \frac{p(x, z; \theta)}{q(z)} \right) \qquad \rightarrow \mathbb{E}_{z \sim q(z)}(f(z)) = \sum_{z} q(z) f(z) \\
&\geqslant \mathbb{E}_{z \sim q(z)} \left[\ln \left(\frac{p(x, z; \theta)}{q(z)} \right) \right] \qquad oppsite\ of\ Jensen (since\ ln(\cdot)\ is\ concave)
\end{aligned}$$

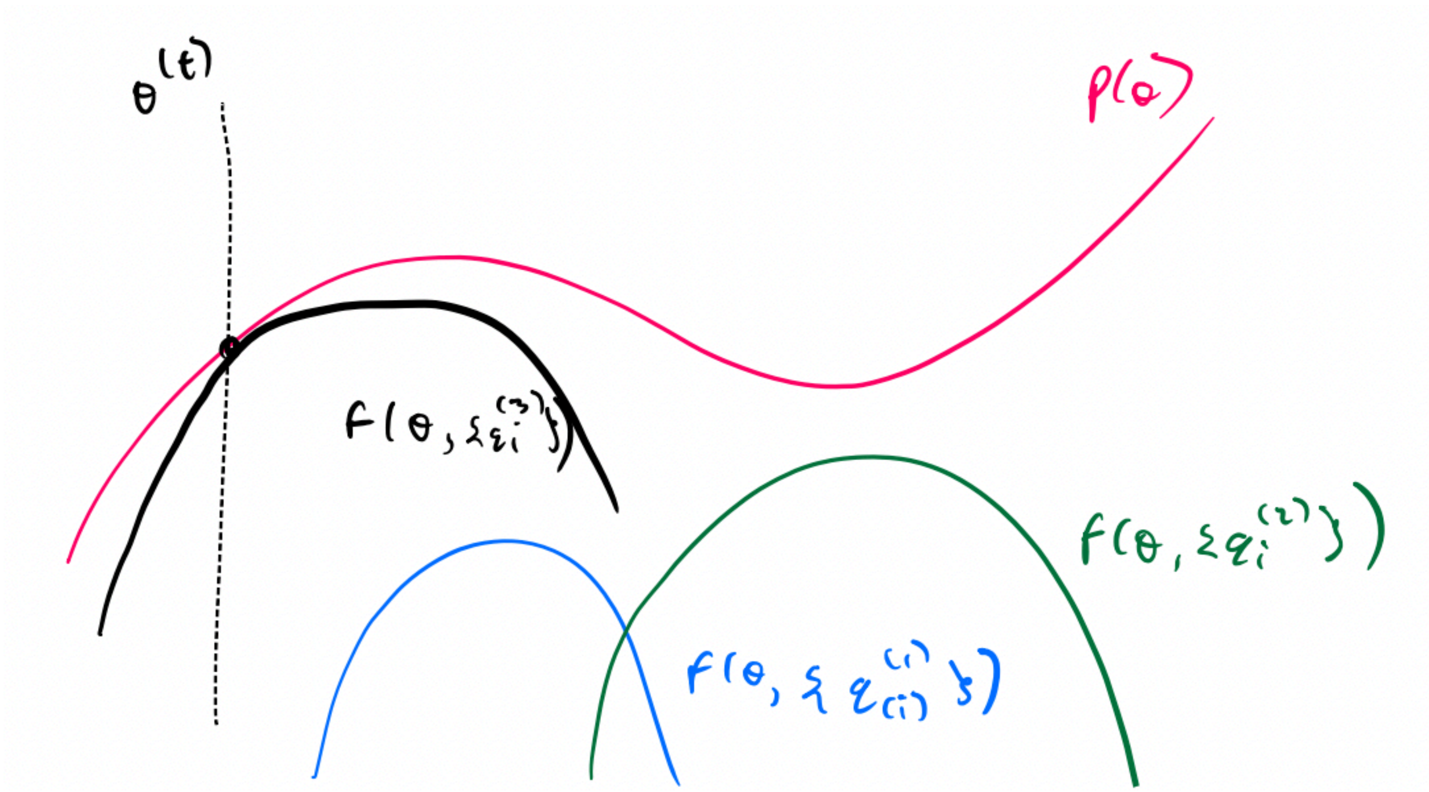
Therefore, our log-likelihood can be written as

$$\begin{aligned}
P(\theta) &= \sum_{i=1}^{n} \ln p(x_i; \theta) \geqslant \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q(z_i)} \left[\ln \left(\frac{p(x_i, z_i; \theta)}{q_i(z_i)} \right) \right] \\
&= F(\theta, \{q_i\}_{i=1}^{n})
\end{aligned}$$

where $ln \left(\frac{p(x_i, z_i; \theta)}{q_i(z_i)} \right)$ is the lower bound for any $\{q_i\}_{i=1}^{n}$.

Alternatively maximizing the lower bound

The expression for the likelihood holds for any $\{q_i\}$, so how do we choose? If we have some guess of the parameters θ, we should choose $\{q_i\}$ to try to make the lower bound tight at that value of θ, i.e. make the inequality hold with equality at that value of θ .



Equivalently, this is the same as alternatingly maximizing F over $\{q_i\}$ and θ (similar to k-means).

Maximizing over q_i : Suppose we fix $\theta^{(t)}$, what should we choose $q_i^{(t)}$?

The inequality arises from the step where we used Jensen's inequality. How do we get this step to hold with equality ($\sum_{i=1}^n \ln p(x_i;\theta) = \sum_{i=1}^n \mathbb{E}_{z_i\sim q(z_i)}\left[\ln\left(\frac{p(x_i,z_i;\theta)}{q_i(z_i)}\right)\right]$)?

The function should be a constant function, i.e.

$$\frac{p(x_i,z_i;\theta)}{q_i(z_i)} = c_i$$

for some constant c_i which does not depend on the value taken by the random variable z_i .

since $\sum_{z_i=1}^k q_i^{(t)}(z_i) = 1$, we get,

$$c_i = \sum_{z_i=1}^k p(x_i,z_i;\theta)$$

Therefore:

$$\begin{aligned} q_i^{(t)}(z_i) &= \frac{p(x_i,z_i;\theta^{(t)})}{\sum_{z_i=1}^k p(x_i,z_i;\theta^{(t)})} \\ &= \frac{p(x_i,z_i;\theta^{(t)})}{p(x_i;\theta)} \\ &= p(z_i|x_i;\theta^{(t)}) \end{aligned}$$

i.e., the posterior distribution of z_i given x_i and $\theta^{(t)}$.

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

- $F(\theta, q_i^{(t)}) \leqslant P(\theta)$ for all θ
- $F(\theta^{(t)}, q_i^{(t)}) = P(\theta^{(t)})$

Maximizing over θ : Fix $q_i^{(t)}$, maximize over θ

$$
\begin{aligned}
&\arg\max_{\theta} F(\theta, q_i^{(t)}) \\
&\quad = \arg\max_{\theta} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln \left(\frac{p(x_i, z_i; \theta)}{q_i^{(t)}(z_i)} \right) \right] \\
&\quad = \arg\max_{\theta} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(x_i, z_i; \theta) \right] - \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln(q_i^{(t)}(z_i)) \right] \qquad (\mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln(q_i^{(t)}(z_i)) \right] \textit{ does not depend on } \theta, \textit{we've fixed } q_i^{(t)}) \\
&\quad = \arg\max_{\theta} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(x_i, z_i; \theta) \right] \\
&\quad := \arg\max_{\theta} Q(\theta; \theta^{(t)})
\end{aligned}
$$

Q is the (expected) complete likelihood and is usually more tractable. $\theta^{(t)}$ is what we get , θ is what we need to find.

- $Q(\theta; \theta^{(t)})$ versus the incomplete likelihood: $P(\theta) = \sum_{i=1}^{n} \ln p(x_i; \theta)$.

General EM algorithm

step 0: Initialize $\theta^{(1)}, t = 1$.

step 1: (E-Step) update the posterior of latent variables z_i :

$$q_i^{(t)}(z_i) = p(z_i | x_i; \theta^{(t)})$$

and obtain Expectation of complete likelihood:

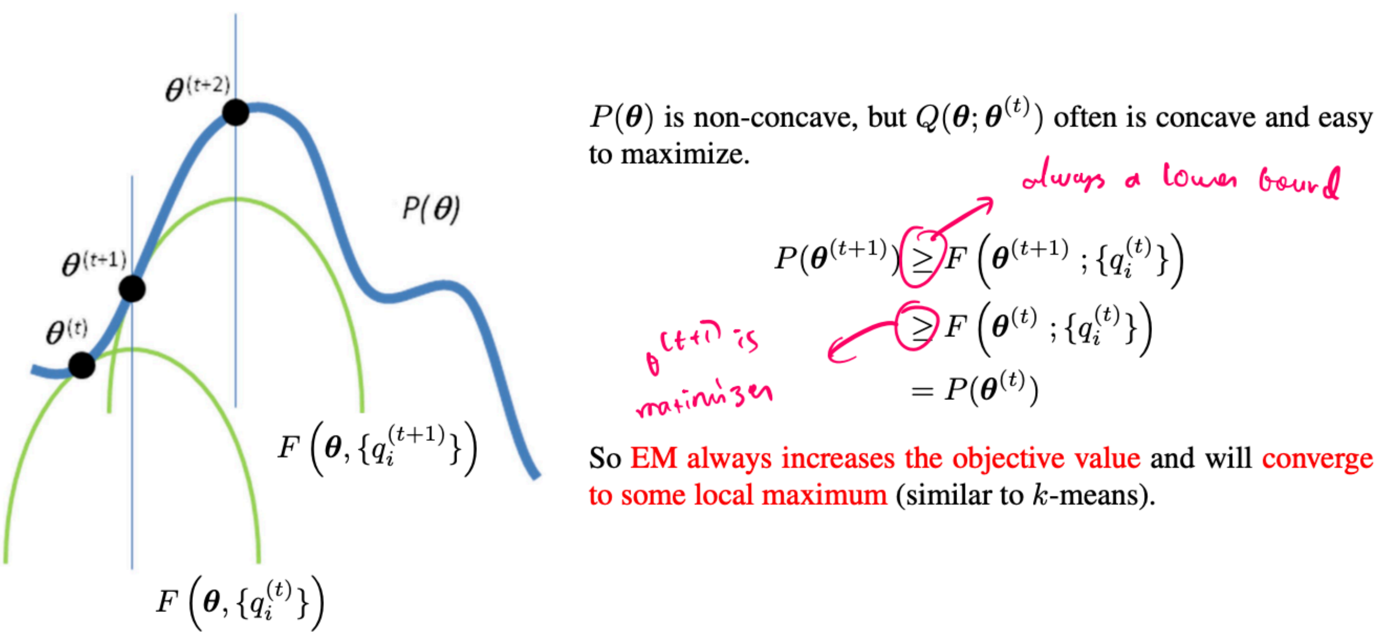
$$Q(\theta; \theta^{(t)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(x_i, z_i; \theta) \right]$$

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

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} Q(\theta; \theta^{(t)})$$

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

Pictorial explanation:



Applying EM to learn GMMs

E-Step:

$$
\begin{aligned}
q_i^{(t)}(z_i = j) &= p(z_i = j | x_i; \theta^{(t)}) \\
&= \frac{p(x_i, z_i = j; \theta^{(t)})}{p(x_i; \theta^{(t)})} \qquad p(x_i; \theta^{(t)}) \; not \; depend \; on \; j \\
&\propto p(x_i, z_i = j; \theta^{(t)}) \\
&= p(z_i = j; \theta^{(t)}) p(x_i | z_i = j; \theta^{(t)}) \\
&= \pi_j^{(t)} N(x_i | \mu_j^{(t)}, \Sigma_j^{(t)})
\end{aligned}
$$

This computes the "soft assignment" $\gamma_{ij} = q_i^{(t)}(z_i = j)$, i.e. conditional probability of x_i belonging to cluster j .

M-Step:

$$
\begin{aligned}
\arg\max_\theta Q(\theta; \theta^{(t)}) &= \arg\max_\theta \sum_{i=1}^n \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(x_i, z_i; \theta) \right] \\
&= \arg\max_\theta \sum_{i=1}^n \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(z_i; \theta) + \ln p(x_i | z_i; \theta) \right] \\
&= \arg \max_{\{\pi_j, \mu_j, \Sigma_j\}} \sum_{i=1}^n \sum_{j=1}^k \gamma_{ij} \left(\ln \pi_j + \ln N(x_i | \mu_j, \Sigma_j) \right)
\end{aligned}
$$

$\gamma_{ij} \ln \pi_j$ only depends on π_j , $\gamma_{ij} \ln N(x_i | \mu_j, \Sigma_j)$ only depends on μ_j, Σ_j .

To find $\pi_1, \cdots, \pi_k$, solve

$$
\arg\max_\pi \sum_{i=1}^n \sum_{j=1}^k \gamma_{ij} \ln \pi_j
$$

To find each μ_j, Σ_j , solve

$$
\arg\max_{\mu_j, \Sigma_j} \sum_{i=1}^n \gamma_{ij} \ln N(x_i | \mu_j, \Sigma_j)
$$

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

$$
\pi_j = \frac{\sum_i \gamma_{ij}}{n}
$$

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

$$
\mu_j = \frac{\sum_i \gamma_{ij} x_i}{\sum_i \gamma_{ij}}
$$

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

$$
\Sigma_j = \frac{1}{\sum_i \gamma_{ij}} \sum_i \gamma_{ij} (x_i - \mu_j)(x_i - \mu_j)^T
$$

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

Putting it together: EM for learning GMMs

step 0: Initialize π_j, μ_j, Σ_j for each $j \in [k]$.

step 1: (E-Step) update the "soft assignment" (fixing parameters):

$$
\gamma_{ij} = p(z_i = j | x_i) \propto \pi_j N(x_i | \mu_j, \Sigma_j)
$$

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

$$\pi_j = \frac{\sum_i \gamma_{ij}}{n} \quad \mu_j = \frac{\sum_i \gamma_{ij} x_i}{\sum_i \gamma_{ij}} \quad \Sigma_j = \frac{1}{\sum_i \gamma_{ij}} \sum_i \gamma_{ij} (x_i - \mu_j)(x_i - \mu_j)^T$$

step 3: return to step 1 if not converged.