CSCI 567: Machine Learning

Vatsal Sharan
Fall 2022

Lecture 10, Nov 10

USC University of Southern California

Administrivia

- Start on your project if you haven’t already!
  - Make groups (of 4) by tomorrow (Nov 11), minimum team size is 3.
  - Top teams as of Nov 16 can get a bonus!

- HW4 is due in about one weeks (Nov 16 at 2pm).
  - We’ll release another question on Gaussian mixture models tomorrow.

- Today’s plan:
    - Clustering
    - Gaussian Mixture Models and Expectation Maximization (EM)
    - In the discussion, we will go over popular evaluation metrics for supervised learning

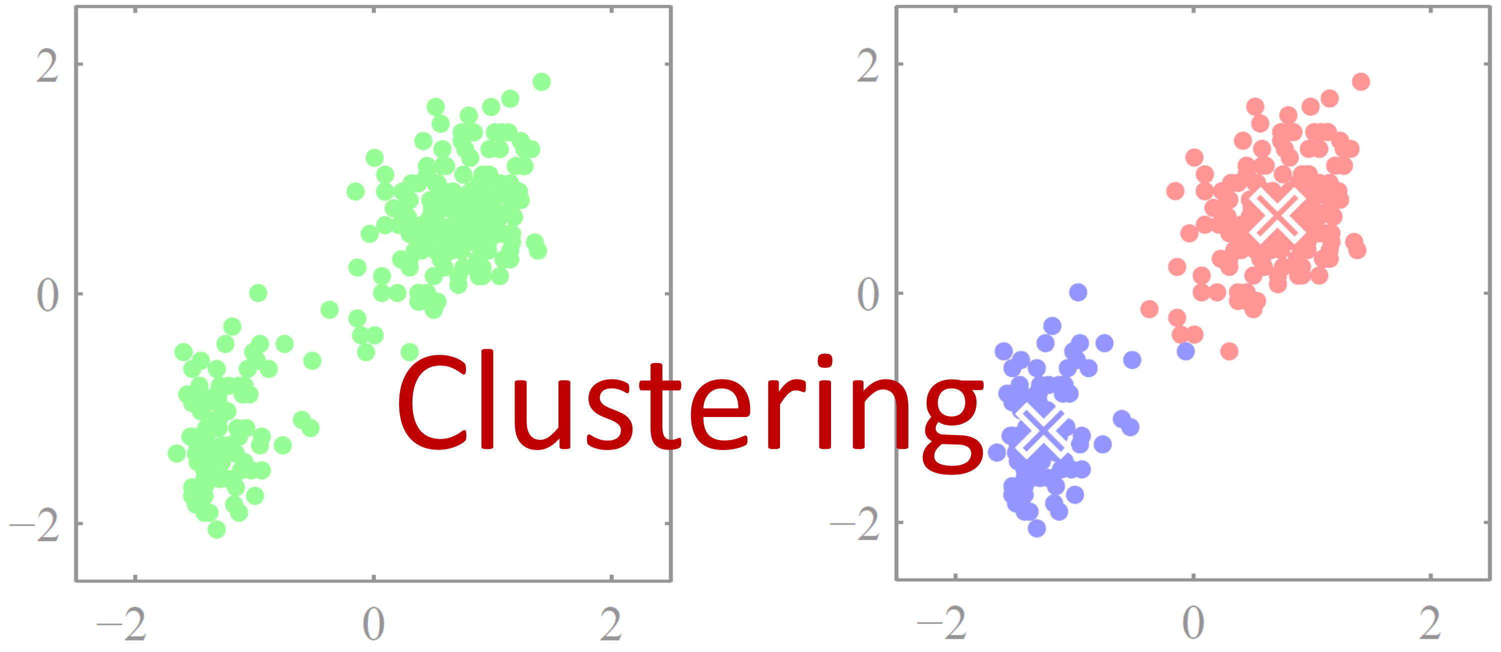
A simplistic taxonomy of ML

Supervised learning:
Aim to predict outputs of future datapoints

Unsupervised learning:
Aim to discover hidden patterns and explore data

Reinforcement learning:
Aim to make sequential decisions

Clustering



Clustering

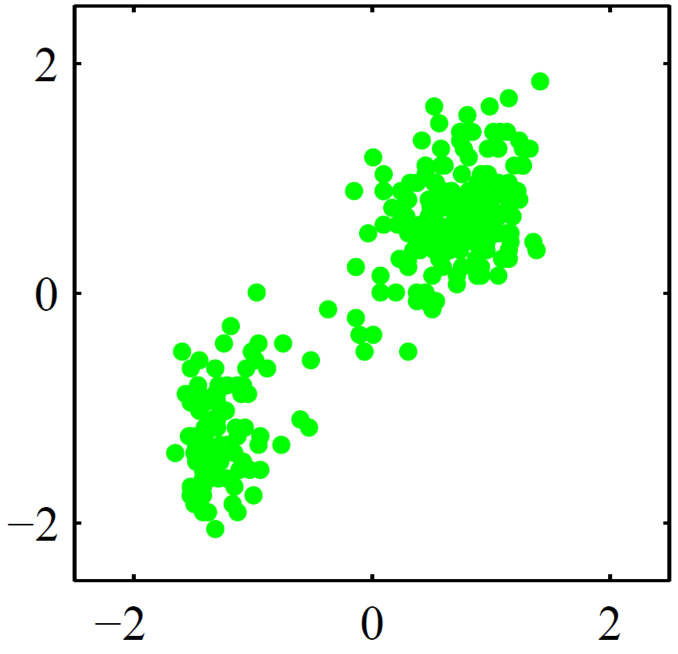
- Introduction
- Formalizing and solving the objective (alternating minimization)
- k-means algorithm

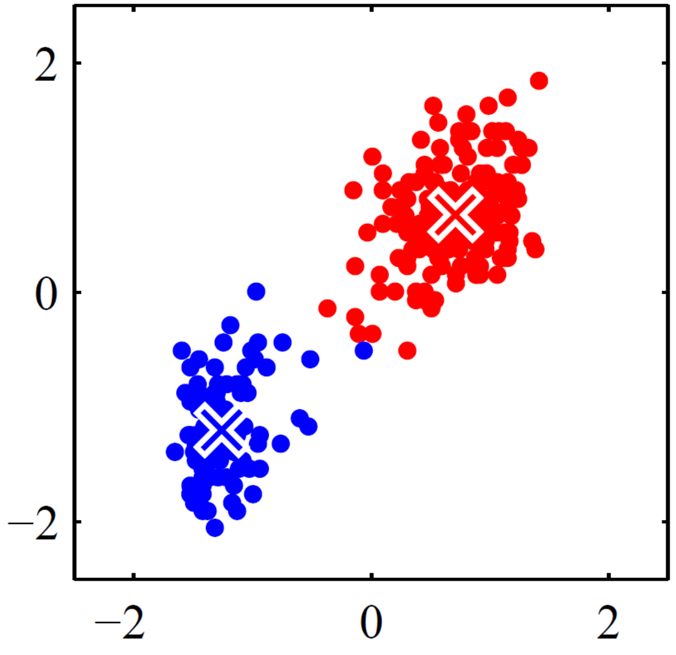
Clustering: Informal definition

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



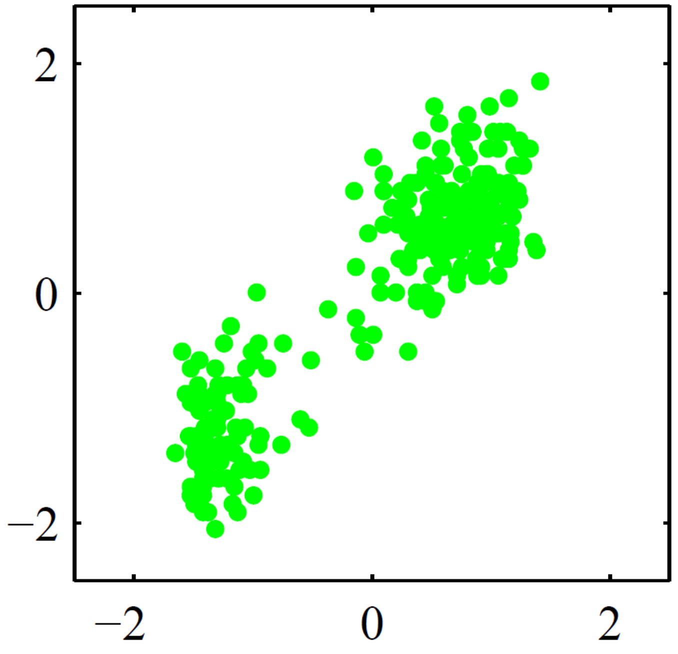


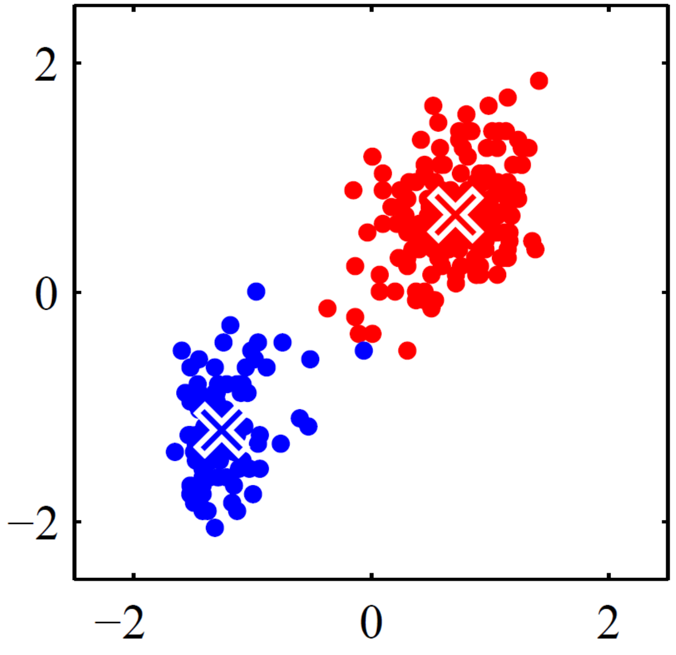
Clustering: More formal definition

Given: data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ 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
- find the cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$





Many applications

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)
- . . .

Clustering

- Introduction
- Formalizing and solving the objective (alternating minimization)
- k-means algorithm

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 $\boldsymbol{\mu}_j$ to minimize

$$F\left(\{\gamma_{ij}\}, \{\boldsymbol{\mu}_j\}\right) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2$$

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

How to solve this? Alternating minimization

Unfortunately, finding the exact minimizer of the k-means objective is *NP-hard!*

Therefore, we use a heuristic (*alternating minimization*) that alternatingly minimizes over $\{\gamma_{ij}\}$ and $\{\boldsymbol{\mu}_j\}$:

Initialize $\{\boldsymbol{\mu}_j^{(1)} : j \in [k]\}$

For $t = 1, 2, \ldots$

- find

$$\{\gamma_{ij}^{(t+1)}\} = \operatorname*{argmin}_{\{\gamma_{ij}\}} F\left(\{\gamma_{ij}\}, \{\boldsymbol{\mu}_j^{(t)}\}\right)$$

- find

$$\{\boldsymbol{\mu}_j^{(t+1)}\} = \operatorname*{argmin}_{\{\boldsymbol{\mu}_j\}} F\left(\{\gamma_{ij}^{(t+1)}\}, \{\boldsymbol{\mu}_j\}\right)$$

Alternating minimization: Closer look

The first step

$$\begin{aligned}\min_{\{\gamma_{ij}\}} F\left(\{\gamma_{ij}\}, \{\boldsymbol{\mu}_j\}\right) &= \min_{\{\gamma_{ij}\}} \sum_i \sum_j \gamma_{ij} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2 \\ &= \sum_i \min_{\{\gamma_{ij}\}} \sum_j \gamma_{ij} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2\end{aligned}$$

is simply to **assign each x_i to the closest $\boldsymbol{\mu}_j$**, i.e.

$$\gamma_{ij} = \mathbb{I}\left[j == \operatorname*{argmin}_c \|\boldsymbol{x}_i - \boldsymbol{\mu}_c\|_2^2 \right]$$

for all $j \in [k]$ and $i \in [n]$.

Alternating minimization: Closer look

The second step

$$\begin{aligned}\min_{\{\boldsymbol{\mu}_j\}} F\left(\{\gamma_{ij}\}, \{\boldsymbol{\mu}_j\}\right) &= \min_{\{\boldsymbol{\mu}_j\}} \sum_i \sum_j \gamma_{ij} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2 \\ &= \sum_j \min_{\boldsymbol{\mu}_j} \sum_{i:\gamma_{ij}=1} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2\end{aligned}$$

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

$$\boldsymbol{\mu}_j = \frac{\sum_{i:\gamma_{ij}=1} \boldsymbol{x}_i}{|\{i : \gamma_{ij} = 1\}|} = \frac{\sum_i \gamma_{ij} \boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

for each $j \in [k]$.

Clustering

- Introduction
- Formalizing and solving the objective (alternating minimization)
- k-means algorithm

k-means algorithm

Step 0 Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$

Step 1 For the centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ being fixed, assign each point to the closest center:

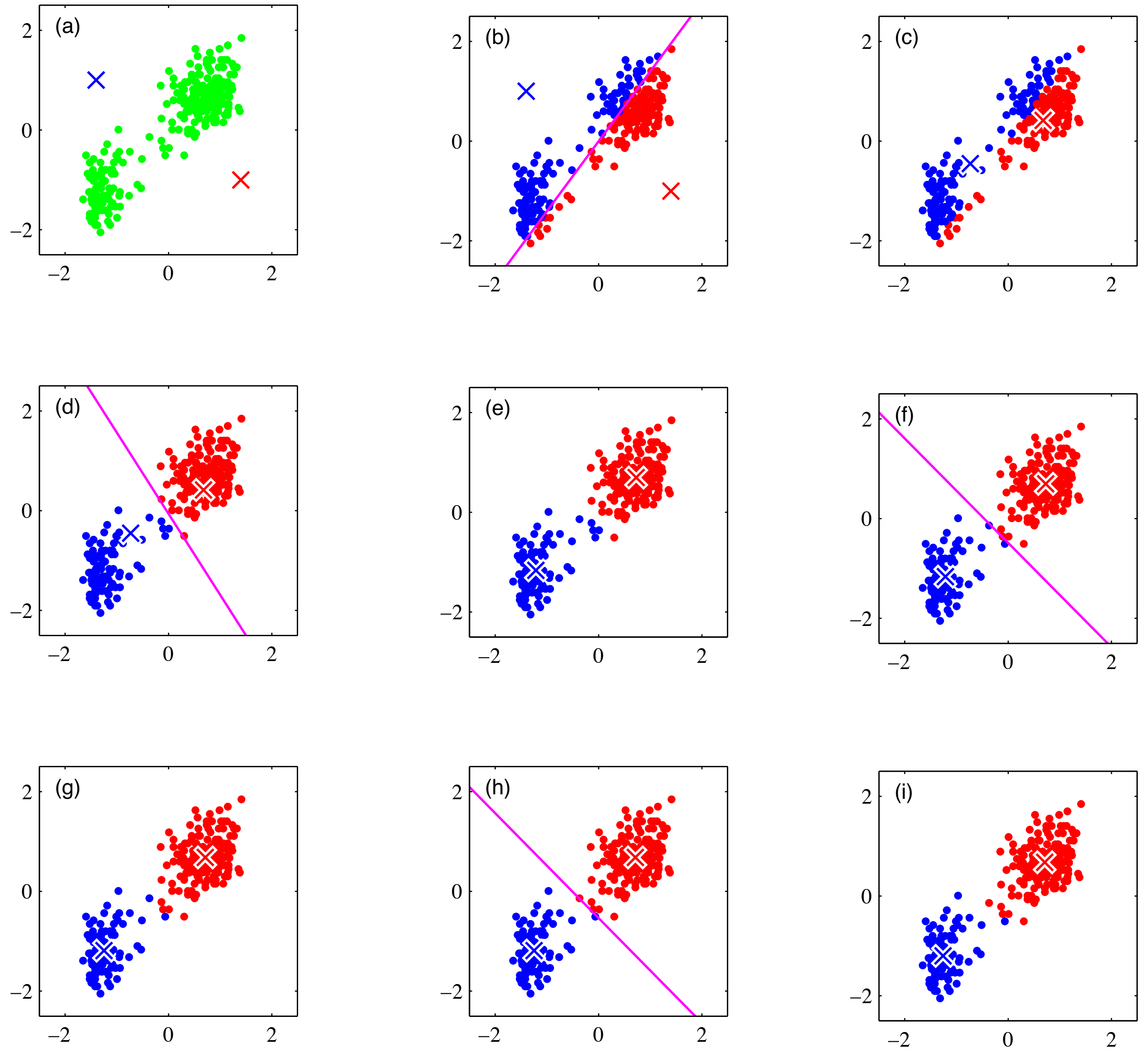
$$\gamma_{ij} = \mathbb{I}\left[j == \operatorname*{argmin}_c \|\boldsymbol{x}_i - \boldsymbol{\mu}_c\|_2^2\right]$$

Step 2 For the assignments $\{\gamma_{ij}\}$ being fixed, update the centers

$$\boldsymbol{\mu}_j = \frac{\sum_i \gamma_{ij}\boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

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

k-means algorithm: Example



k-means algorithm: Convergence

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

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

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

- find

$$\{\gamma_{ij}^{(t+1)}\} = \operatorname*{argmin}_{\{\gamma_{ij}\}} F\left(\{\gamma_{ij}\}, \{\boldsymbol{\mu}_j^{(t)}\}\right)$$
$$= \mathbb{I}\left[j == \operatorname*{argmin}_{c} \|\boldsymbol{x}_i - \boldsymbol{\mu}_c\|_2^2\right]$$

- find

$$\{\boldsymbol{\mu}_j^{(t+1)}\} = \operatorname*{argmin}_{\{\boldsymbol{\mu}_j\}} F\left(\{\gamma_{ij}^{(t+1)}\}, \{\boldsymbol{\mu}_j\}\right)$$
$$= \frac{\sum_i \gamma_{ij} \boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

k-means algorithm: Convergence

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

- objective strictly decreases at each step if the algorithm has not converged.
- #possible_assignments is finite (k^n, exponentially large though)

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.

k-means algorithm: Convergence

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

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

- #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

k-means algorithm: How to initialize?

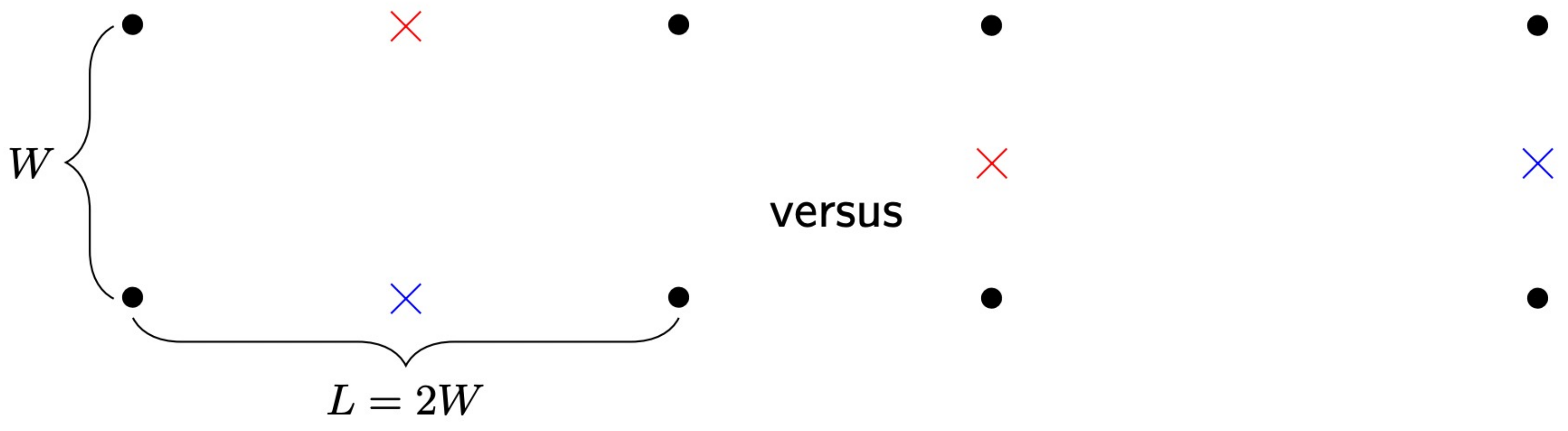
There are different ways to initialize:

- randomly pick k points as initial centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
- 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 algorithm: Local vs Global minima

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

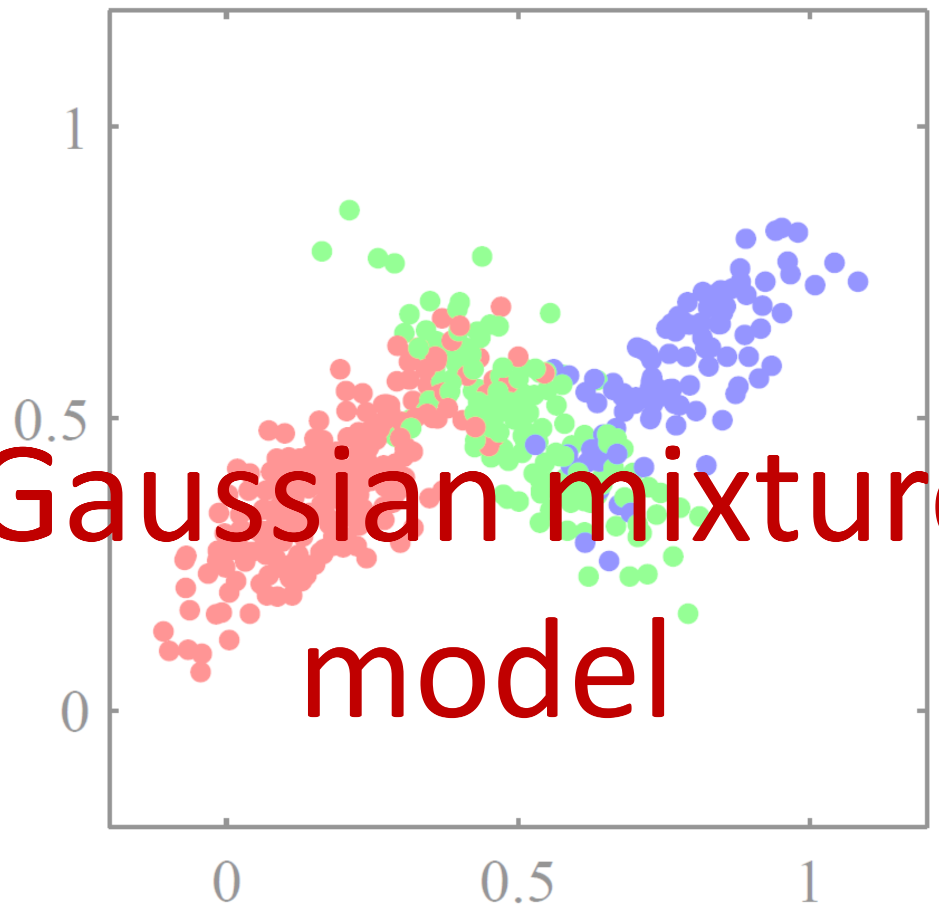


K-means converges immediately in both cases, but

- 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**.

k-means algorithm: 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

Gaussian Mixture Model

- Introduction
- Learning the parameters
- EM algorithm
- EM for the Gaussian Mixture Model

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 models:
the **Expectation Maximization (EM) algorithm**.

A generative model

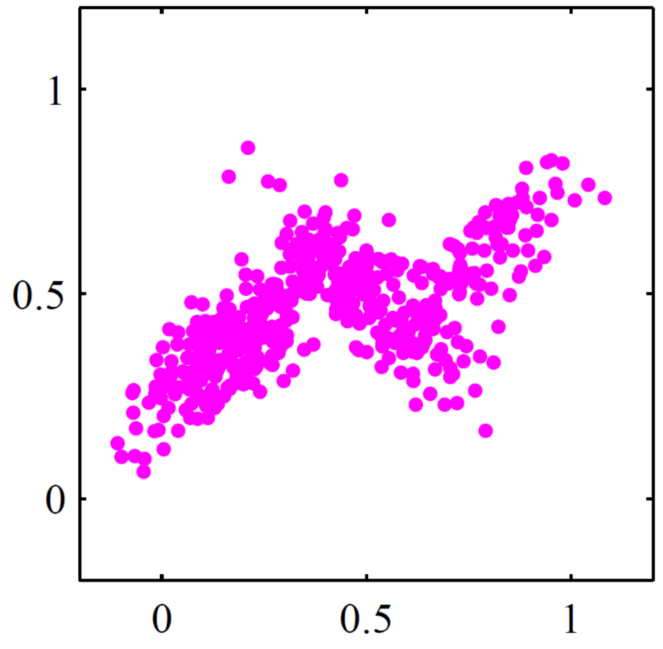
For classification, we discussed the sigmoid model to "explain" how the labels are generated.

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 $\boldsymbol{x} \sim p$.

Why do generative modelling?

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



What probabilistic model generates data like this?

GMM: Intuition

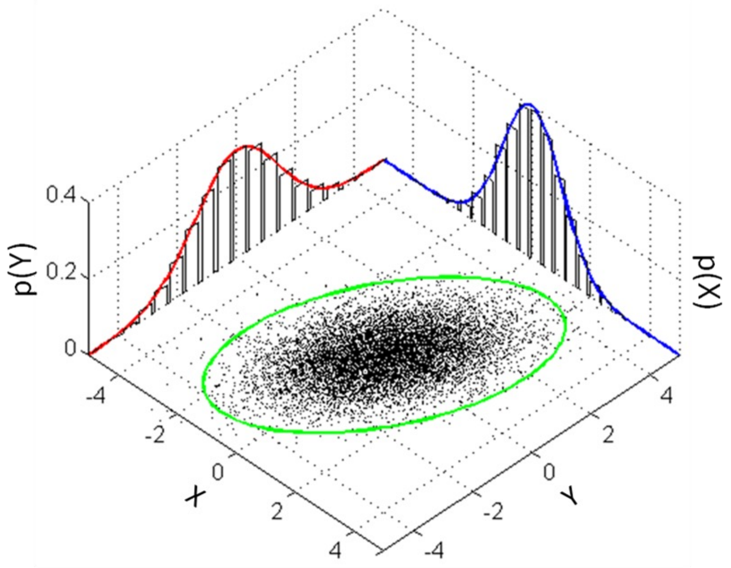
GMM is a natural model to explain such data.

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**".



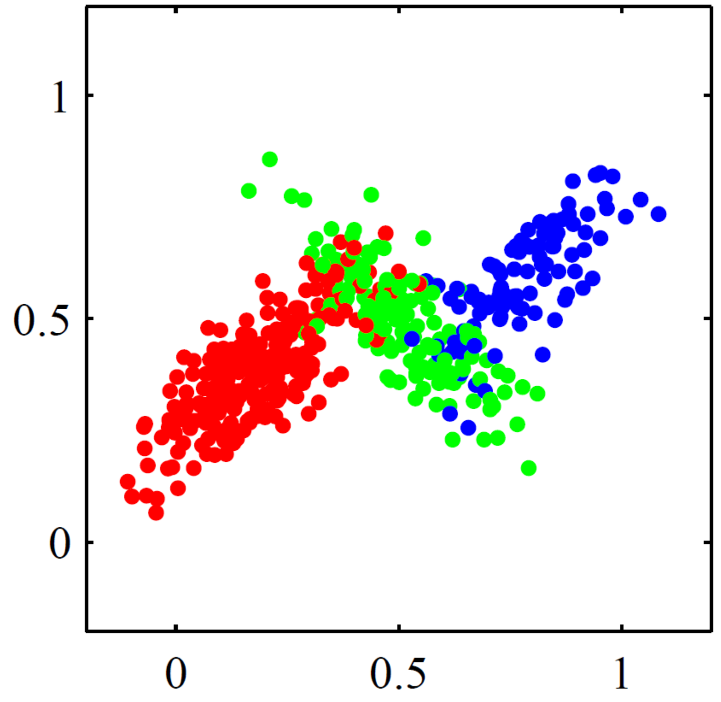


Figure from Wikipedia

GMM: Formal definition

A GMM has the following density function:

$$p(\boldsymbol{x}) = \sum_{j=1}^{k} \pi_j N(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

where

- k: the number of Gaussian components (same as #clusters we want)
- $\pi_1, \ldots, \pi_k$: mixture weights, a distribution over k components
- $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$: mean and covariance matrix of the k-th Gaussian
- N: the density function for a Gaussian

Another view

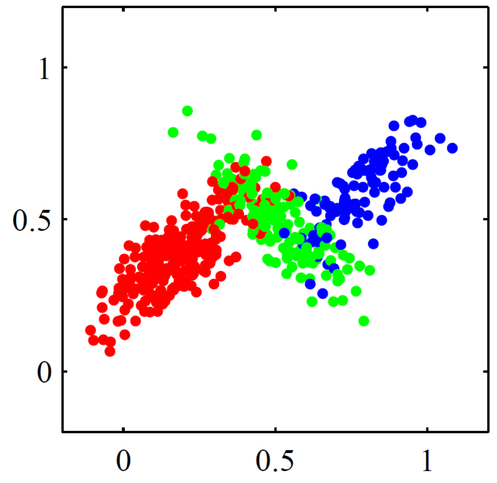
By introducing a **latent variable** $z \in [k]$, which indicates cluster membership, we can see p as a **marginal distribution**

$$p(\boldsymbol{x}) = \sum_{j=1}^{k} p(\boldsymbol{x}, z = j) = \sum_{j=1}^{k} p(z = j) p(\boldsymbol{x} | z = j) = \sum_{j=1}^{k} \pi_j N(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

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

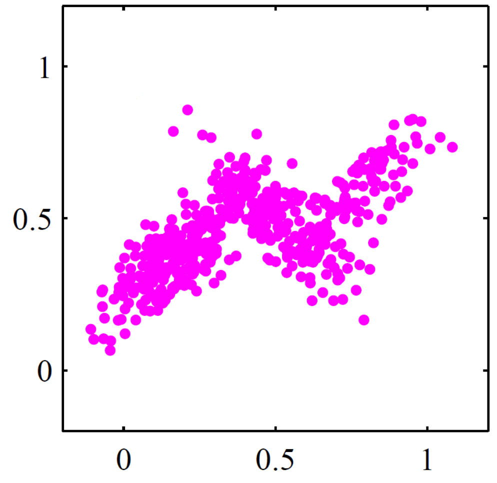
- $\boldsymbol{x}$ is observed
- z is unobserved/latent

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 $\boldsymbol{\theta} = \{\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}_{j=1}^k$.

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

$$p(z_i = j \mid \boldsymbol{x}_i) := \gamma_{ij} \in [0, 1]$$

i.e. "soft assignment" of each point to each cluster, as opposed to "hard assignment" by k-means.

GMM is more explanatory than k-means

- both learn the cluster centers $\boldsymbol{\mu}_j$'s
- in addition, GMM learns cluster weight π_j and covariance $\boldsymbol{\Sigma}_j$, thus
  - we can *predict probability of seeing a new point*
  - we can *generate synthetic data*

Gaussian Mixture Model

- Introduction
- Learning the parameters
- EM algorithm
- EM for the Gaussian Mixture Model

How do we learn the parameters?

As always, we want to do **maximum-likelihood estimation (MLE)**: find

$$\operatorname*{argmax}_{\boldsymbol{\theta}} \ \ln \prod_{i=1}^{n} p(\boldsymbol{x}_i \,; \boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ln p(\boldsymbol{x}_i \,; \boldsymbol{\theta}) := \operatorname*{argmax}_{\boldsymbol{\theta}} P(\boldsymbol{\theta}).$$

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.

$$\begin{aligned} P(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \ln p(\boldsymbol{x}_i \,; \boldsymbol{\theta}) = \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} p(\boldsymbol{x}_i, z_i = j \,; \boldsymbol{\theta}) \right) \\ &= \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} p(z_i = j \,; \boldsymbol{\theta}) p(\boldsymbol{x}_i | z_i = j \,; \boldsymbol{\theta}) \right) = \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} \pi_j N(\boldsymbol{x}_i \mid \mu_j, \boldsymbol{\Sigma}_j) \right). \end{aligned}$$

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 $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ for each $j \in [k]$

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

$$\gamma_{ij} = p(z_i = j \mid \boldsymbol{x}_i) \propto \pi_j N\left(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$$

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

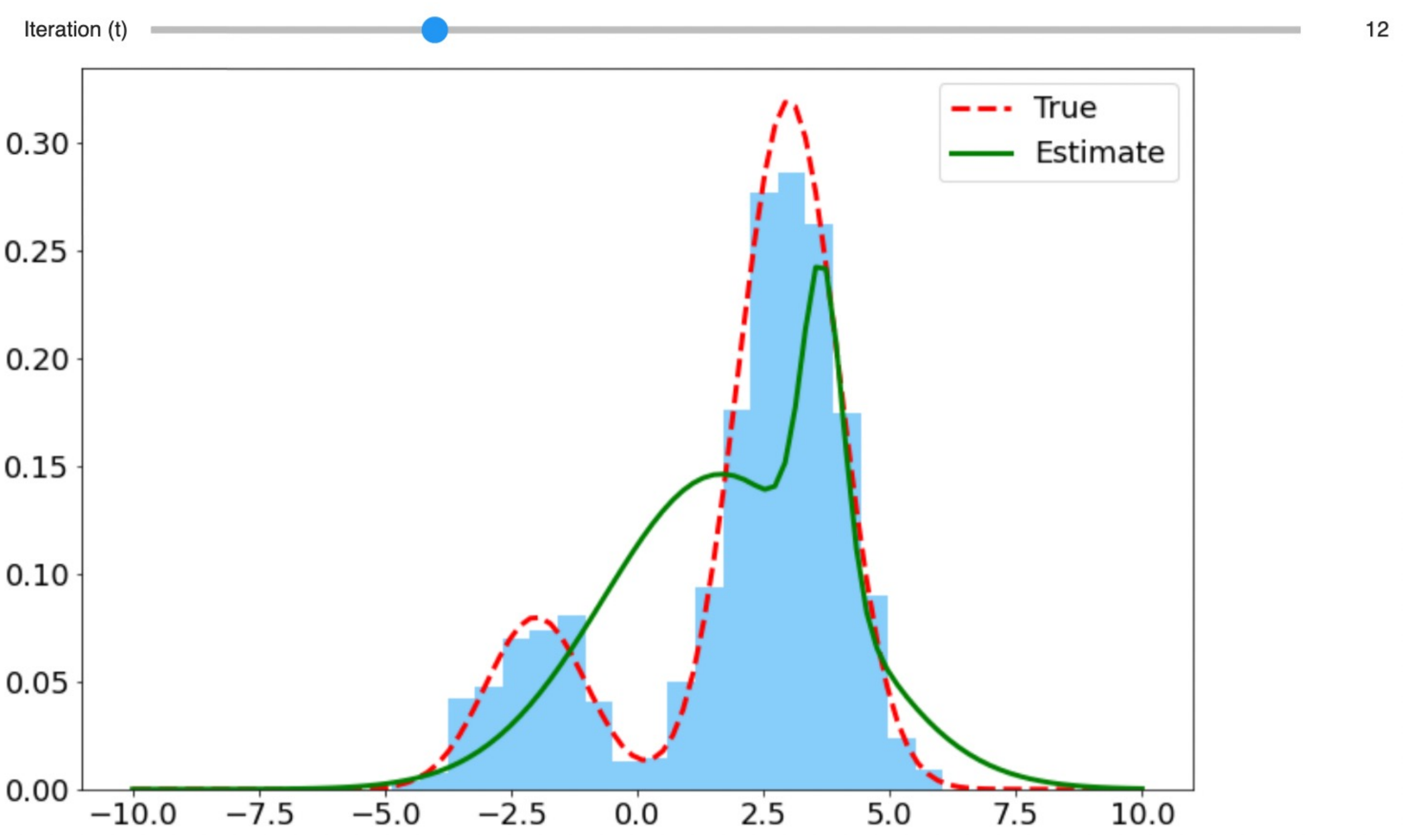
$$\pi_j = \frac{\sum_i \gamma_{ij}}{n} \qquad \boldsymbol{\mu}_j = \frac{\sum_i \gamma_{ij} \boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

$$\boldsymbol{\Sigma}_j = \frac{1}{\sum_i \gamma_{ij}} \sum_i \gamma_{ij} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}}$$

Step 3 return to Step 1 if not converged

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

Demo



See Colab notebook

Gaussian Mixture Model

- Introduction
- Learning the parameters
- EM algorithm
- EM for the Gaussian Mixture Model

EM algorithm

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

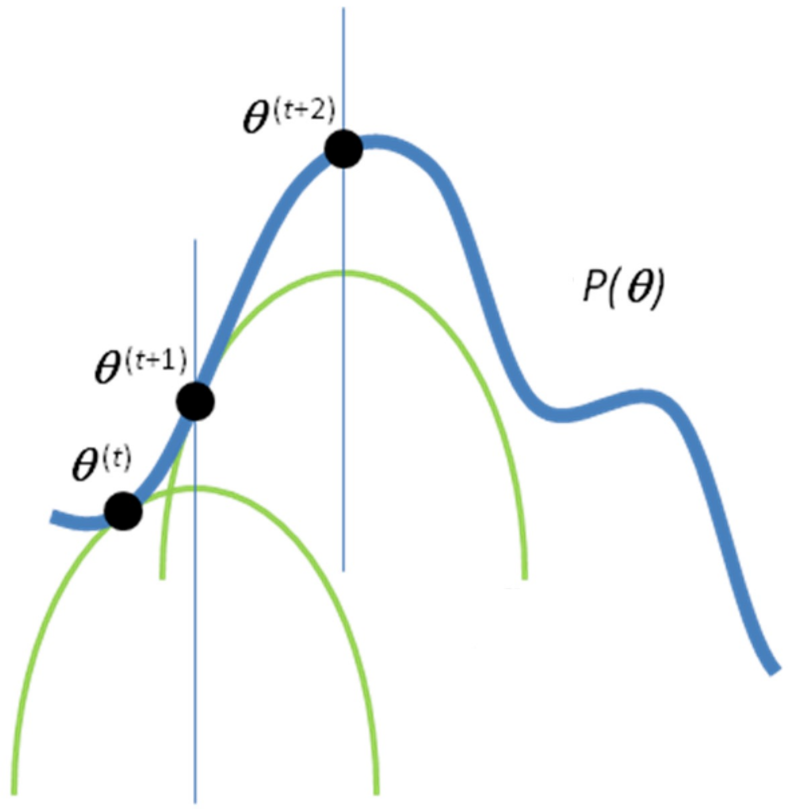
$$P(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln p(\boldsymbol{x}_i \,; \boldsymbol{\theta}) = \sum_{i=1}^{n} \ln \int_{z_i} p(\boldsymbol{x}_i, z_i \,; \boldsymbol{\theta}) dz_i$$

- $\boldsymbol{\theta}$ is the parameters for a general probabilistic model
- $\boldsymbol{x}_i$'s are observed random variables
- z_i's are latent variables

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

Figure from Wikipedia

A lower bound on the log likelihood

Finding the lower bound of P:

$$\begin{aligned}\ln p(\boldsymbol{x}\,;\boldsymbol{\theta}) &= \ln\left(\sum_{z=1}^{k} p(\boldsymbol{x}, z\,;\boldsymbol{\theta})\right)\\ &= \ln\left(\sum_{z=1}^{k} q(z)\frac{p(\boldsymbol{x}, z\,;\boldsymbol{\theta})}{q(z)}\right)\\ &= \ln\left(\mathbb{E}_{z\sim q(z)}\left[\frac{p(\boldsymbol{x}, z\,;\boldsymbol{\theta})}{q(z)}\right]\right)\\ &\geq \mathbb{E}_{z\sim q(z)}\left[\ln\left(\frac{p(\boldsymbol{x}, z\,;\boldsymbol{\theta})}{q(z)}\right)\right].\end{aligned}$$

Therefore, our log-likelihood can be written as,

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

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 $\boldsymbol{\theta}$, we should choose $\{q_i\}$ to try to make the lower bound tight at that value of $\boldsymbol{\theta}$, i.e. make the inequality hold with equality at that value of $\boldsymbol{\theta}$.

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

Suppose we fix $\boldsymbol{\theta}^{(t)}$, 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? The function should be a constant function, i.e.

$$\frac{p(\boldsymbol{x}_i, z_i\ ; \boldsymbol{\theta}^{(t)})}{q_i^{(t)}(z_i)} = c_i$$

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

Maximizing over $\{q_i\}$

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

$$c_i = \sum_{z_i=1}^{k} p(\boldsymbol{x}_i, z_i \;; \boldsymbol{\theta}^{(t)})$$

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

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

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

- $F\left(\boldsymbol{\theta}, \{q_i^{(t)}\}\right) \leq P(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$.
- $F\left(\boldsymbol{\theta}^{(t)}, \{q_i^{(t)}\}\right) = P(\boldsymbol{\theta}^{(t)})$

Maximizing over $\boldsymbol{\theta}$

Fix $\{q_i^{(t)}\}$, maximize over $\boldsymbol{\theta}$:

$$\begin{aligned}
&\operatorname*{argmax}_{\boldsymbol{\theta}} F\left(\boldsymbol{\theta}, \{q_i^{(t)}\}\right) \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln\left(\frac{p(\boldsymbol{x}_i, z_i\ ; \boldsymbol{\theta})}{q_i^{(t)}(z_i)}\right)\right] \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(\boldsymbol{x}_i, z_i\ ; \boldsymbol{\theta})\right] - \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln(q_i^{(t)}(z_i))\right] \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} \left[\ln p(\boldsymbol{x}_i, z_i\ ; \boldsymbol{\theta})\right] \\
&:= \operatorname*{argmax}_{\boldsymbol{\theta}}\ Q(\boldsymbol{\theta}\ ; \boldsymbol{\theta}^{(t)}) \qquad\qquad (\{q_i^{(t)}\} \text{ are computed via } \boldsymbol{\theta}^{(t)})
\end{aligned}$$

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

- versus the incomplete likelihood: $P(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln p(\boldsymbol{x}_i\ ; \boldsymbol{\theta})$

General EM algorithm

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

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

$$q_i^{(t)}(z_i) = p(z_i \mid \boldsymbol{x}_i \,; \boldsymbol{\theta}^{(t)})$$

and obtain **Expectation** of complete likelihood

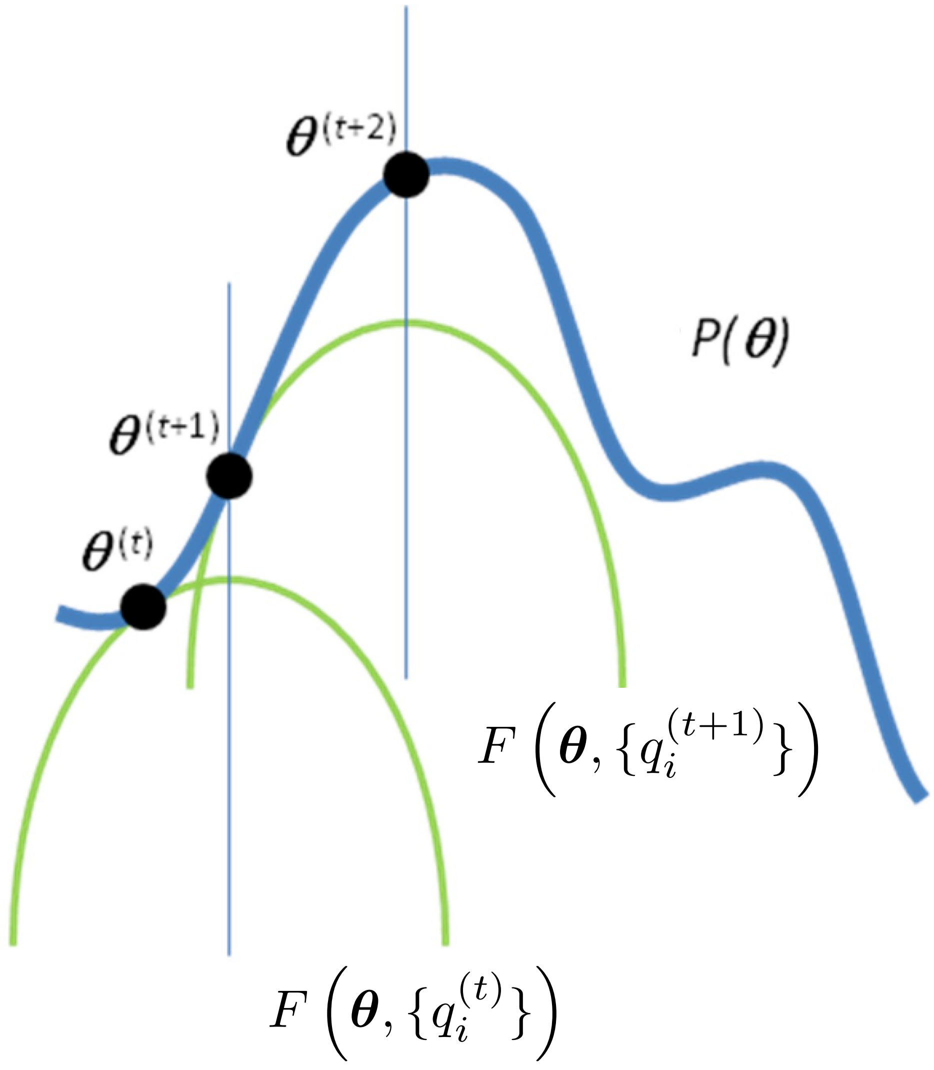
$$Q(\boldsymbol{\theta} \,; \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} [\ln p(\boldsymbol{x}_i, z_i \,; \boldsymbol{\theta})]$$

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

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

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

Pictorial explanation


$\boldsymbol{\theta}^{(t+2)}$
$\boldsymbol{\theta}^{(t+1)}$
$\boldsymbol{\theta}^{(t)}$
$P(\boldsymbol{\theta})$
$F\left(\boldsymbol{\theta}, \{q_i^{(t+1)}\}\right)$
$F\left(\boldsymbol{\theta}, \{q_i^{(t)}\}\right)$

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

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

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

Gaussian Mixture Model

- Introduction
- Learning the parameters
- EM algorithm
- EM for the Gaussian Mixture Model

Applying EM to learn GMMs: E-Step

E-Step:

$$\begin{aligned} q_i^{(t)}(z_i = j) &= p\left(z_i = j \mid \boldsymbol{x}_i \,; \boldsymbol{\theta}^{(t)}\right) \\ &\propto p\left(\boldsymbol{x}_i, z_i = j \,; \boldsymbol{\theta}^{(t)}\right) \\ &= p\left(z_i = j \,; \boldsymbol{\theta}^{(t)}\right) p(\boldsymbol{x}_i \mid z_i = j \,; \boldsymbol{\theta}^{(t)}\Big) \\ &= \pi_j^{(t)} N\left(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)}\right) \end{aligned}$$

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

Applying EM to learn GMMs: M-Step

M-Step:

$$\begin{aligned}
\operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} [\ln p(\boldsymbol{x}_i, z_i \; ; \boldsymbol{\theta})] \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathbb{E}_{z_i \sim q_i^{(t)}} [\ln p(z_i \; ; \boldsymbol{\theta}) + \ln p(\boldsymbol{x}_i | z_i \; ; \boldsymbol{\theta})] \\
&= \operatorname*{argmax}_{\{\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}} \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \left(\ln \pi_j + \ln N(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)
\end{aligned}$$

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

$$\operatorname*{argmax}_{\boldsymbol{\pi}} \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \ln \pi_j$$

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

$$\operatorname*{argmax}_{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j} \sum_{i=1}^{n} \gamma_{ij} \ln N(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

Applying EM to learn GMMs: M-Step

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

$$\boldsymbol{\mu}_j = \frac{\sum_i \gamma_{ij}\boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

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

$$\boldsymbol{\Sigma}_j = \frac{1}{\sum_i \gamma_{ij}} \sum_i \gamma_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}}$$

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

You will verify some of these in HW4.

Applying EM to learn GMMs: Putting it together

EM for learning GMMs:

Step 0 Initialize $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ for each $j \in [k]$

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

$$\gamma_{ij} = p(z_i = j \mid \boldsymbol{x}_i) \propto \pi_j N\left(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$$

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

$$\pi_j = \frac{\sum_i \gamma_{ij}}{n} \qquad \boldsymbol{\mu}_j = \frac{\sum_i \gamma_{ij} \boldsymbol{x}_i}{\sum_i \gamma_{ij}}$$

$$\boldsymbol{\Sigma}_j = \frac{1}{\sum_i \gamma_{ij}} \sum_i \gamma_{ij} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}}$$

Step 3 return to Step 1 if not converged