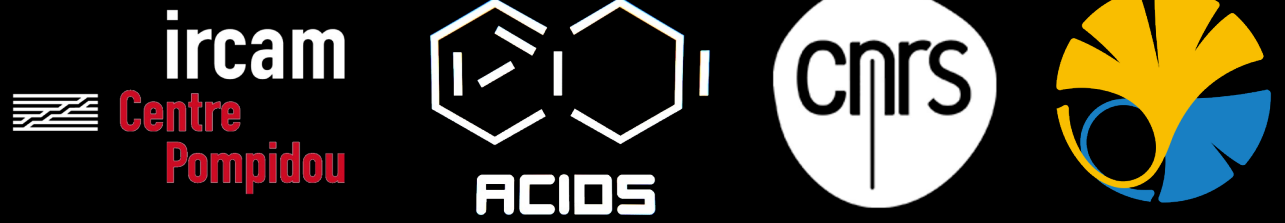


Latent models, EM and GMM

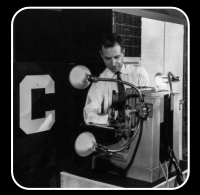
Creative Machine Learning - Course 06

Pr. Philippe Esling
esling@ircam.fr

Brief history of AI

1943 - Neuron

First model by McCulloch & Pitts (purely theoretical)



1957 - Perceptron

Actual **learning machine** built by Frank Rosenblatt

Learns character recognition analogically

1986 - Backpropagation

First to learn neural networks efficiently (*G. Hinton*)

1989 - *Convolutional NN*

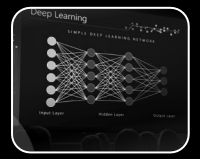
Mimicking the vision system in cats (*Y. LeCun*)

This lesson

2012 - *Deep learning*

Layerwise training to have deeper architectures

Swoop all state-of-art in classification competitions



Lesson #6

2015 - *Generative model*

First wave of interest in generating data

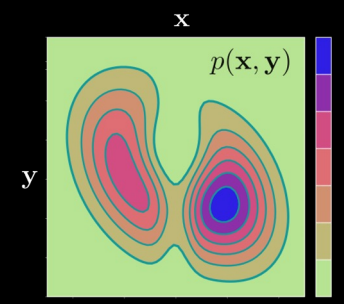
Led to current model craze (VAEs, GANs, Diffusion)

2012 onwards Deep learning era

Brief history of AI

Pre-requisites for understanding generative models

1

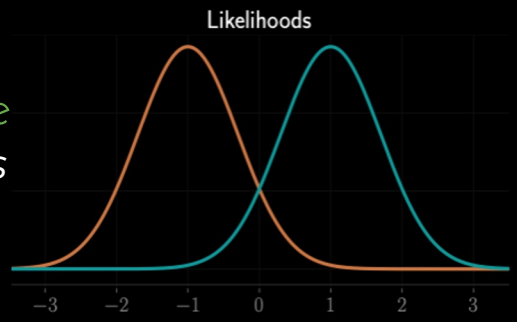


Probability theory

Random variables, distributions, independence

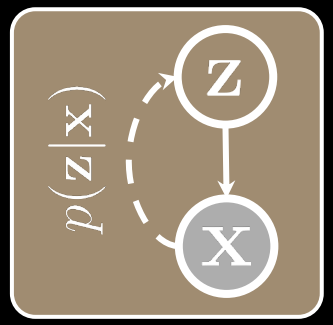
Bayesian inference

Bayes' theorem, likelihood, conjugate priors



2

3

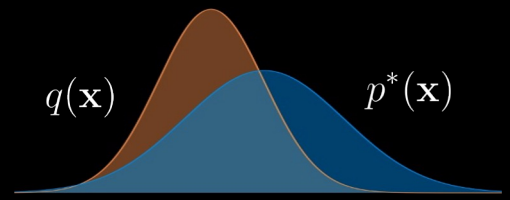


Latent models

Latent variables, probability graphs

Approximate inference

Latent variables, probability graphs



4

Lesson #6

2015 - *Generative model*

First wave of interest in generating data

Led to current model craze (VAEs, GANs, Diffusion)

2012 onwards Deep learning era

Supervised learning

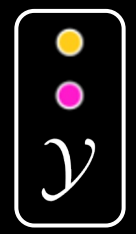
Typical machine learning tasks

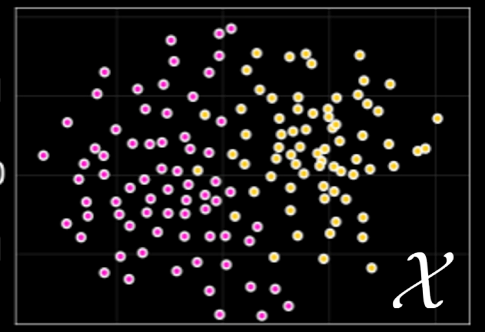
Until now, we mostly dealt with **supervised** problems

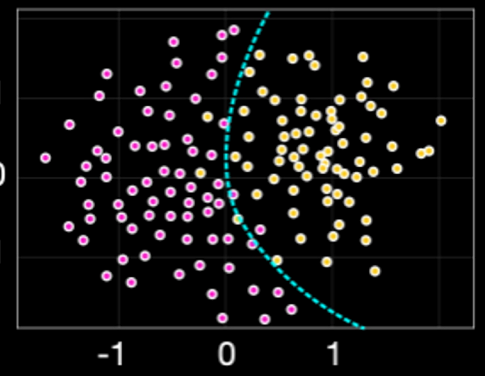
- *Classification, regression*
- *Implies that we have labels*

Based on *labeled* data, assign class to *unlabeled data*

- *Multiple approaches to solve this task*
- *We have also seen the probabilistic formulation*







Probabilistic formulation

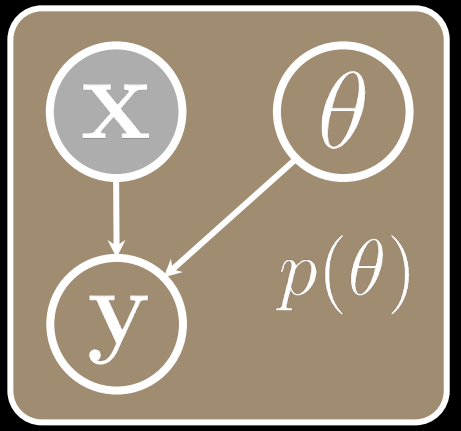
We have data $\mathbf{x} \in \mathbb{R}^n$ and labels $\mathbf{y} \in \mathbb{R}^m$

Supervised learning tries to model $p(\mathbf{y}|\mathbf{x})$

We define a parametric model $p_\theta(\mathbf{y}|\mathbf{x})$

Optimization might be seen as solving for

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \theta) d\theta$$



Graphical model

Unsupervised learning

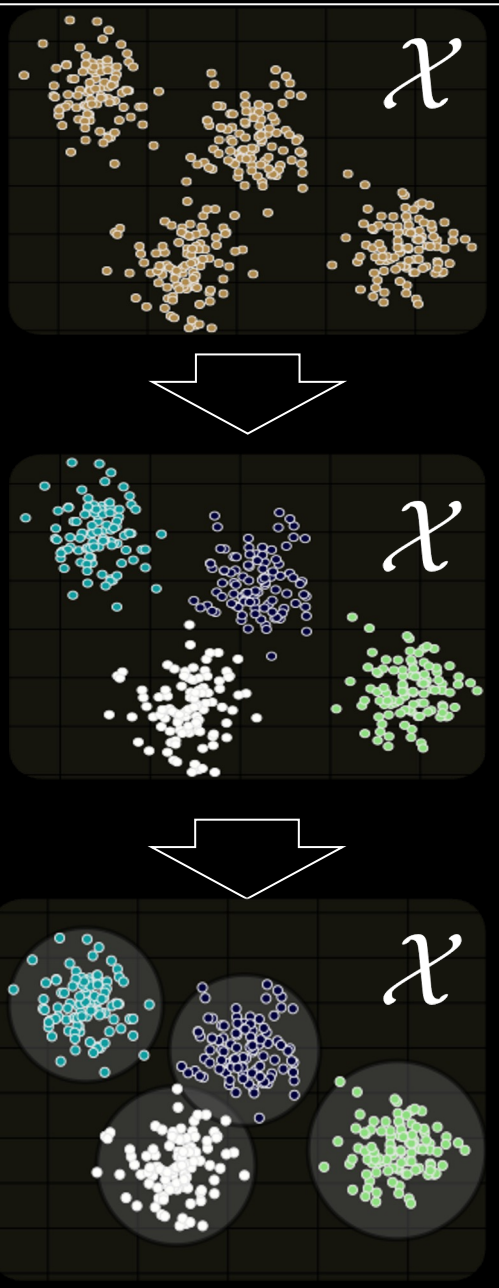
Laying out the framework

In some cases, we might only have access to $\mathbf{x} \in \mathbb{R}^n$

- *No labels are available (**costly**, **ill-defined**)*
- *Structure still exists in any data*
- *As humans we perceive it easily*

How to learn from this ?

Here we can clearly see four clusters
This is a form of unsupervised structure



Unsupervised clustering

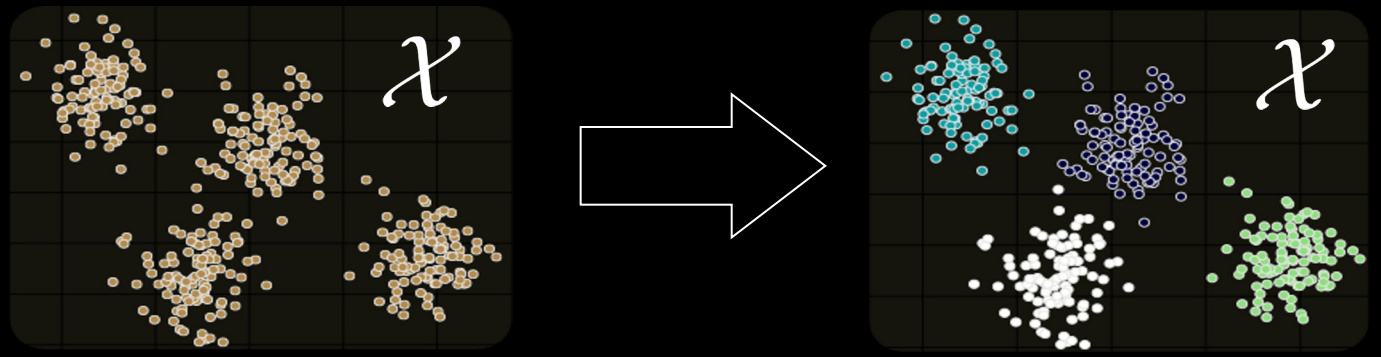
Aim to model the entire data distribution $\mathbf{x} \sim p(\mathbf{x})$

How to choose the *form* of the distribution ?
How to *optimize* it directly ?

Seems to be missing information

Unsupervised learning

We need to **add information** to our problem, but only have $\mathbf{x} \in \mathbb{R}^n$



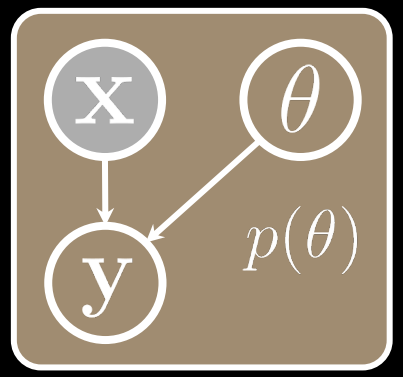
*Structure is clearly defined by the **cluster identity***

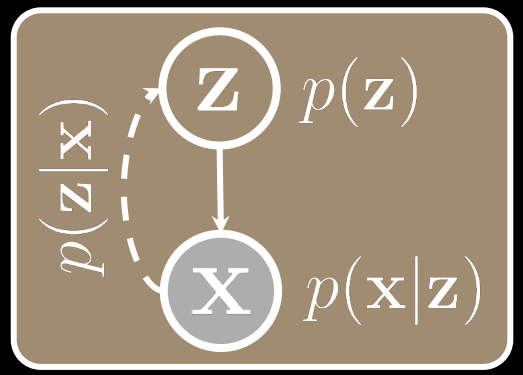
Introducing the need for **latent variables** $\mathbf{z} \in \mathbb{R}^z$

Allows to clearly define the *hidden information*

Can be seen as the **hidden factors that generated the data**

Supervised





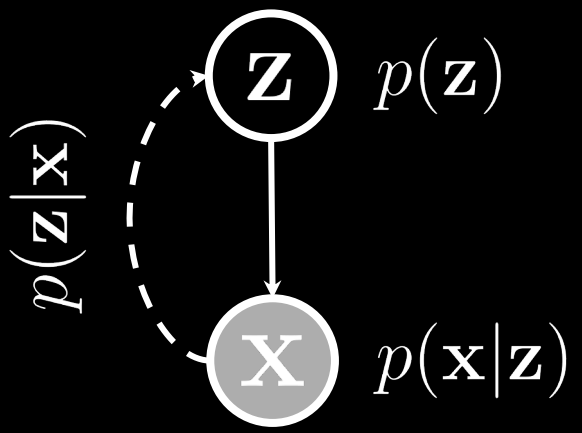
Latent (unsupervised)

Latent variables

We are interested in the **generative problem** $\mathbf{x} \sim p(\mathbf{x})$

This problem is highly complex (distribution)

So we introduce **latent variables** $\mathbf{z} \in \mathbb{R}^z \ll \mathbb{R}^x$



latent variable: i.e. hair color, eye color, …

Happy, female *Neutral, male*

$$p(\mathbf{x}) = \int_{\mathbb{R}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Understanding latent variables

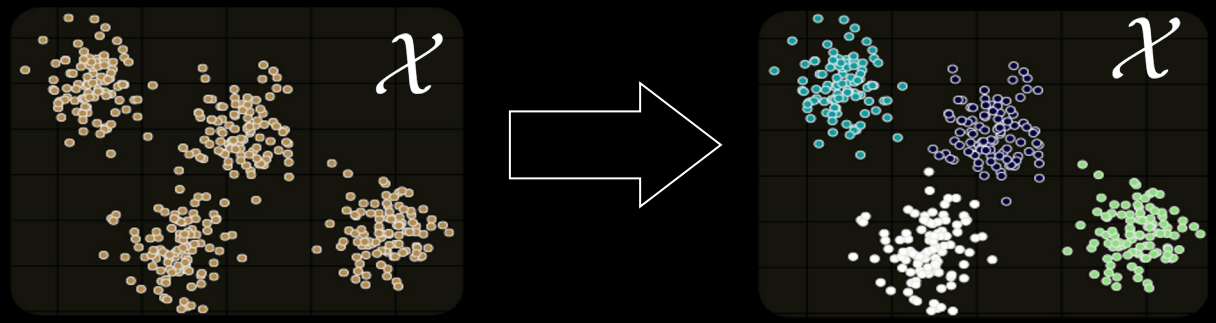
Variables that led to generate the data

Hidden variation factors in the data structure

Variables that simplify our optimization task

Defining clustering

Clustering tasks



Goal

Attach label to each data points (*"unsupervised classification"*)
Aim to assign same label to data points that are "close"
Clustering implicitly relies on a distance metric

Data *(input)* $\mathcal{X}$ → Unsupervised Learning → $\mathbf{z}$ Structure *(output)*

Objective function ?

First approach

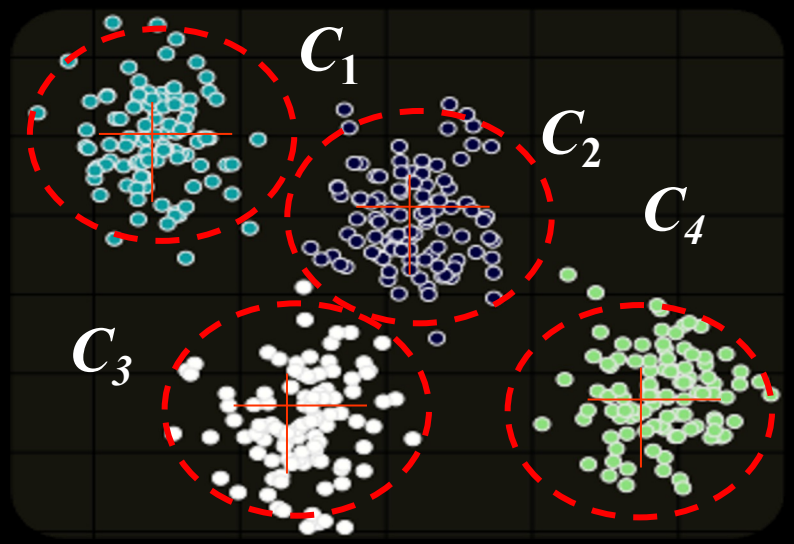
Address this in a *non-probabilistic* way
How can we cluster a given set of points ?

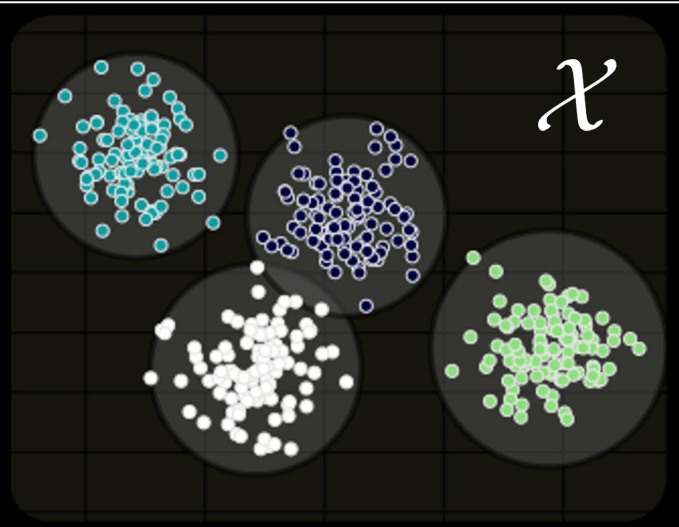
Clustering intuition

What defines a **good** clustering ?

Evaluate the within-clusters distance (centroid-based)

$$\underset{C_j, m_{i,j}}{\operatorname{argmin}} \left[\sum_j \sum_i (\mathbf{x}_i - \mathbf{c}_j) \right]$$





How to optimize ?

Fixed *memberships* $m_{i,j}$

We could *infer centroids*

$$\mathbf{c}_j = \frac{\sum_{i=1}^{n} m_{i,j} \mathbf{x}_i}{\sum_{i=1}^{n} m_{i,j}}$$

Fixed centroids $\mathbf{c}_j$

We could infer the memberships

distance between point and centroid

$$m_{i,j} = \underset{k}{\operatorname{argmin}} \ (\mathbf{x}_i - \mathbf{c}_j)^2$$

Implementing clustering

Summarizing our observations

Obviously, memberships $m_{i,j}$ and centroids $\mathbf{c}_j$ are correlated

It seems that we have an alternating problem definition

1. Knowing the memberships $m_{i,j}$

   We compute centroids

$$\mathbf{c}_j = \frac{\sum_{i=1}^{n} m_{i,j}\mathbf{x}_i}{\sum_{i=1}^{n} m_{i,j}}$$

2. Knowing the centroids $\mathbf{c}_j$

   We compute memberships

$$m_{i,j} = \underset{k}{\operatorname{argmin}} \ (\mathbf{x}_i - \mathbf{c}_j)^2$$

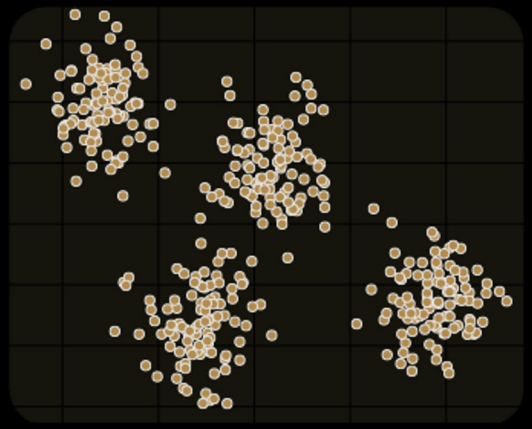
This can be solved by an iterative algorithm

- Starts from *random cluster centroids initialization*
- Alternates between the two types of updates

Seminal K-Means clustering algorithm

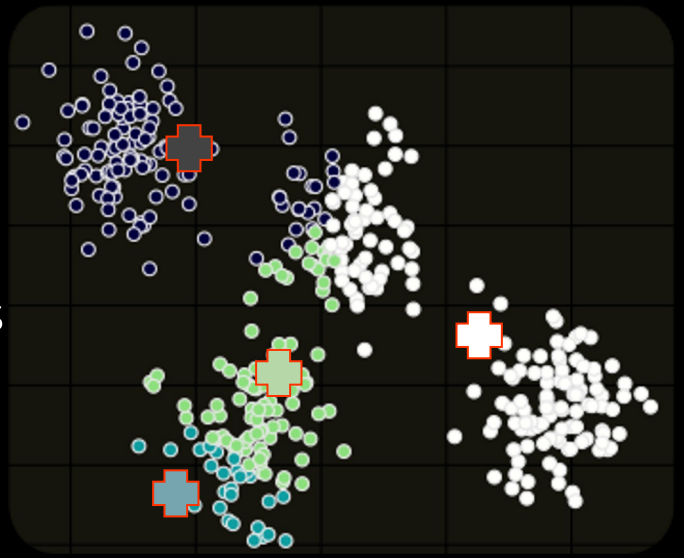
K-Means for clustering

Detailing the K-Means algorithm



Initialization

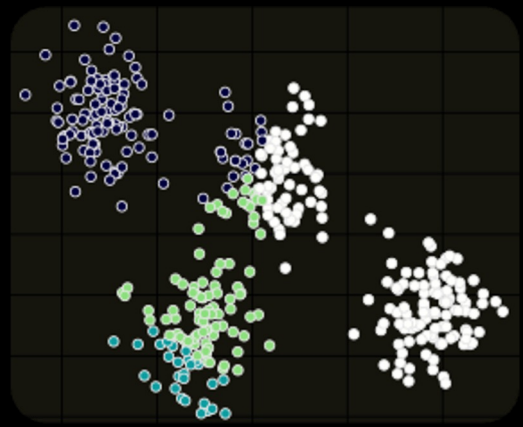
1. Random init of cluster centers
2. Compute membership
3. Assign to points

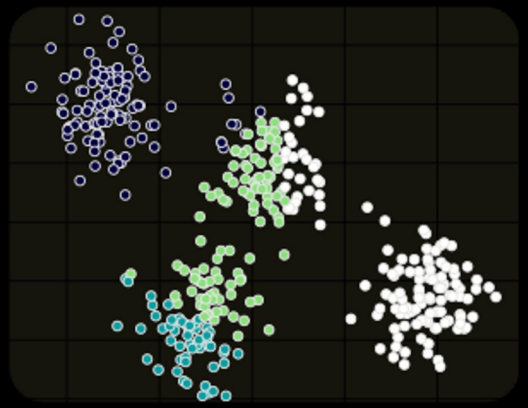


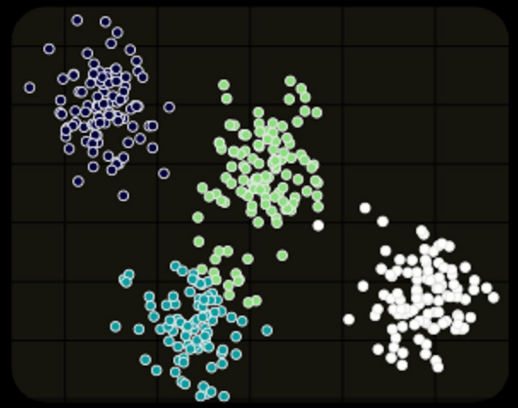
Alternate updates

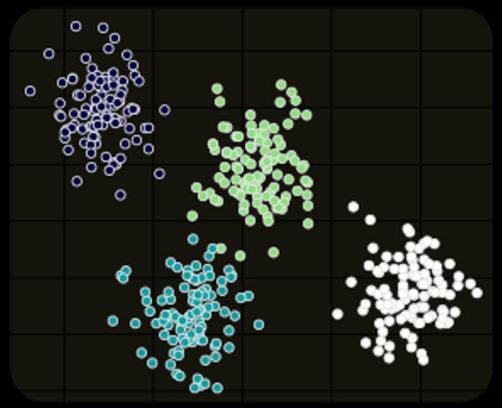
Then perform loop of alternating updates

1. Re-compute the centroid of each cluster
2. Determine membership of all the datapoints









Details of the K-Means algorithm

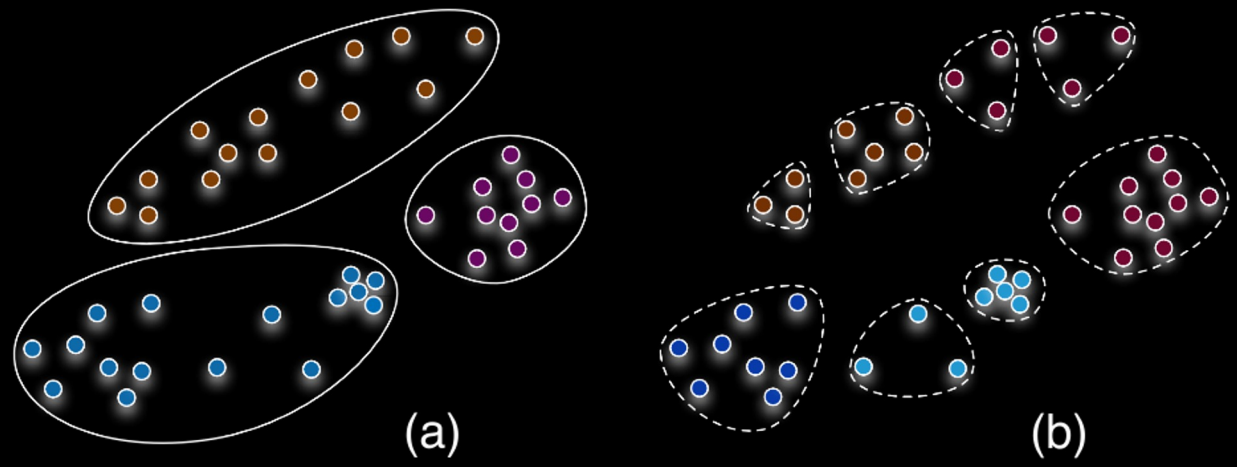
Issues in the algorithm

How do we define the *stopping criterion* ?

1. *Iterations number or thresholding*
2. *Measure of cluster quality* ---------- $\mathcal{L}_K = \sum_{n=1}^{N} \sum_{k=1}^{K} \| x_n - \mathbf{c}_k \|^2$

How do we choose the number of clusters

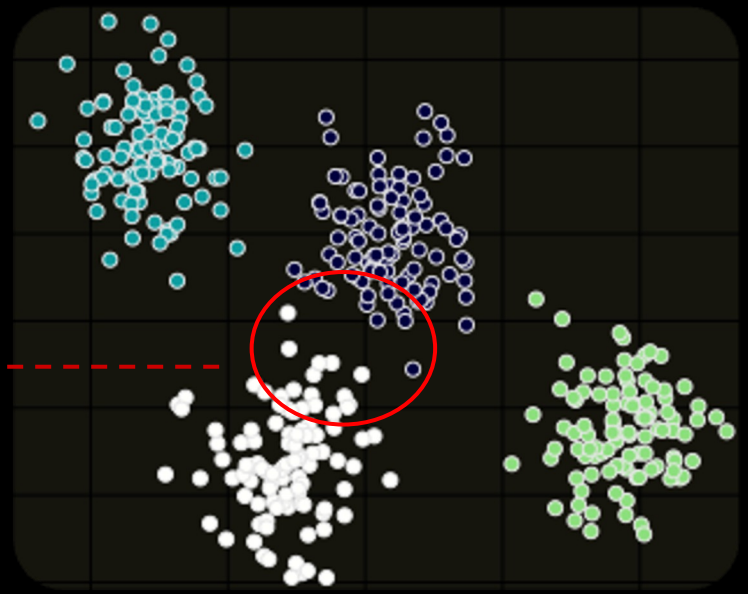
1. *Even with the right number, will we find a good optima?*
2. ***Problem** = Any subset of points is a valid clustering*



Hard or soft clustering

Problems with deterministic clustering

K-Means assign single memberships
*Each point belongs to **one** cluster*



For some examples this is inadequate

K-Means is a form of hard (crisp) clustering

Membership is an indicator function

Would seem more flexible to define ***probabilities*** of membership

Given datapoints $\mathbf{X}_i$ cluster index C_k and centroid positions $\mathbf{C}_k$

Aim to define

- Membership probabilities $p(\mathbf{x}_i|C_k)$
- Cluster parameters probability $p_\theta(C_k|\mathbf{x}_i)$

Probabilistic clustering: **centroids**

Looping back to the Bayesian framework

- All parameters are probabilities
- We can safely rely on the Bayes rule to have

$$p(C_k|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|C_k)p(C_k)}{\sum_{l=1}^{k} p(\mathbf{x}_i|C_l)p(C_l)}$$

$\mathbf{x}_i$ Datapoints
C_k Cluster index
$\mathbf{c}_k$ Centroid positions

Need to assign weights to each cluster

- Weights may not be equal
- Similar to **prior probabilities** $p(C_k) = \frac{1}{m}\sum_{i=1}^{m} p(C_k|\mathbf{x}_i)$

Then we can infer centroid positions as

$$\mathbf{c}_k = \frac{\sum_{i=1}^{m} \mathbf{x}_i p(C_k|\mathbf{x}_i)}{\sum_{i=1}^{m} p(C_k|\mathbf{x}_i)}$$

weight is the probability for calculation of centroids

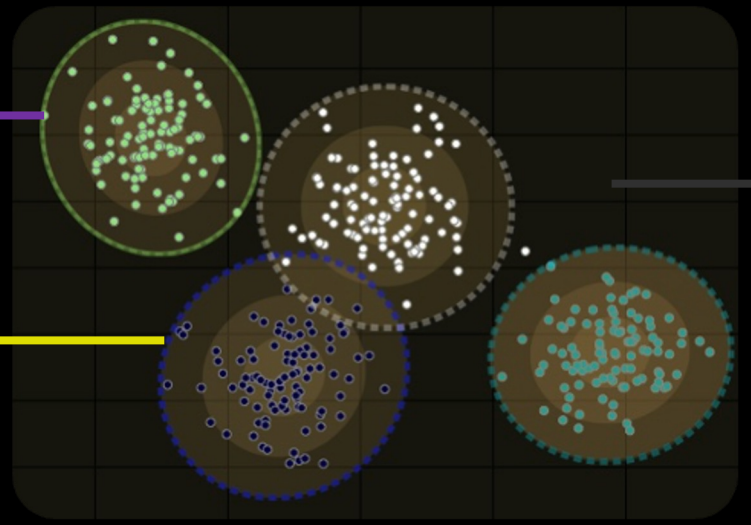
Probabilistic interpretation of clustering

Formal definition of the problem

- We can *define one Gaussian distribution* per cluster
- For each cluster we have $p(\mathbf{x}_i|C_k) = \mathcal{N}(\mu_k, \mathbf{\Sigma}_k)$

$p(\mathbf{x}_i|C_k = 1)$

$p(\mathbf{x}_i|C_k = 2)$



$p(\mathbf{x}_i|C_k = 3)$

$p(\mathbf{x}_i|C_k = 4)$

- However note that the *full density is multi-modal*
- Each mode represents a sub-population
  - Unimodal Gaussian for each group

$$p(\mathbf{x}_i) = \sum_k \underbrace{p(\mathbf{x}_i|C_k)}_{\text{Unimodal}} \boxed{p(C_k)}$$

Mixing coefficients

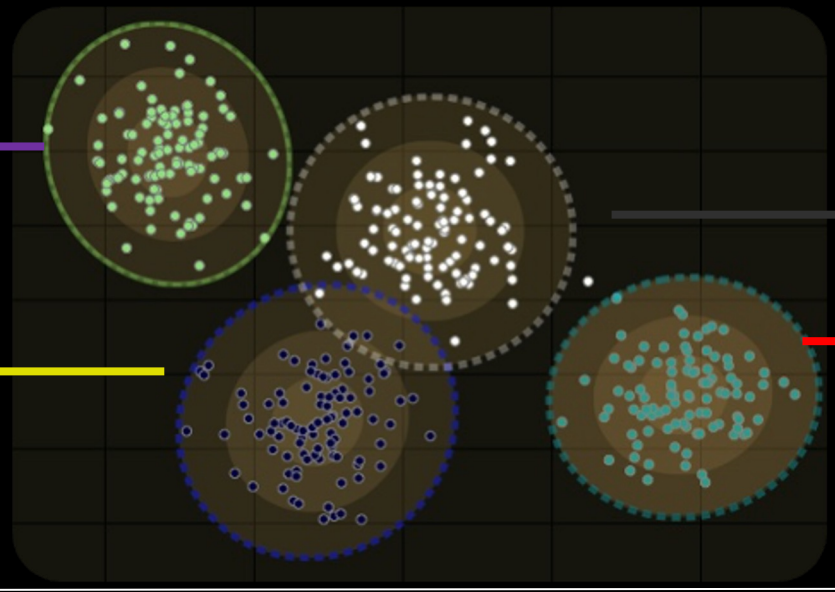
Probabilistic interpretation of clustering

Performing optimization for the problem

We have the exact same issue as K-Means

If we knew the cluster parameters $\theta_{\mathbf{k}} = \{\mu_k, \mathbf{\Sigma}_k\}$

- Then estimating $p(\mathbf{x}_i|C_k)$ is easy
- And also consequently estimating $p(C_k)$
  - ***Maximum likelihood estimation***



$p(\mathbf{x}_i|C_k = 1)$

$p(\mathbf{x}_i|C_k = 2)$

$p(\mathbf{x}_i|C_k = 3)$

$p(\mathbf{x}_i|C_k = 4)$

Probabilistic interpretation of clustering

Performing optimization for the problem

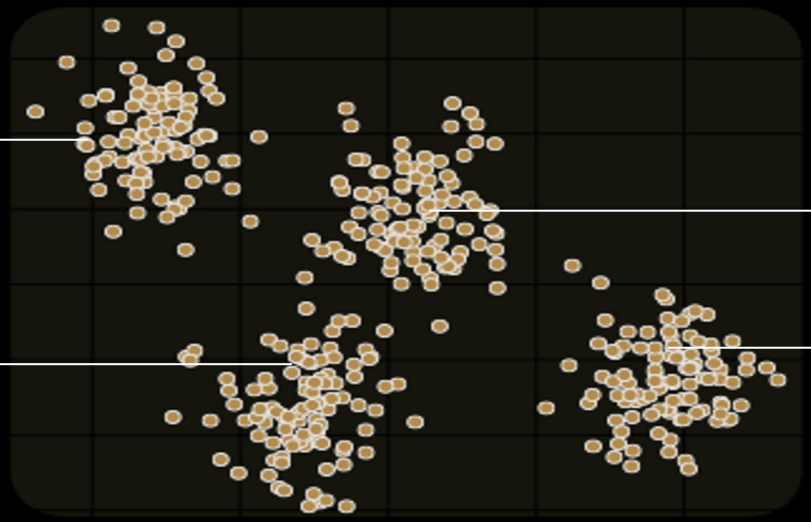
- But in reality, we do not have access to $\theta_{\mathbf{k}} = \{\mu_k, \mathbf{\Sigma}_k\}$
  - Now estimating $p(\mathbf{x}_i|C_k)$ and $p(C_k)$ is hard
  - We need to address a much more complex problem

$$\max_{\theta_{\mathbf{k}}} \sum_i \log \sum_k p(\mathbf{x}_i|C_k;\theta_{\mathbf{k}})p(C_k;\theta_{\mathbf{k}})$$

  - Need a similar alternating approach

? $p(\mathbf{x}_i|C_k = 1)$

? $p(\mathbf{x}_i|C_k = 2)$



$p(\mathbf{x}_i|C_k = 3)$?

$p(\mathbf{x}_i|C_k = 4)$?

Expectation-Maximization

Introducing Expectation-Maximization (EM)

All data ($\mathbf{X}$) observable

- Use the *Maximum likelihood* estimator
- Optimize the analytic form of $\mathcal{L}(\theta) = \log p(\mathbf{x}; \theta)$

Missing (**latent**) data

- The full data is $\mathbf{X}$ (observed) *and* $\mathbf{Z}$ (latent)
- The likelihood becomes

$$\mathcal{L}(\theta) = \log p(\mathbf{x}, \mathbf{z}; \theta)$$

Performing **optimization**

- As the likelihood is intractable in most cases
- We are going to approximate it

Expectation-Maximization

EM iteratively maximizes likelihood

The complete likelihood is $\mathcal{L}(\theta) = \log p(\mathbf{x}, \mathbf{z}; \theta)$

- Start from a random initial guess $\theta^{(0)}$
- Then iteratively apply (until convergence)

E-Step

Compute the *expectation* of the likelihood

$$\mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\mathcal{L}(\theta)\right] = \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta^{(\mathbf{t})}) p(\mathbf{z},\mathbf{x};\theta^{(\mathbf{t})})$$

proba L(theta)

This function is called the Q-function $\mathcal{Q}(\theta;\theta^{(\mathbf{t})}) = \mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\mathcal{L}(\theta)\right]$

M-Step

Compute parameters by *maximizing* the Q-function

$$\theta^{(\mathbf{t+1})} = \underset{\theta}{\operatorname{argmax}}\ \mathcal{Q}(\theta;\theta^{(\mathbf{t})})$$

Expectation-Maximization

Summarizing the algorithm in simple terms

- Start from a random initial guess
- Then iteratively apply (until convergence)

E-Step

1. Use current parameters (and observations) to reconstruct the hidden (latent) structure

M-Step

2. Use hidden structure (and observations) to re-estimate parameters

Analysis

- Very similar to simple K-means
- Consists of assignment and updates
- Can use *random initialization*
  - Problem of local minima

Required tools for EM

Steps to **derive** the EM algorithm

1. Write the likelihood of the complete data *(decide distribution)*

$$\mathcal{L}(\theta) = \log p(\mathbf{x}, \mathbf{z}; \theta)$$ sum of Gaussians

2. E-step: write the Q function (*expectation given the observed data*)

$$\mathcal{Q}(\theta; \theta^{(\mathbf{t})}) = \mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\boldsymbol{\mathcal{L}(\theta)}\right] = \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}; \theta^{(\mathbf{t})}) p(\mathbf{z}, \mathbf{x}; \theta^{(\mathbf{t})})$$

3. M-step: solve maximization (*deriving closed form solution if possible*)

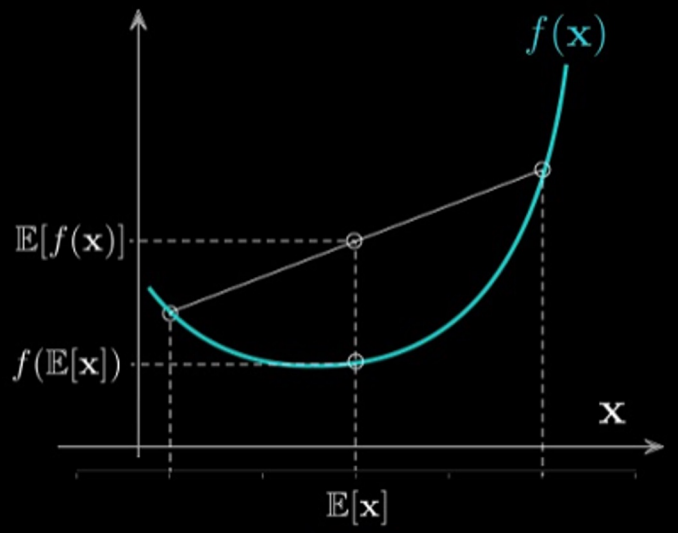
$$\theta^{(\mathbf{t+1})} = \underset{\theta}{\operatorname{argmax}}\ \mathcal{Q}(\theta; \theta^{(\mathbf{t})})$$

Usually the Q-function is too complex to obtain directly

Approximate it with several tools

- **Lower-bounding functions**
- **Kullback-Leibler divergence**
- **Jensen's inequality**

Background knowledge



Jensen inequality

If $f : \mathbb{R} \to \mathbb{R}$ is a convex function

$$\mathbb{E}\left[f(\mathbf{x})\right] \geq f\left(\mathbb{E}\left[\mathbf{x}\right]\right)$$

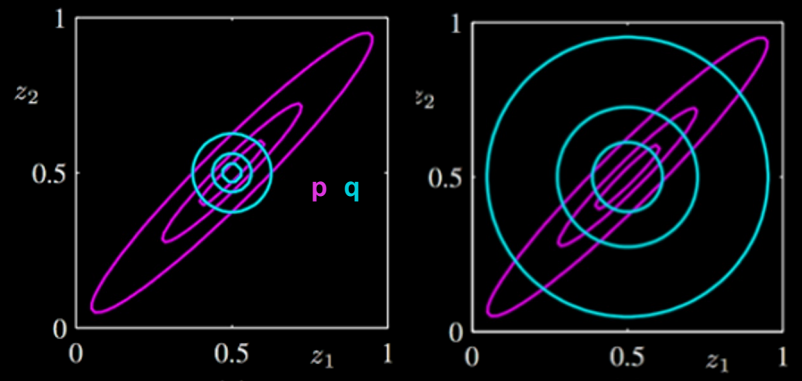
Kullback-Leibler divergence

similarity between two distributions

Measure statistical difference between distributions

/!\ assume q is a approxi of p -> D[p|q]
/!\ assume p is a approxi of q -> D[q|p]

$$\mathcal{D}_{KL}\left[p(\mathbf{z}) \parallel q(\mathbf{z})\right] \neq \mathcal{D}_{KL}\left[q(\mathbf{z}) \parallel z(\mathbf{z})\right]$$



$$\mathcal{D}_{KL}\left[p(\mathbf{z}) \parallel q(\mathbf{z})\right] = \int p(\mathbf{z}) \log \frac{p(\mathbf{z})}{q(\mathbf{z})} d\mathbf{z}$$

= Expectation[log(p/q)] with p known

$$\mathcal{D}_{KL}\left[p(\mathbf{z}) \parallel q(\mathbf{z})\right] \geq 0. \ \forall p, q$$

$$\mathcal{D}_{KL}\left[p(\mathbf{z}) \parallel q(\mathbf{z})\right] \neq \mathcal{D}_{KL}\left[q(\mathbf{z}) \parallel z(\mathbf{z})\right]$$

$$\mathcal{D}_{KL}\left[p(\mathbf{z}) \parallel q(\mathbf{z})\right] = 0 \text{ only if } p = q$$

Expectation Maximization

Writing down the E(xpectation)-Step

We want to maximize the data likelihood

$$\mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\mathcal{L}(\theta)\right] = \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta^{(\mathbf{t})}) p(\mathbf{z},\mathbf{x};\theta^{(\mathbf{t})})$$

However, distribution $p(\mathbf{x},\mathbf{z};\theta)$ might be too complex

Lower-bound idea

Find a function that satisfies $q_\theta(\mathbf{z}) \leq \mathcal{L}(\theta)$

Then we know that $\max_\theta q_\theta(\mathbf{z}) \longrightarrow \max_\theta \mathcal{L}(\theta)$

Problem: Quality of lower-bound dictates quality of approximation

Instead of finding a fixed lower-bound $q(\mathbf{z})$

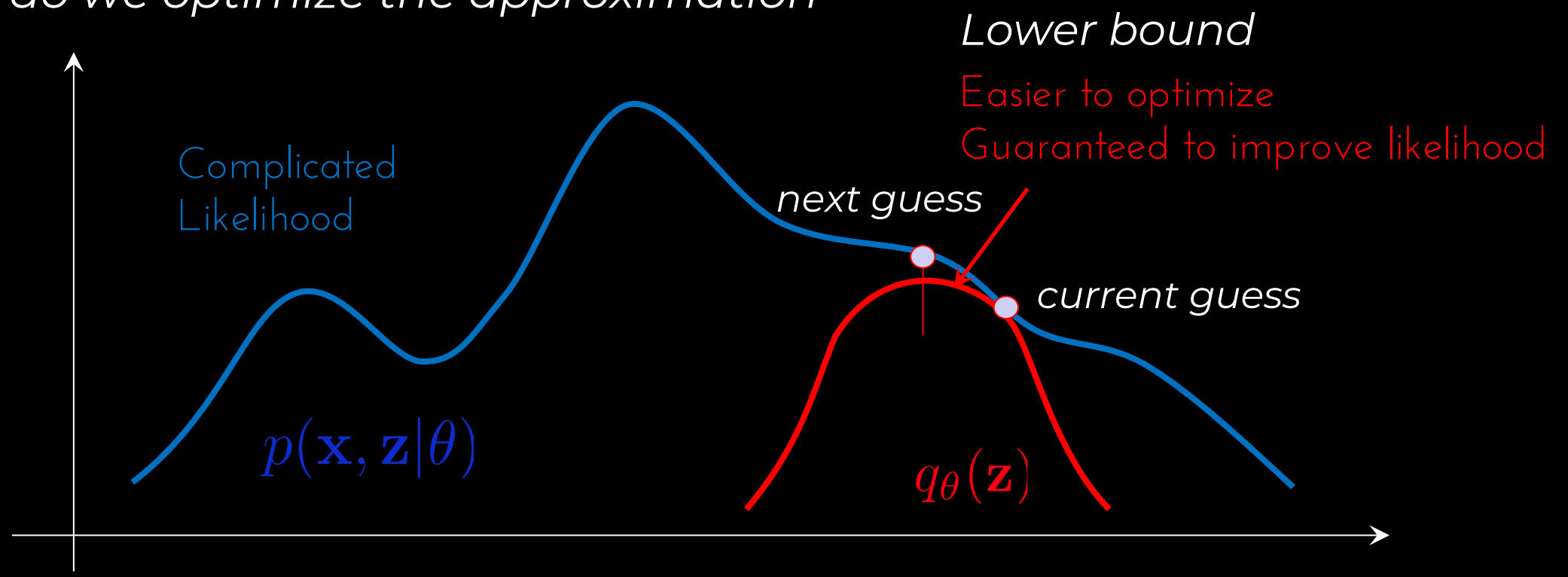
We aim to define a *parametric distribution* $q_\theta(\mathbf{z})$

Then, we can minimize the difference to our function $\mathcal{D} = \mathcal{L}(\theta) - q_\theta(\mathbf{z})$

Intuitive understanding of *lower bounding*

Understanding the idea behind lower bounding

How do we optimize the approximation



E-step = computing the lower bound

M-step = maximizing the lower bound

Expectation Maximization

How to maximize likelihood by pushing the lower bound ?

We can introduce directly the distribution

$$\mathcal{L}(\theta) = \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta) = \log \sum_{\mathbf{z}} \frac{q(\mathbf{z}) p(\mathbf{x}, \mathbf{z}; \theta)}{q(\mathbf{z})}$$

Jensen's inequality

$f(\mathbb{E}[\mathbf{x}]) \geq \mathbb{E}[f(\mathbf{x})]$

(case concave)

$$\geq \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{x}, \mathbf{z}; \theta) - \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})$$

$$= \sum_{\mathbf{z}} q(\mathbf{z}) \log \left[p(\mathbf{z}|\mathbf{x}; \theta) p(\mathbf{x}; \theta) \right] - \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})$$

$$= \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{x}; \theta)}{q(\mathbf{z}} + \log p(\mathbf{x}; \theta)$$

Constant with $q(\mathbf{z})$

Negative KL-divergence $\mathcal{D}_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}; \theta)\right]$

Final expression

$$\mathcal{L}(\theta) \geq -\mathcal{D}_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x}; \theta)\right] + const$$

Expectation-Maximization

Optimizing the lower bound with respect to $q(\mathbf{z})$

$$\mathcal{L}(\theta) \geq -\mathcal{D}_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x};\theta)\right] + const$$

KL-divergence is non-negative, and equals to zero when

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x};\theta)$$

Note: the calculation of $q(\mathbf{z})$ is based on current $\theta^{(t)}$

Therefore our optimal solution is simply

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x};\theta^{(t)})$$

Next, we can optimize the parameters in the M-Step

$$\theta^{(\mathbf{t+1})} = \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\log p(\mathbf{z}|\mathbf{x};\theta)\right]$$

Properties of EM

Issues with the EM algorithm

Note that the global optimal is not guaranteed

- Likelihood $\mathcal{L}(\theta) = \log p(\mathbf{x}, \mathbf{z}; \theta)$ is usually non-convex
- EM is a greedy algorithm (alternative ascent)

Moving to a more general EM formulation

- We have explicitly assumed our density as Gaussian
- However, we noted that clustering is multi-modal
- This defines **Gaussian Mixture Models (GMM)**

Probabilistic interpretation of clustering

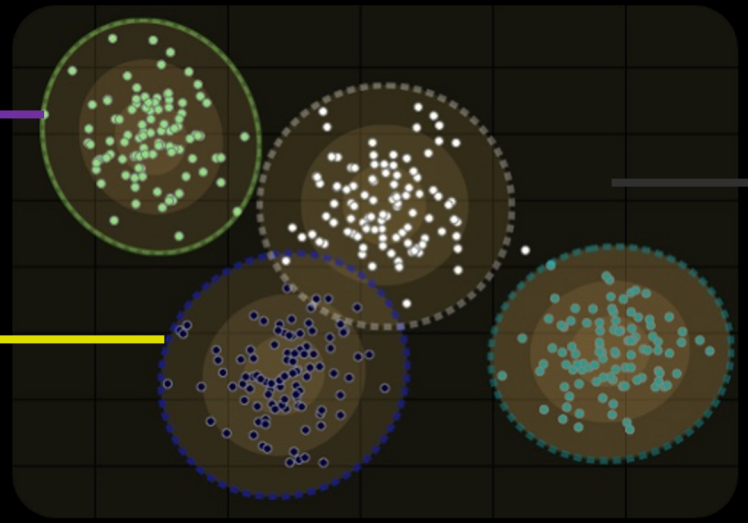
Formal definition of **probabilistic clustering**

We can *define one Gaussian distribution* per cluster

$$p(\mathbf{x}_i|C_k) = \mathcal{N}(\mu_k, \Sigma_k)$$

$p(\mathbf{x}_i|C_k = 1)$

$p(\mathbf{x}_i|C_k = 2)$



$p(\mathbf{x}_i|C_k = 3)$

$p(\mathbf{x}_i|C_k = 4)$

- However note that the full density is multi-modal
- Each mode represents a sub-population
  - Unimodal Gaussian for each group

Mixing coefficients

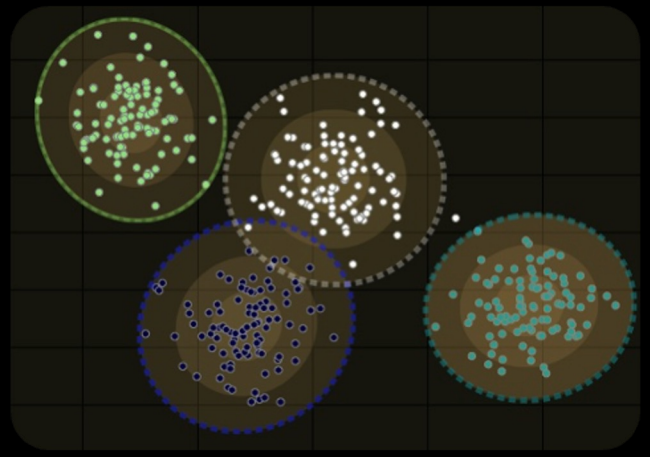
$$p(\mathbf{x}_i) = \sum_k \underbrace{p(\mathbf{x}_i|C_k)}_{\text{Unimodal}} p(C_k)$$

Mixture models

We have seen with *Maximum Likelihood* how to deal with

$$p(\mathbf{x}|\theta) = \mathcal{N}(\mathbf{x}|\mu, \mathbf{\Sigma}) \qquad \theta = \{\mu, \mathbf{\Sigma}\}$$

Now we have a more complicated **mixture** distribution



$$\begin{aligned} p(\mathbf{x}|\theta) = \pi_1 \mathcal{N}(\mathbf{x}|\mu_1, \mathbf{\Sigma}_1) \\ + \pi_2 \mathcal{N}(\mathbf{x}|\mu_2, \mathbf{\Sigma}_2) \\ + \pi_3 \mathcal{N}(\mathbf{x}|\mu_3, \mathbf{\Sigma}_3) \\ + \pi_4 \mathcal{N}(\mathbf{x}|\mu_4, \mathbf{\Sigma}_4) \end{aligned}$$

Mixing coefficients

$$\theta = \{\pi_1, \pi_2, \pi_3, \pi_4, \mu_1, \mu_2, \mu_3, \mu_4, \mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \mathbf{\Sigma}_3, \mathbf{\Sigma}_4\}$$

Mixture distribution

$$p(\mathbf{x}|\theta) = \sum_c \pi_c \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c)$$

Gaussian Mixture Model (GMM)

The full distribution is a Gaussian Mixture Model (GMM)

$$p(\mathbf{x}|\theta) = \sum_c \pi_c \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c)$$

Generally speaking, *mixture models* are defined as sums of distributions

Components of a mixture model

Base distribution *(here Gaussian)* | $p(\mathbf{x}|z=c,\theta) = \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c)$

Mixing probabilities *(here memberships)* | $p(z=c|\theta) = \pi_c$

Note this implies constraints $\pi_c \geq 0, \forall c \quad \sum_c \pi_c = 1$

Optimizing this model implies a complicated problem

$$\max_\theta p(\mathbf{x}|\theta) = \prod_{i=1}^N p(\mathbf{x}_i|\theta)$$

$$\max_\theta \prod_{i=1}^N \sum_c \pi_c \mathcal{N}(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c)$$

Optimizing GMM

As we have seen previously, we have a latent variable

$$p(z = c|\theta) = \pi_c$$

This defines *mixing probabilities*, with then

$$p(\mathbf{x}|z = c, \theta) = \mathcal{N}\left(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c\right)$$

Therefore, we can retrieve our model by marginalization

$$p(\mathbf{x}|\theta) = \sum_c p(\mathbf{x}|z = c, \theta)p(z = c|\theta)$$

If we knew the latent $p(z = c|\theta)$ optimization is easy as

$$p(\mathbf{x}|z = c, \theta) = \mathcal{N}\left(\mathbf{x}|\mu_c, \mathbf{\Sigma}_c\right) \qquad \mu_c = \frac{\sum_i p(z = c|\mathbf{x}_i, \theta)\mathbf{x}_i}{\sum_i p(z = c|\mathbf{x}_i, \theta)}$$

However, we need to optimize both the means and the latent

Goal of the Expectation-Maximization algorithm

Expectation-Maximization

EM iteratively maximizes likelihood

The complete likelihood is $\mathcal{L}(\theta) = \log p(\mathbf{x}, \mathbf{z}; \theta)$

- Start from a random initial guess $\theta^{(0)}$
- Then iteratively apply (until convergence)

E-Step

Compute the *expectation* of the likelihood

$$\mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\mathcal{L}(\theta)\right] = \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x};\theta^{(\mathbf{t})}) p(\mathbf{z}, \mathbf{x}; \theta^{(\mathbf{t})})$$

This function is called the Q-function $\mathcal{Q}(\theta; \theta^{(\mathbf{t})}) = \mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\mathcal{L}(\theta)\right]$

M-Step

Compute parameters by *maximizing* the Q-function

$$\theta^{(\mathbf{t+1})} = \underset{\theta}{\operatorname{argmax}}\ \mathcal{Q}(\theta; \theta^{(\mathbf{t})})$$

Expectation-Maximization

Optimizing the lower bound with respect to $q(\mathbf{z})$

$$\mathcal{L}(\theta) \geq -\mathcal{D}_{KL}\left[q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x};\theta)\right] + const$$

KL-divergence is non-negative, and equals to zero when

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x};\theta)$$

Note: the calculation of $q(\mathbf{z})$ is based on current $\theta^{(t)}$

Therefore our optimal solution is simply

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x};\theta^{(t)})$$

Next, we can optimize the parameters in the M-Step

$$\theta^{(\mathbf{t+1})} = \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\log p(\mathbf{z}|\mathbf{x};\theta)\right]$$

E-Step for GMM

We know the optimal approximation $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}; \theta^{(t)})$

So we simply need to compute the latent probabilities

$$\pi_{c,i} = p(z_i = c|\mathbf{x}, \theta)$$

Noting that we need to satisfy the constraint $\sum_{c=1}^{C} \pi_{c,i} = 1$

Deriving the optimal solution

$$p(z = c|\mathbf{x}_i, \theta) = \mathcal{N}(\mathbf{x}_i|\mu_c, \mathbf{\Sigma}_c)$$
$$= \frac{1}{\sqrt{(2\pi)^d \mathbf{\Sigma}_c}} \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mu_i)^T \mathbf{\Sigma}_c^{-1}(\mathbf{x}_i - \mu_c)\right]$$

Then, we can simply normalize with $\pi_{c,i} = \pi_{c,i} / \sum_{c=1}^{C} \pi_{c,i}$

M-Step for GMM

M-step seeks to optimize the parameters as

$$\theta^{(\mathbf{t+1})} = \underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{z}|\mathbf{x};\theta^{(t)}}\left[\log p(\mathbf{z}|\mathbf{x};\theta)\right]$$

Writing out the full density

$$\mathbb{E}_{\mathbf{z}}\left[\log p(\mathbf{x},\mathbf{z}|\theta)\right] = \sum_i \sum_c q(z_i = c) \log\left(\frac{1}{Z}\exp\left[-\frac{(x_i-\mu_c)^2}{2\sigma_c^2}\right]\pi_c\right)$$

$$= \sum_i \sum_c q(z_i = c)\left(\log\frac{\pi_c}{Z} - \frac{(x_i-\mu_c)^2}{2\sigma_c^2}\right)$$

Finding the maximum by deriving *one mean*

$$\frac{\partial}{\partial\mu_1} = \sum_i q(z_i=1)\left(0 - \frac{(x_i-\mu_1)(-1)}{2\sigma_1^2}\right) \quad \sum_i q(z_i=1)x_i - \sum_i q(z_i=1)\mu_1$$

Final solution

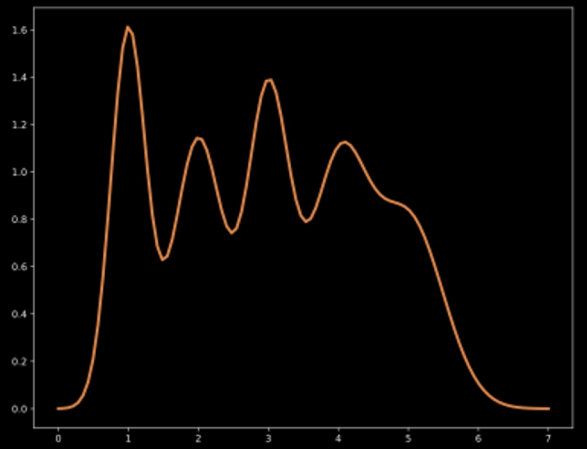
$$\mu_1 = \frac{\sum_i q(z_i=1)x_i}{\sum_i q(z_i=1)}$$

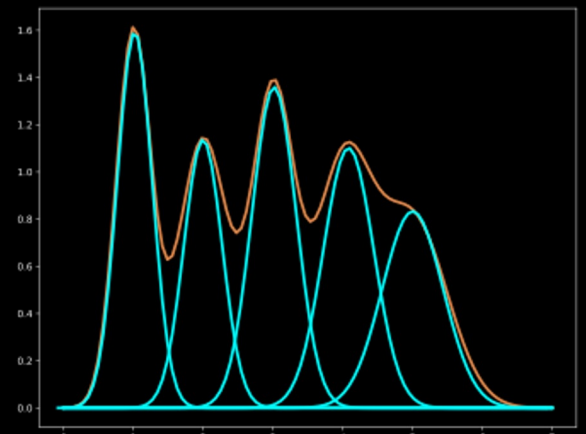
Similar derivations for all parameters and multivariate case

Mixture density estimates

GMMs are useful for *density estimation*

- Consider the following distribution
- Example of *spectral harmonics*
- Clearly *not a single Gaussian*





Mixture density estimates

- Allows to directly target
- Provides a generative model on our data
- *Problem of number of components*