Unsupervised learning

Introduction

The machine learning techniques that we studied in the previous lessons were about supervised learning, meaning that we assumed that our training datasets were labelled.

When dealing with real problems, many times we have to face situations were we only have unlabelled or partially labelled samples, this problems belong to the discipline of unsupervised learning. Hence, what we want is to find a funcition approximation $f : X \to Y$ by having a dataset that contains unlabelled data $D = \{(x_n)_{n=1}^N\}$.

In the next chapters we will see different techniques for unsupervised learning.

Gaussian Mixture Models

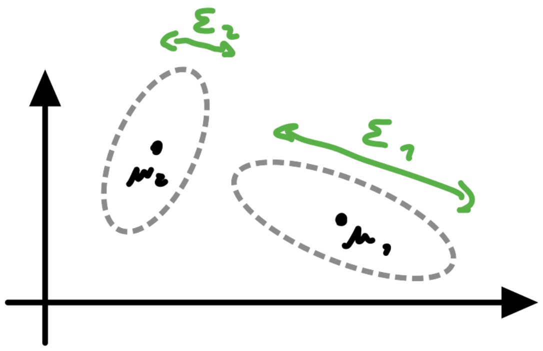
The core idea behind Gaussian Mixture Models (GMM) is that we assume that our data was generated from a certain probability distribution $P(x)$ expressed as a weighted sum of k Gaussians:

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k)$$

Where:

- π_k is the weight of the k-th Gaussian (also called the prior). The higher the weight, the higher is the probability that data was generated from that specific Gaussian.
- μ_k is the mean of the k-th Gaussian.
- Σ_k is the covariance matrix of the k-th Gaussian.

Given all of these parameters, it's easy to generate a new dataset by sampling data points from the distribution.

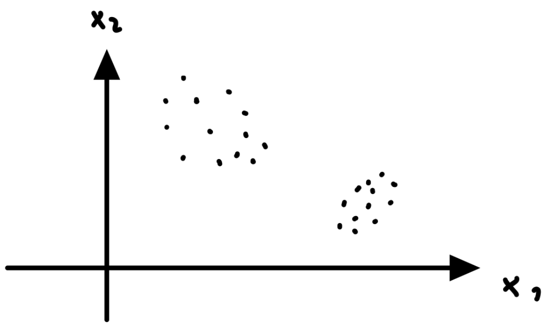


Once we have generated our data, we can forget about their parameters of the original distribution and try to learn them by starting from scratch, given that the number of distributions k is known.

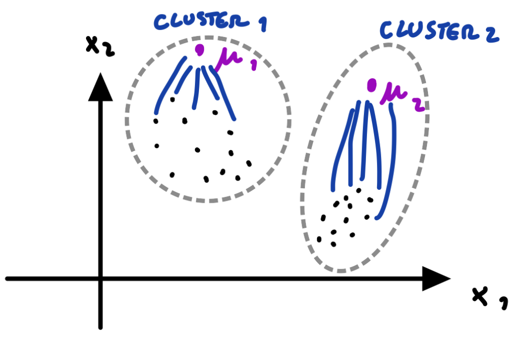
K-Means

We start by assuming that we know that the number of Gaussians k is known and that both covariance and priors are the same for each distribution.

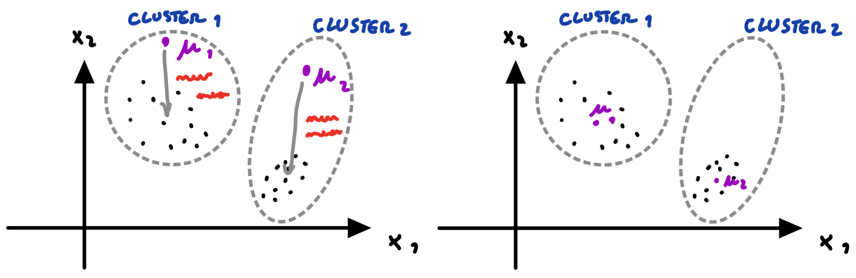
Our goal is to find the $\mu_1, ..., \mu_k$ means given the dataset and k. We will start from a set of unlabelled data like:



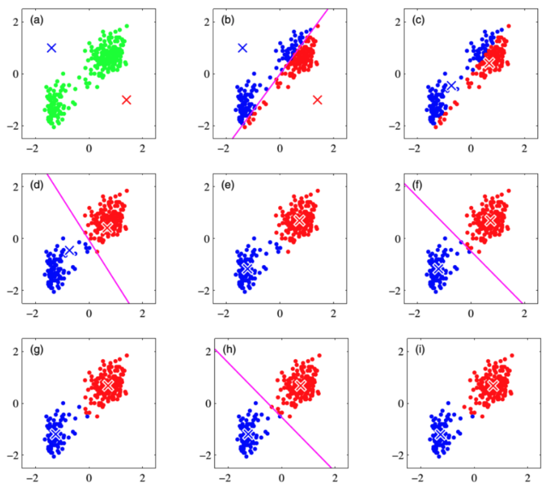
K-means is an iterative algorithm. The first step is to initialize the k means in two possible ways: either we choose a totally random value, or we choose a random point of the dataset. Given these means, we create k clusters based on the minimum distance between every point and the various means. The point will belong to a certain cluster if the distance between it and the cluster's mean is the closest:



We can now repeat the computation of the mean by computing the center of mass of each cluster, which corresponds to finding the average of the values for each cluster:



After computing the new means, we re-clusterize based on the same criterion of the minimum distance. The algorithm will iteratively repeat these operation until the we won’t have any changes in assignment between one step and the previous one.



Example of the execution of k-means.

The output of this algorithm will correspond to the computed means.

Problems of k-means

- We depend on the right guess of k. Having a different number of means with respect to the actual value will result at best only in a partial representation of the dataset.
- Outliers influence negatively the computation by attracting the mean towards them.
- An important design choice is the defintion of a proper distance function, since k-means does not consider the covariances of distribution.

Latent variables

Latent variables are a special kind of variables that are useful to describe our problem, but are not given by the dataset. They help us in better defining the formulation of our problem.

When we analyze samples we don't know from which of the k distributions they come from. Hence, we can think about introducing the boolean variables $z_k \in \{0, 1\}$ that will represent this notion. If $z_k = 1$ means that the sample comes from the k-th distribution, 0 otherwise. The informations about all of these latent variables can be econded using a 1-out-of-k approach: $Z = (z_1, ..., z_k)^T$.

We define the probability of belonging to a certain distribution as the posterior $P(z_k = 1) = \pi_k$

Thus, $P(Z) = \prod_{k=1}^{K} \pi_z^{z_k}$ (the likelihood of that vector is given by the products of the posteriors elevated to the value of the latent variable. Notice that only one latent varable has value 1).

Given Z, we compute the conditional probability of x given Z as:

$$P(x|Z) = \mathcal{N}(x; \mu_k, \Sigma_k)$$

Notice that if we know Z, then the probability only depends on the k-th Gaussian. Given the previous considerations, we can express the posterior as:

$$P(x|Z) = \prod_{k=1}^{N} \mathcal{N}(x; \mu_k, \Sigma_k)^{z_k}$$

The total probability of x will be:

$$P(x) = \sum_{Z} P(Z)P(x|Z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k)$$

We obtained the formulation of a Gaussian Mixture Model!

A fundamental assumption that we can derive is that a Gaussian Mixture Model can be expressed as the marginalization of the joint probabilities $\sum_Z P(Z)P(x|Z)$.

Our problem now is that we don't actually know the value of the latent variables.

We can now define the posterior probability, meaning the probability that a certain data comes from a certain distribution as:

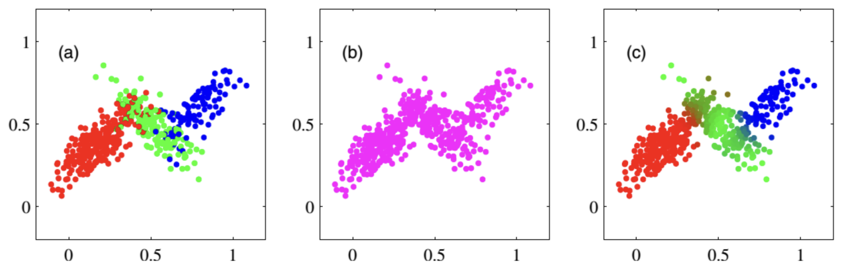
$$\gamma(z_k) = P(z_k = 1|x) = \frac{P(z=1)P(x|z=1)}{P(x)}$$

From where we obtain:

$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(x; \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_k \mathcal{N}(x; \mu_j, \Sigma_j)}$$

Where the denominator $\sum_{j=1}^{K} \pi_k \mathcal{N}(x; \mu_j, \Sigma_j)$ is called normalization layer.

The following is the representation of what happens when using Gaussian Mixture Models on an unlabelled dataset:



- Image (a) represents the ground truth, that we don't know when dealing with the problem.
- Image (b) shows the unlabelled dataset, that we have as input.
- Image (c) shows the resul of applying Gaussian Mixture Model on the unlabelled dataset. Notice that the colors encode the posterior probability.

Computing the posterior: Expectation Minimization

While before we started with the assumption that both covariance and prior are the same for every distribution, now we are interested in determing all the three parameters μ_k, Σ_k, π_k.

In particular, we aim at computing the maximum likelihood:

$$argmax_{\pi,\mu,\Sigma}\ ln(P(x|\pi,\mu,\Sigma))$$

The computed solution for this problem is:

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(x_{nk})x_n$$
$$\Sigma_k = \frac{1}{N_k}\gamma(z_{nk})(x_n-\mu_k)(x_n-\mu_k)^T$$
$$\pi_k = \frac{N_k}{N}$$

Where $N_k = \sum_{n=1}^{N}\gamma(z_{nk})$.

Notice that the solution depends on the dataset $x_n$on the posterior $\gamma(z_{nk})$. We still have to determine z.

At this point, if we assume to know all the other parameters, we can compute the postirior of z.

Expectation maximization exploits the formulas that we derived, by implementing them into an iterative two-steps algorithm that is repeated until convergence. These two steps are:

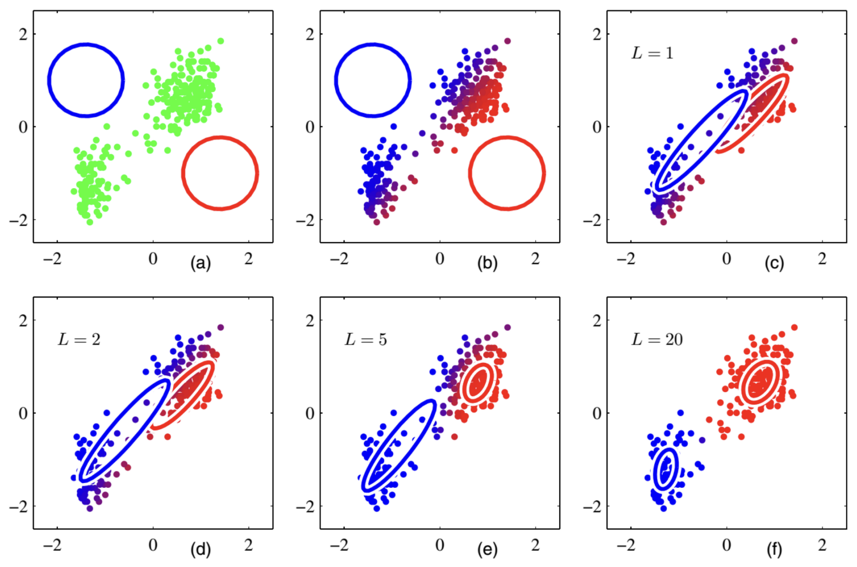
- **Expectation:** given the parameters of the distribution, we computer the posterior of z.
- **Maximization:** given the prediction of the posterior, we compute the parameters of the mode.

The algorithm begins by initializing the parameters $\mu_k^{(0)}, \Sigma_k^{(0)}, \pi_k^{(0)}$ at step 0 with random values.

We start from the expectation step E, where we compute the posterior given the parameters.

The second step is maximization M, where we do the opposite by taking the previously computer posterior and compute the new parameters.

E and M steps are repeated until convergence. This method will output all the parameters.



In order to simplify the notation, we can denote the parameters as $\theta = \mu, \Sigma, \pi$ and the updated parameters as θ'. Now, we can represent the function that updates the parameters as $Q(\theta'|\theta)$.

The EM algorithm can be generically defined as:

E step:

Compute $Q(\theta'|\theta)$ using the current hypothesis on the parameters θ and the observed data X in order to estimate the pobability distribution over $Y = X \cup Z$:

$$Q(\theta'|\theta) \leftarrow E[lnP(Y|\theta')|\theta, X]$$

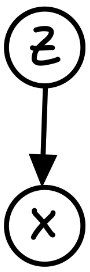
M step:

Update the current hypothesis to θ' by maximizing the Q function:

$$\theta \leftarrow argmax_{\theta'} Q(\theta'|\theta)$$

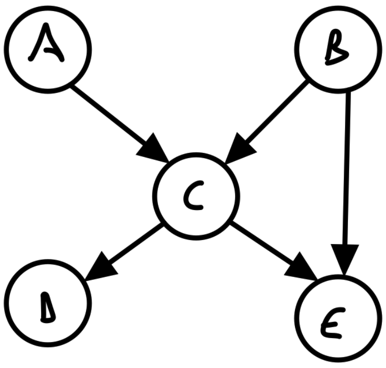
Bayesian Networks

Bayesian networks are basically a graphical model to express the conditional probabilities between two or more variables. For example, we can express $P(X|Z)$ as follows:



Notice that there are two entities in bayesian networks: nodes represent the random variables, while directed edges represent the dependence of a child from its parent\s.

Bayesian networks can grow by representing the relationtip of many variables, like the following example:



Bayesian networks are a particular case of a broader class of graphs called Probabilistic Graphical Models (PGM).

Starting from a graphical representation, we can derive a table of conditional probabilities to represent the dependencies between all the random variables in the form of tabular data.

Let's suppose that we have $B, C \in \{0, 1\}$. The corresponding tabular representation will be something like this:

$P(E|B) =$

E \ B	0	1
0	0,7	0,8
1	0,3	0,2

the total sum of the columns must always be 1

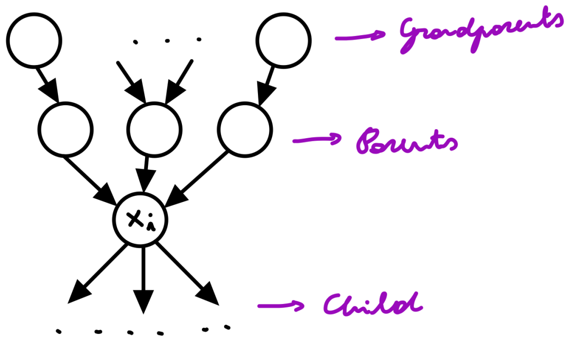
We can express these tables in a more compact way by deleting the last row, since its values can be derived from theprevious ones.

The same can be done with multiple variables, like in this example:

$P(C|A,B) =$

	A=0		A=1	
C	B=0	B=1	B=0	B=1
0	0.3	0.6	0.2	0.3
1	–	–	–	–

Given a certain node x_i, we call parents all the nodes that are directly connected to it (and with the arrow pointing towards x_i). Bayesian networks have the property that every node is conditionally dependent only from its parents, meaning that a node x_i will be independent from all the nodes that are not his parents. This means that we will express the conditional probability of a node only considering its parents.



We can represent the property of conditional independence for bayesian networks as follows:

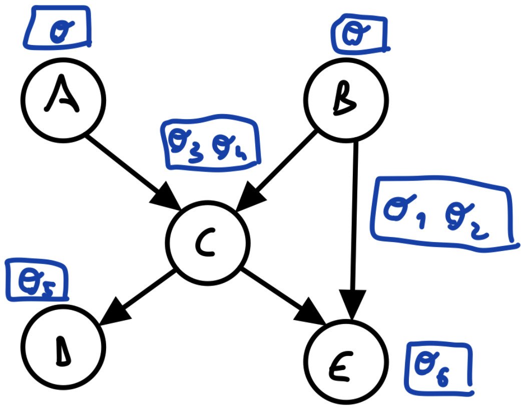
$$P(x_i|Parent(x_i)) = P(P(x_i|Parent(x_i), Z)$$

This property makes us able to represent complex problems in a very compact way.

When computing probabilities involving many variables, we can simpy exploit the chain rule and the probability tables:

$P(A, B, C, D, E) = P(A|B, C, D, E) \cdot P(B|C, D, E)...$

We can express a parametric model of bayesian networks by assigning parameters to each relation:



θ
A
θ
B
$\theta_3\ \theta_4$
$\theta_1\ \theta_2$
C
θ_5
D
E
θ_6