*K-Means *

Ref : http://ethen8181.github.io/machine-learning/clustering/GMM/GMM.html#Assessing-Convergence

Ref : https://krasserm.github.io/2019/11/21/latent-variable-models-part-1/

Ref : https://github.com/Adaickalavan/Machine-Learning-CSMM102x-John-Paisley-Columbia-University-EdX/blob/master/Week%209%20Clustering/hw3_clustering.py

Ref : Pattern Recognition and Machine Learning , Bishop. Springer Publication

1. K Means Clustering

> We begin by considering the problem of identifying groups, or clusters, of data points in a multidimensional space. Suppose we have a data set {x1, . . . , xN} consisting of N observations of a random D-dimensional Euclidean variable x. Our goal is to partition the data set into some number K of clusters, where we shall suppose for the moment that the value of K is given. Intuitively, we might think of a cluster as comprising a group of data points whose inter-point distances are small compared with the distances to points outside of the cluster. We can formalize this notion by first introducing a set of D-dimensional vectors μ_k where k = 1, . . . , K, in which μ_k is a prototype associated with the kth cluster. As we shall see shortly, we can think of the μ_k as representing the centres of the clusters. Our goal is then to find an assignment of data points to clusters, as well as a set of vectors {μk}, such that the sum of the squares of the distances of each data point to its closest vector μ_k, is a minimum.

> Let us define some notation to describe the assignment of data points to clusters. For each data point xn, we introduce a corresponding set of binary indicator variables $r_{n\,k}$ ∈ {0, 1}, where k = 1, . . . , K describing which of the K clusters the data point x_n is assigned to, so that if data point xn is assigned to cluster k then $r_{n\,k}$ = 1, and $r_{n\,k}$ = 0 for j = k. This is known as the 1-of-K coding scheme. We can then define an objective function, sometimes called a distortion measure, given by
> $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$

> J represents the sum of the squares of the distances of each data point to it assigned vector μ_k. μk. Our goal is to find values for the r_{nk} and the μ_k so as to minimize J.
>
> - We can do this through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the r_{nk} and the μ_k.
> - First we choose some initial values for the μ_k. Then in the first phase we minimize J with respect to the r_{nk}, keeping the μ_k fixed.
> - In the second phase we minimize J with respect to the μ_k , keeping r_{nk} fixed.
> - This two-stage optimization is then repeated until convergence.

> **We shall see that these two stages of updating r_{nk} and updating μ_k correspond respectively to the E (expectation) and (maximization) steps of the EM algorithm, and to emphasize this we shall use the terms E step and M step in the context of the K-means algorithm.**

2. Mixture of Gaussians

> We now turn to a formulation of Gaussian mixtures in terms of discrete latent variables. This will provide us with a deeper insight into this important distribution, and will also serve to motivate the expectation-maximization algorithm.

> The Gaussian mxture distributions can be wrtten as a linear superposition of Gaussians in the form :

- $p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$ ----eqn 1

> Let us introduce a K-dimensional binary random variable z having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy z_k ∈ {0, 1} and $\Sigma_k z_k = 1$, and we see that there are K possible states for the vector z according to which element is nonzero.

> We shall define the joint distribution p(x, z) in terms of a marginal distribution p(z) and a conditional distribution p(x|z), corresponding to the graphical model.

> The marginal distribution over z is specified in terms of the mixing coefficients π_k, such that :

- $p(z_k = 1) = \pi_k$

> where the parameters $\{\pi_k\}$ must satisfy $0 \leq \pi_k \leq 1$ -----eqn2 together with $\Sigma_{k=1}^{K}\pi_k = 1$ ----eqn3 in prder to be valid probabilities. Because *z* uses 1-of-K reepresentation, we can also write this distribution in the following form :
>
> - $p_z = \Sigma_{k=1}^{K}\pi_k^{zk}$ -- eqn 4

> Similarlly, the conditional distribution of x given aparticular value of z is a Gaussian
>
> - $p(x,z) = \prod_{k=1}^{K}\mathcal{N}(x|\mu_k,\Sigma_k)$ -- eqn 5

> The joint distribution is given by p(z)p(x|z), and **the marginal distribution of x** is then obtained by **summing the joint distribution over all possible states of z** to give
>
> - $p(x) = \Sigma_z p(z)P(x|z) = \Sigma_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)$ ---- eqn 6

> Thus the marginal of x is is a Gaussian mixture of the form define above. If we have several observations $x_1,\ldots,x_N$, then, because we have represented the marginal distribution in the form $p(x) = \Sigma_z p(x,z)$, it follows that for every observed data point x_n there is a corresponding latent variable z_n.

> We have therefore found an equivalent formulation of the Gaussian mixture involving an explicit latent variable. It might seem that we have not gained much by doing so. However, we are now able to work with the joint distribution $p(x,z)$ instead of the marginal distribution $p(x)$, and this will lead to significant simplifications, most notably through the introduction of the expectation-maximization (EM) algorithm.

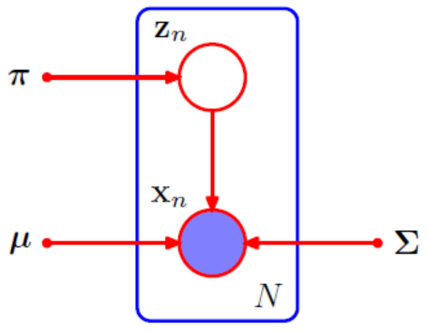
> Another quantity that will play an important role is the conditional probability of z given x. We shall use $\gamma(z_k)$ to denote $p(z_k = 1|x)$, whose value can be found using Bayes' theorem

- $\gamma(z_k) = p(z_k = 1|x) = \frac{p(z_k=1)p(x|z_k=1)}{\Sigma_{j=1}^{K}p(z_j=1)p(x|z_j=1)}$
- $\gamma(z_k) = p(z_k = 1|x) = \frac{\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)}{\Sigma_{j=1}^{K}\pi_j\mathcal{N}(x|\mu_j,\Sigma_j)}$

> We shall view π_k **as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability** once we have observed x. As we shall see later, $\gamma(z_k)$ ***can also be viewed as *the responsibility that component k takes for 'explaining' the observation x.***

---

3. Maximum Likehood

> Suppose we have a data set of observations $\{x_1, \dots, x_N\}$, and we wish to model this data using a mixture of Gaussians. We can represent this data set as an N × D matrix X in which the nth row is given by x_n^T . Similarly, the corresponding latent variables will be denoted by an N × K matrix Z with rows z_n^T . If we assume that the data points are drawn independently from the distribution, then we can express the Gaussian mixture model for this i.i.d. data set using the graphical representation as shown inthe image below
>
> Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{\mathbf{x}_n\}$, with corresponding latent points $\{\mathbf{z}_n\}$, where $n = 1, \dots, N$.
>
> 
>

> The log Likelihood function from the eqn 1 is given by

- $lnp(X|\pi, \mu, \Sigma) = \Sigma_{n=1}^N ln\{\sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k)\}$ ---- eqn 8

> Before discussing how to maximize this function, it is worth emphasizing that there is a significant problem associated with the maximum likelihood framework applied to Gaussian mixture models, due to the presence of singularities. For simplicity, consider a Gaussian mixture whose components have covariance matrices given by $\Sigma_k = \sigma_k^2 I$, where I is the unit matrix, although the conclusions will hold for general covariance matrices.

> There are issues of singularity and severe Overfitting. We shall see that this difficulty does not occur if we adopt a Bayesian approach. For the moment,however, we simply note that in applying maximum likelihood to Gaussian mixture models we must take steps to avoid finding such pathological solutions and instead seek local maxima of the likelihood function that are well behaved. We can hope to avoid the singularities by using suitable heuristics, for instance by detecting when a Gaussian component is collapsing and resetting its mean to a randomly chosen value while also resetting its covariance to some large value, and then continuing with the optimization.

> **A further issue in finding maximum likelihood solutions arises from the fact that for any given maximum likelihood solution, a K-component mixture will have a total of**

> **K! equivalent solutions corresponding to the K! ways of assigning K sets of parameters to K components.**

> Maximizing the log likelihood function for a Gaussian mixture model turns out to be a more complex problem than for the case of a single Gaussian. The difficulty arises from the presence of the summation over k that appears inside the logarithm in eqn 8, so that the logarithm function no longer acts directly on the Gaussian. If we set the derivatives of the log likelihood to zero, we will no longer obtain a closed form solution.

4. EM for Gaussian Mixtures

> An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the expectation-maximization algorithm, or EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997)

> - EM can be generalized to obtain the variational inference framework.

> Let us begin by writing down the conditions that must be satisfied at a maximum of the likelihood function. Setting the derivatives of $lnp(X|\pi, \mu, \Sigma)$ in (9.14) with respect to the means μ_k of the Gaussian components to zero, we obtain

EM for the GMM

> **Algorithm** : Maximum Likelihood EM for GMM.

> **Given** : $x_i, \ldots\ldots, x_n$ where $x \in \mathbb{R}^d$

> **Goal** :Maximize the LOWER Bound L = $\sum_{i=1}^{n} lnp(x_i|\pi, \mu, \Sigma)$

> Iterate until L is "$small$"

> - **E-step** : For $i = 1, \ldots\ldots, n$ set

> - $\phi_i(k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)}$

> Recall that **GMM's goal** is to output a set of soft assignments per data point (allocating the probability of that data point belonging to each one of the clusters). To begin with, let's just assume we actually know the parameters π_k, μ_k **and** $\sum_k$ **(from**

> **some random initialization)** and we need a formula to compute the soft assignments having fixed the values of all the other parameters.

> The soft assignments are quantified by the responsibility vector ϕ . For each observation i , we form a responsibility vector with elements ϕ_1(1), ϕ_1(2) all the way up to ϕ_1(K). Where K is the total number of clusters, or often referred to as the number of components. The cluster responsibilities for a single data point i should sum to 1.

> In order to model more complex data distribution, we can use a linear combination of several Gaussians instead of using just one. To compute the mixture of Gaussians, we introduce a set of cluster weights,π_k , one for each cluster k where $\sum_i^K \pi_k = 1$ and $0 \leq \pi_k \leq 1$ (meaning that the sum must add up to one and each of them is between 0 and 1). This parameter tells us what's the prior probability that the data point in our data set x comes from the kth cluster. We can think it as controlling each cluster's size.

> The next part of the equation, $N(x_i|\mu_k, \Sigma_k)$ tells us the following : Given that we knew that the observation comes from the k^{th} cluster, what is the likelihood of observing our data point x_i coming from this cluster. To compute this part, the scipy package provides a convenient function multivariate_normal.pdf that computes the likelihood of seeing a data point in a multivariate Gaussian distribution.

> After multiplying the prior and the likelihood, we need to normalize over all possible cluster assignments so that the responsibility vector becomes a valid probability. And this is essentially the computation that's done for the E step.

> - **M-step** : For $k = 1, \ldots, K$, define $n_k = \sum{}_{i=1}{}^n \phi_i(k)$

and update the values as follows :

> - $\pi_k = \frac{n_k}{n}$,

> - $\mu_k = \frac{1}{n_k} \sum{}_{i=1}{}^n \phi_i(k)$ and

> - $\Sigma_k = \frac{1}{n_k} \sum{}_{i=1}{}^n \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T$

- NOTE : The update value of μ_k is used for updating Σ_k

> First, the cluster weights π_k , show us how much each cluster is represented over all data points (each cluster's relative size). This weight is given by the ratio of the soft count n_K over the total number of data points N .

> When updating our parameters' estimates for each cluster k , we need to account for the associated weights $\phi_i(k)$ for every one of our observation. So every time we're touching a data point x_i it's going to be multiplied by $\phi_i(k)$.

> Another thing that's worth noticing is that, when we're updating the parameter μ_k and Σ_k , instead of dividing the summation with the raw count of the total number of data points in that cluster N , we will use the effective number of observations in that cluster (the sum of the responsibilities in that cluster) as the denominator. This is denoted as $n_k = \sum_{i=1}{}^{n} \phi_i(k)$

Assesing CONVERGENCE of the EM algorithm

> Apart from training the model, we also want a way to monitor the convergence of the algorithm. We do so by computing the log likelihood of the data given the current estimates of our model parameters and responsibilities.

> During the E-step we used the follwing formula to calculate the weighted probability of every data point x_i coming form the cluster j and summed up all the weighted values.

> - $\sum_j \pi_j N(x_i|\mu_j, \Sigma_j)$

> If we were to assume the observed data points were generated independently, the likelihood of the data can be written as :
>
> - $p(x|\pi, \mu, \Sigma) = \prod_{n=1}{}^{N} \sum_{j=1}{}^{K} \pi_j N(x_i|\mu_j, \Sigma_j)$

> This basically means that we multiply all the probability for every data point together to obtain a single number that estimates the likelihood of the data fitted under the model's parameter. We can take the log of this likelihood so that the product becomes a sum and it makes the computation a bit easier:

> - $ln(p(x|\pi, \mu, \Sigma)) = \sum_{i=1}{}^{N} ln\{\sum_{j=1}{}^{K} \pi_j N(x_i|\mu_j, \Sigma_j\}$

> **If the log likelihood of the data occuring under the current model's parameter does not improve by a tolerance value that we've pre-specified, then the algorithm is deemed converged.**