GMM

Shivam Malhotra (190808) & Naiza Singla (190523)

April 3, 2022

1 K Means Clustering

The K-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. The K-means algorithm tries to minimize distortion for samples $x_i \in X$„ which is defined as:

$$\Gamma = \sum_{n,k} r_{nk} \parallel x_n - \mu_k \parallel^2 \tag{1}$$

where k is the cluster index, n is the sample index, r_{nk} is the assignment measure i.e. r_{nk} is 1 if $x_k \in C_k$ else 0, where C_k is the k^{th} cluster.

This distortion measure tries to minimize the possibility of a single large clusters or multiple very small clusters. In the case of a very large cluster, $\parallel x_k - \mu_k \parallel$ will be a large value for the large cluster, and thus it's summation would give an even larger value (distortion).

1.1 Optimization Algorithm

The K-means clustering algorithm uses iterative refinement to produce a final values for the variables r_{nk} and μ_k. The algorithms starts with initial estimates for the K centroids, which can either be randomly generated or randomly sampled from the given data set. The algorithm then iterates between two steps:

1.1.1 Centroid Update Step

In this step, the centroids are recomputed, given the values of r_{nk}. This can be done by minimizing the distortion measure with respect to the values of μ_k.

$$\frac{\partial \Gamma}{\partial \mu_k} = 2 \sum_n r_{nk} \cdot \parallel x_n - \mu_k \parallel = 0 \tag{2}$$

$$\therefore \mu_k = \frac{\sum_n r_{nk} \cdot x_n}{\sum_n r_{nk}} \tag{3}$$

Thus, μ_k corresponds to mean of samples in the k^{th} cluster.

1.1.2 Assignment Step

In this step, each data point is assigned to a cluster, i.e. given the values of μ_k, we need to compute the assignment labels (r_{nk}) for each data point. Each data point is assigned to its nearest centroid, based on the squared Euclidean distance. Formally,

$$r_{nk} = \begin{cases} 1, & \text{if } k = \operatorname{argmin}_{k'} \parallel x_n - \mu_{k'} \parallel \\ 0, & \text{otherwise} \end{cases}$$

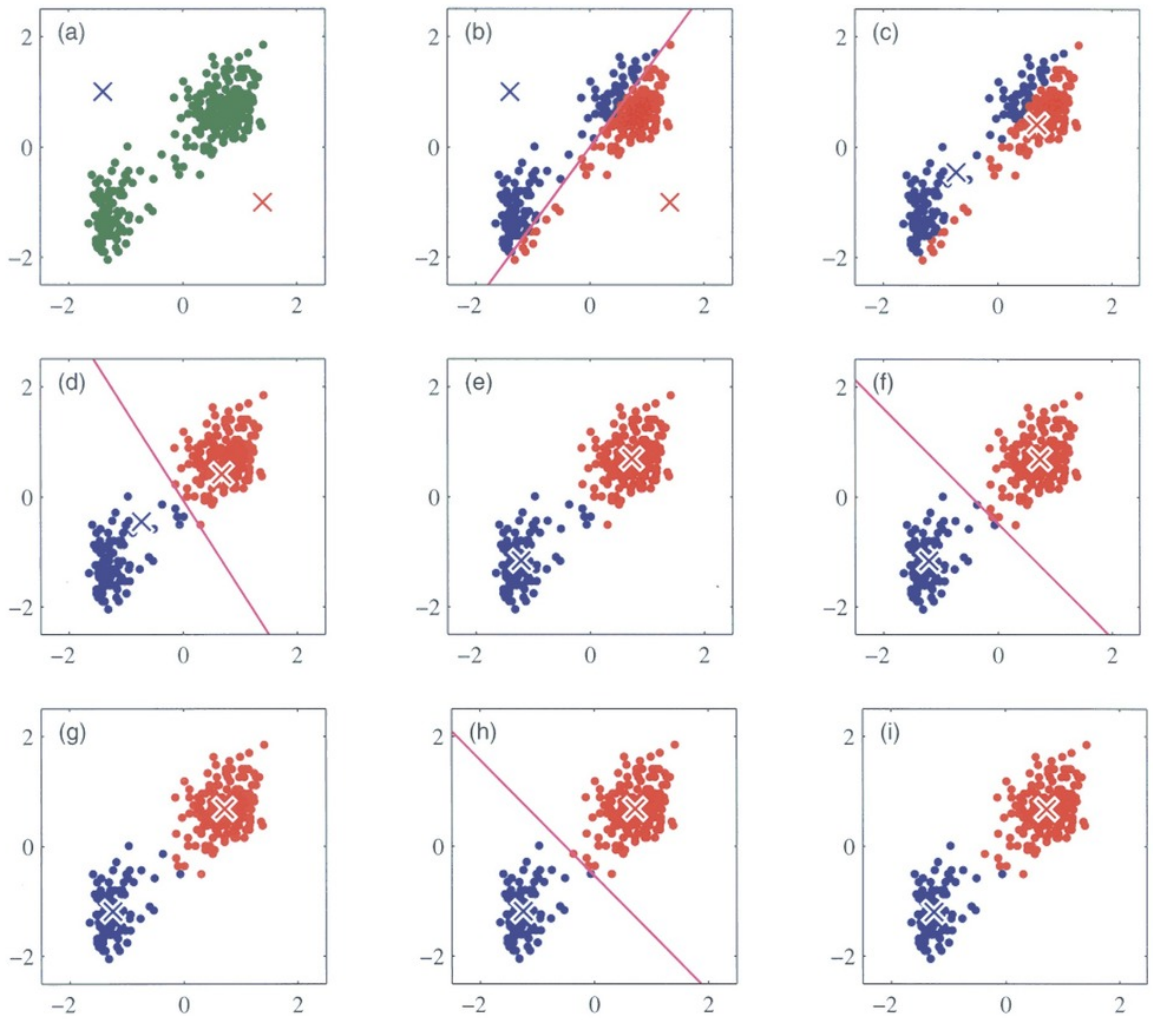

Figure 1: Step wise iterations of the K-Means Optimization Algorithm.

Source: Bishop 2006

2 Gaussian Mixture Models

2.1 Latent Variables

In statistics, latent variables, opposed to observable variables are variables that are not directly observed but are rather inferred from other variables that are observed (directly measured).

Latent variables are useful in modeling non standard distributions. Let us say we want to estimate a distribution $p(x)$ which does not follow any standard distribution known to us, then we can express the distribution using a set of latent variables $z \in Z$ denoting a cluster:

$$p(x) = p(x|z=1) \cdot p(z=1) + p(x|z=2) \cdot p(z=2) \ldots \tag{4}$$

$$p(x) = \sum_{z} p(x|z)p(z) \tag{5}$$

Here $p(z=i)$ is the probability that z belongs to the i^{th} cluster, and $p(x|z=i)$ is the conditional probability of x, given that z belongs to the i^{th} cluster. Note that here $p(x|z)$, can be any distribution like gaussian, normal, exponential etc. and z is a discrete random variable. We can model each distribution separately i.e. $p(x|z=i)$ can be different for different i. We can also represent z as a one hot encoded vector i.e. $z_i = 1$ if z corresponds to cluster i.

2.2 Modelling Gaussian Mixture Models

Using the above concepts, we can now model a GMM using a set of latent variables z as follows:

$$p(x) = \sum_k \pi_k \cdot \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k}) \tag{6}$$

Here π_k represent the probability that $z_k = 1$ (one-hot encoding) , and each cluster is modeled as a multi-dimensional multi-variate normal distribution mean (μ), and variance (Σ).

2.3 Estimation of Parameters (MLE)

Given N samples s_n and k clusters, we now want to estimate the paramaters μ_k and Σ_k for each of the k clusters. k is a hyperparameter and we must decide it manually based on intuition from the data. We have the following learnable parameters for each cluster:

- π: the probability of the cluster.
- μ: mean of the normal distribution of the cluster.
- Σ: variance of the normal distribution of the cluster.

We use the MLE Algorithm to find the optimal values for the learnable parameters:

2.3.1 MLE for $\mu_\mathbf{k}$

$$\ln \mathcal{L} = \sum_n \ln p(s_n|\theta) \tag{7}$$

$$= \sum_n \ln(\sum_k \pi_k \cdot \mathcal{N}(\mathbf{s_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k})) \tag{8}$$

Differentiating with respect to μ_k, we obtain :

$$\frac{\partial \ln \mathcal{L}}{\partial \mu_{k'}} = \sum_n \frac{\pi_{k'} \cdot \mathcal{N}(\mathbf{s_n}|\mu_{\mathbf{k'}}, \mathbf{\Sigma_{k'}})}{\sum_k \mathcal{N}(\mathbf{s_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k})} \cdot (-\Sigma_{k'}^{-1}(s_n - \mu_{k'})) = 0 \tag{9}$$

$$= \sum_n \frac{\pi_{k'} \cdot \mathcal{N}(\mathbf{s_n}|\mu_{\mathbf{k'}}, \mathbf{\Sigma_{k'}})}{\gamma_{nk'}} \cdot (-\Sigma_{k'}^{-1}(s_n - \mu_{k'})) = 0 \tag{10}$$

Here γ_{nk} represent the probability that $z_k = 1$ for given s_n i.e. the probability that s_n belongs to the $\mathrm{k^{th}}$ cluster. Upon solving we get:

$$\sum_n \gamma_{nk}\mu_k = \sum_n \gamma_{nk} s_n \tag{11}$$

$$\therefore \mu_k = \frac{\sum_n \gamma_{nk} s_n}{\mathrm{N}_k} \tag{12}$$

where $\mathrm{N_k}$ ($\sum_n \gamma_{nk}$) is the number of samples assigned to a cluster.

2.3.2 MLE for $\mathbf{\Sigma_k}$

Differentiating $\ln \mathcal{L}$ with respect to Σ_k:

$$\frac{\partial \ln \mathcal{L}}{\partial \Sigma_{k'}} = \sum_n \gamma_{nk'} [\frac{-1}{2|\Sigma_{k'}|^2} \cdot \frac{\partial |\Sigma_{k'}|}{\underbrace{\partial \Sigma_{k'}}_{|\Sigma_{k'}|\Sigma_{k'}^{-1^T}}} + \frac{1}{|\Sigma_{k'}|} \cdot \frac{\partial \overbrace{((s_n - \mu_{k'})^T)\Sigma_{k'}^{-1}(s_n - \mu_{k'})}^{-(\Sigma_{k'}^{-1^T})(s_n-\mu_{k'})(s_n-\mu_{k'})^T\Sigma_{k'}^{-1^T}}}{\partial \Sigma_{k'}}] \tag{13}$$

$$\sum_n \gamma_{nk'} [\Sigma_{k'}^{-1^T} - (\Sigma_{k'}^{-1})(s_n - \mu_{k'})(s_n - \mu_{k'})^T \Sigma_{k'}^{-1^T}] = 0 \tag{14}$$

$$\Sigma_{k'} = \frac{1}{\mathcal{N}_{k'}} \sum_n \gamma_{nk'} (s_n - \mu_{k'})(s_n - \mu_{k'})^T \tag{15}$$

2.3.3 MLE for $\pi_{\mathbf{k}}$

Differentiating $\ln \mathcal{L}$ with respect to π_k and using the constraint that $\sum_k \pi_k = 1$, we get:

$$\frac{\partial(\ln \mathcal{L} + \lambda(\sum_k \pi_{k'} - 1))}{\pi_{k'}} = \sum_n \frac{\mathcal{N}(\mathbf{s_n}|\mu_{\mathbf{k'}}, \mathbf{\Sigma_{k'}})}{\sum_k \mathcal{N}(\mathbf{s_n}|\mu_{\mathbf{k}}, \mathbf{\Sigma_k})} + \lambda = 0 \tag{16}$$

Multiplying both sides with $\pi_{k'}$, and summing over k, we obtain :

$$\sum_n 1 + \lambda = 0 \tag{17}$$

$$\lambda = -\mathcal{N} \tag{18}$$

Substituting back, we obtain:

$$\sum_n \frac{\gamma_{nk'}}{\pi_{k'}} - N = 0 \tag{19}$$

$$\therefore \pi_{k'} = \frac{\mathcal{N}_k}{\mathcal{N}} \tag{20}$$

2.3.4 Final Algorithm

Since none of the MLE estimates is independent of each other, we follow an iterative approach similar to K-means clustering. The algorithm steps can be detailed as follows:

- Initialize all trainable parameters to random values.
- Assignment Step: Calculate γ_{nk} for each s_n.
- Update Step: Update μ_k and Σ_k.
- Stop if converged else go to second step.

2.4 Expectation Maximization Algorithm

In most optimization problems, we have to find optimal θ, given $\mathcal{L}$. We can employ the following strategy:

If $\frac{\partial \mathcal{L}}{\partial \theta} = 0$, we can simply solve for θ. Otherwise, if θ are independent, we can use iterative update approach. Note that this approach only works if $\frac{\partial \mathcal{L}}{\partial \theta}$ is solvable. If $\frac{\partial \mathcal{L}}{\partial \theta}$ is not solvable, we can use gradient descent algorithm. There is an alternative approach if we do not wish to use gradient descent algorithm. We explain a new approach that works well for latent variable models.

Given, a latent variable model, we need to optimize:

$$\ln p(s|\theta) = \ln(\sum_z p(s, z|\theta)) \tag{21}$$

$$= \ln(\sum_z p(z|\theta) \cdot p(x|z, \theta)) \tag{22}$$

Due to marginalization of probability over z, we now have a summation term in logarithm, which is tricky to solve, although it was possible in case of GMM. To get rid of marginalization, we assume a value of z, and later average over all z and define a new loss function:

$$\mathcal{Q} = \underbrace{\sum_z p(z|s, \theta^{\text{old}})}_{\text{average}} \ln(\underbrace{p(s, z|\theta)}_{exponential}) \tag{23}$$

$$= \mathcal{E}_{z|s,\theta^{\text{old}}}[\ln p(s, z|\theta)] \tag{24}$$

Note that when differentiation we assume θ^{old} to be constant, otherwise we will lead into the same problem of summation we have been trying to avoid. We can use an iterative method to optimize θ with the new loss function:

- Initialize all trainable parameters to random values.
- Assignment Step: Calculate γ_{nk} for each s_n.
- Update Step: Update μ_k, Σ_k, π_k using $\mathcal{Q}$.
- Stop if converged else go to second step.

2.4.1 EM Algorithm for GMM

We can use the EM algorithm to define a new loss function for GMM as follows:

$$p(s, z|\theta) = \prod_n \prod_k \{\pi_k \mathcal{N}(s_n|\mu_k, \Sigma_k)\}^{z_{nk}} \tag{25}$$

$$\ln p(s, z|\theta) = \sum_n \sum_k z_{nk} \ln\{\pi_k \mathcal{N}(s_n|\mu_k, \Sigma_k)\} \tag{26}$$

$$\therefore \mathcal{Q} = \sum_z p(z|s, \theta^{\text{old}}) \ln(p(s, z|\theta)) \tag{27}$$

$$= \sum_n \sum_k \gamma_{nk} \ln\{s_n|\mu_k, \Sigma_k\} \tag{28}$$