CS 337, Fall 2023
Gaussian Mixture Models (GMMs) for Clustering

Scribes: Soham Joshi, Rahul Kumar, Racha Yashaswi Ratna, Kalvala Vivek, Deepasha, Nagasai Saketh Naidu *
Edited by: Sanjeev Kumar

October 10, 2023

1 K-means Clustering

Recall the (hard) K-means clustering problem. We have the data set $\mathcal{D} = \{\mathbf{x_1}, \cdots, \mathbf{x_n}\}$ consisting of N observations of a random d-dimensional euclidean variable $\mathbf{X}$. We need to partition this dataset into K clusters, with cluster centres $\mu_\mathbf{1}, \cdots, \mu_\mathbf{K}$ and assignments $\mathbf{C^1}, \cdots, \mathbf{C^n}$ such that sum of squared distances of all datapoints from their assigned cluster centres is minimized.

$$\min \sum_{i=1}^{n} \sum_{k=1}^{K} C_k^i \|\mu_\mathbf{k} - \mathbf{x_i}\|^2$$

where $C_k^i = \mathbb{I}(\mu_\mathbf{k}$ is the closest cluster centre to $\mathbf{x_i})$. More formally, we can write

$$C_k^i = \begin{cases} 1, & \text{if } k = \arg\min_j \|\mu_\mathbf{j} - \mathbf{x_i}\|^2 \\ 0, & \text{otherwise} \end{cases}$$

Moreover, this problem can be extended to *soft* K-means clustering, where we have,

$$C_k^i = \frac{\exp\left(-\beta \|\mu_\mathbf{k} - \mathbf{x_i}\|^2\right)}{\sum_{j=1}^{K} \exp\left(-\beta \|\mu_\mathbf{j} - \mathbf{x_i}\|^2\right)}$$

Where, β is a parameter reflecting the amount of flexibility in our soft assignment and called the stiffness parameter.

*Equal contributions by all scribes

1.1 Analysis of an abstract algorithm

Consider the abstract algorithm 1 which is an alternating minimization algorithm, which approximates the parameters minimizing the function $C(\phi, \theta)$, where functions `mintheta` and `minphi` are defined as $\texttt{minphi}(\theta) = \arg\min_{\phi' \in \mathbf{\Phi}} C(\phi', \theta)$ and $\texttt{mintheta}(\phi) = \arg\min_{\theta' \in \mathbf{\Theta}} C(\phi, \theta')$. We can observe that the K-means clustering algorithm follows this template if we take θ as an assignments/labels, and ϕ as cluster centres. Under this paradigm, we note the following observations about this template,

1. The variable `cost` in line 11 of algorithm 1 never increases in value. This is because `cost` $= C(\phi, \theta)$ where θ (resp. ϕ) is updated in line 6 (resp. 8) such that the value of the function C reduces or remains the same. Hence, in each update, the value of the variable `cost` never increases.

2. If θ remains unchanged in line 6 in a particular iteration, then `cost` $= C(\phi, \theta)$ will remain unchanged. Hence, after that iteration, `prevcost` = `cost`, and the algorithm will terminate.

3. Paraphrasing the last point, θ can repeat a value atmost twice, and if the value repeats twice the algorithm terminates. An example where it repeats twice is, $\mathbf{\Phi} = \{\phi_0, \phi_1\}, \mathbf{\Theta} = \{\theta_0, \theta_1\}$ such that,

$$C(\phi_0, \theta_0) = 2$$
$$C(\phi_0, \theta_1) = 3$$
$$C(\phi_1, \theta_0) = 1$$
$$C(\phi_1, \theta_1) = 3$$

   with initialisation $\phi = \phi_0$ at line 3, then in iteration 1, $\theta \leftarrow \theta_0$, in iteration 2, $\phi \leftarrow \phi_1$ and in iteration 3, we get $\theta \leftarrow \theta_0$. Hence, this example shows a value can be repeated twice for θ.

4. If $\mathbf{\Theta}$ is a finite set, but $\mathbf{\Phi}$ is infinite, then in every alternate iteration, value of θ must change (if not, the algorithm terminates). Moreover, since the number of values θ can take is $\|\mathbf{\Theta}\|$, the algorithm must terminate in $2\|\mathbf{\Theta}\| + 1$ steps (since, in the final iteration, the value of θ repeats).

Question 1: (True or False? Justify) In line 11, the cost will never increase its value.
Solution: True.
(θ,ϕ) is updated only in line 6 or line 8. In either case, the updated cost is no more than the cost before the update.
Question 2: (True or False? Justify) If θ remains unchanged when executing line number 6 in a particular iteration, then that will be the last iteration before the program terminates.
Solution: True.
If line 6 is executed, then ϕ remains unchanged. If θ is also unchanged then prevcost = cost in line 13 and the loop ends.
Question 3: (True or False? Justify) If θ is assigned a value θ_o in line 6 at a particular iteration, then it will never be assigned the same value in a later iteration.
Solution: False.
Consider the following example: $\Phi = \{\phi_0,\phi_1\}$, $\Theta = \{\theta_0,\theta_1\}$ such that C(ϕ_0,θ_0)=2, C(ϕ_0,θ_1)=3,

Algorithm 1 Abstract algorithm approximating $\arg\min_{\theta,\phi} C(\phi, \theta)$

```
1: cost ← ∞
2: flag ← true
3: Initialize:
       φ ∈ Φ
4: repeat
5:     if flag = true then
6:         θ ← mintheta(φ)
7:     else
8:         φ ← minphi(θ)
9:     end if
10:    prevcost ← cost
11:    cost ← C(φ, θ)
12:    flag ← NOT flag
13: until prevcost = cost
14: return (φ, θ)
```

$C(\phi_1,\theta_0)=1$, $C(\phi_1,\theta_1)=3$. Then, if at line 3, $\phi = \phi_0$, then Line 6 will be executed and θ is set to θ_0. In the next iteration, ϕ is set to ϕ_1 and in the next(and last) iteration, θ is again set to θ_0.

Question 4: (True or False? Justify) Suppose Θ is a finite set, but Φ is infinite. Then the algorithm could run forever if $\phi^{'} \in \Phi, \arg\min C(\phi^{'}, \theta)$ is not unique for some Θ, depending on how **minphi** breaks ties.

Solution: False.

θ can be repeated at most twice (with the first repetition leading to termination of the loop). Since Θ is finite, and every two iterations θ must be updated, there can be at most $2|\Theta| + 1$ iterations of the loop.

1.2 Generative Model vs Discriminative Model

Suppose you are given a classification problem with data points $\mathbf{X} = \{x_1, x_2....x_n\}$, labels $\mathbf{Y}$. Our main goal is to estimate the $P(Y = y|x)$ i.e. probability of y for a given data point x. For this, we can use one of the two approaches

1.2.1 Discriminative Models

In this model, we use some functional form of the probability $P(\mathbf{Y}|\mathbf{X})$ and estimate its parameters with the help of training data. Such models, estimate $P(\mathbf{Y}|\mathbf{X})$ using observed data $\mathcal{D}$ sampled from the joint distribution of observed and target variables are called discriminative or conditional models. All models pertaining to regression or classification which make predictions using observed data are instances of discriminative models. Common examples include neural networks, SVM and linear regression.

1.2.2 Generative Models

Generative models are a statistical model of the probability distribution $P(\mathbf{X}, \mathbf{Y})$ on a given observable $\mathbf{X}$ and target $\mathbf{Y}$. In this model, we find the likelihood probability $P(\mathbf{X}|\mathbf{Y})$ and the

prior probability $P(\mathbf{Y})$ with the help of training data and use the Bayes theorem to calculate the posterior Probability

$$P(\mathbf{Y}|\mathbf{X}) \ = \frac{P(\mathbf{Y}) \times P(\mathbf{X}|\mathbf{Y})}{P(\mathbf{X})}$$

Some examples of these are Naive Bayes and Gaussian Mixture Models

Clustering using Generative Models

The idea of the clustering algorithm is to apply *Maximum Likelihood Estimation (MLE)* on the random variable $\mathbf{X}$ given the dataset $\mathcal{D} = \{\mathbf{x_1}, \cdots, \mathbf{x_n}\}$. That is we consider the functional form of $P(\mathbf{Y})$ and $P(\mathbf{X}|\mathbf{Y})$ as,

$$P(\mathbf{Z} = k) = \pi_k, \text{where } \sum_{k=1}^{K} \pi_k = 1$$
$$P(\mathbf{X}|\mathbf{Z} = k) = \mathcal{N}(\mathbf{X}; \mu_k, \mathbb{I})^1$$

where we need to estimate $\{\pi_k, \mu_k\}_{k=1}^K$. For this purpose, we use *Maximum Likelihood Estimation (MLE)*. For a single datapoint $\mathbf{x}$, the expression is,

$$\begin{aligned}\max \log P(x) &= \max \log \sum_k P(\mathbf{Z} = k, \mathbf{x}) \\ &= \max \log \sum_k P(\mathbf{x}|\mathbf{Z} = k) P(\mathbf{Z} = k) \\ &= \max \log \sum_k \pi_k \mathcal{N}(\mathbf{x}; \mu_k, \mathbb{I})\end{aligned}$$

So, we can extend this to multiple datapoints as,

$$\max \sum_{x \in \mathcal{D}} \log P(x) = \max \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x_i}; \mu_k, \mathbb{I})$$

where, the solution to this system of equations will yield the desired parameters $\{\pi_k, \mu_k\}_{k=1}^K$.

We will anaylse this problem by assuming a fully observable setting i.e. we will assume that we know the z_i for each datapoint x_i. So our problem becomes.

$$\begin{aligned}\max \sum_{x \in \mathcal{D}} \log P(x) &= \max \sum_{i=1}^{n} \log \sum_{k=1}^{K} (\mathbb{I}[k = z_i] \pi_k \mathcal{N}(\mathbf{x_i}; \mu_k, \mathbb{I}) \\ &= \max \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{I}[k = z_i] \log \pi_k \mathcal{N}(\mathbf{x_i}; \mu_k, \mathbb{I}) \qquad \text{(A)}\end{aligned}$$

[1]Here, the covariance matrix could also be parametrized as $\Sigma = \Sigma_z$. However for simplicity of subsequent derivations, we assume $\Sigma = \mathbb{I}$.

Value of μ_k

To maximize the likelihood with respect to μ_k, we start with the likelihood for data point x_i belonging to component k, which is:

$$\mathbb{I}[k = z_i] \cdot \mathcal{N}(\mathbf{x}_i; \mu_k, \mathbb{I})$$

The derivative of the logarithm of the Gaussian is:

$$\begin{aligned}
\frac{\partial}{\partial \mu_k} \log(\mathcal{N}(\mathbf{x}_i; \mu_k, \mathbb{I})) &= \frac{\partial}{\partial \mu_k} \left(-\frac{1}{2}(\mathbf{x}_i - \mu_k)^\top \mathbb{I}^{-1}(\mathbf{x}_i - \mu_k) - \frac{D}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbb{I}|) \right) \\
&= \frac{\partial}{\partial \mu_k} \left(-\frac{1}{2}(\mathbf{x}_i - \mu_k)^\top \mathbb{I}^{-1}(\mathbf{x}_i - \mu_k) \right) \\
&= \frac{1}{2} \frac{\partial}{\partial \mu_k} \left((\mathbf{x}_i - \mu_k)^\top \mathbb{I}^{-1}(\mathbf{x}_i - \mu_k) \right) \\
&= -\mathbb{I}^{-1}(\mathbf{x}_i - \mu_k)
\end{aligned}$$

$$\frac{\partial}{\partial \mu_k} \log(\mathcal{N}(\mathbf{x}_i; \mu_k, \mathbb{I})) = -(\mathbf{x}_i - \mu_k)$$

Now, we can find the derivative of the likelihood for component k with respect to μ_k:

$$\frac{\partial}{\partial \mu_k} \left(\mathbb{I}[k = z_i] \cdot \mathcal{N}(\mathbf{x}_i; \mu_k, \mathbb{I}) \right) = \mathbb{I}[k = z_i] \cdot (\mathbf{x}_i - \mu_k)$$

To maximize the likelihood, we set this derivative to zero:

$$\sum_{i=1}^{N} \mathbb{I}[k = z_i] \cdot (\mathbf{x}_i - \mu_k) = 0$$

Now, solve for μ_k:

$$\mu_k = \frac{\sum_{i=1}^{N} \mathbb{I}[k = z_i] \cdot \mathbf{x}_i}{\sum_{i=1}^{N} \mathbb{I}[k = z_i]}$$

Value of Mixing Coefficient π_k

To update the mixing coefficient π_k, we want to find the Maximum Likelihood Estimate (MLE). The mixing coefficient represents the fraction of data points assigned to component k:

Now for finding π_k, we can not directly differentiate w.r.t π_k, as we $\sum_{k=1}^{K} \pi_k = 1$, so we will use the Lagrangian function.

In the general case, the Lagrangian is defined as follows, We have variables $x_1, x_2, ...x_n$ and we need to optimize $f(x_1, x_2, ...x_n)$ under the constraints that $g(x_1, x_2...x_n) = 0$. To do that we will use the Lagrange function $\mathcal{L}(x_1, x_2...x_n, \lambda) \equiv f(x_1, x_2...x_n) - \lambda \cdot g(x_1, x_2...x_n)$, where λ is the Lagrange multiplier and just solve for $x_1, x_2...x_n, \lambda$ i.e.

$$\frac{\partial f(x_1, x_2...x_n)}{\partial x_i} = \lambda \cdot \frac{\partial g(x_1, x_2...x_n)}{\partial x_i}$$
$$g(x_1, x_2...x_n) = 0$$

which will be our required solution.

For the purpose of our problem, $f \equiv \sum_{i=1}^{N} log(P(x_i))$ and $g \equiv \sum_{k=1}^{K} \pi_k - 1$

$$\frac{\partial(\sum_{i=1}^{N} log(P(x_i)))}{\partial \pi_k} = \lambda \cdot \frac{\partial g}{\partial \pi_k}$$
$$\sum_{i=1}^{N} \mathbb{I}(z_i = k) \cdot \frac{1}{\pi_k} = \lambda$$
$$\frac{\sum_{i=1}^{N} \mathbb{I}(z_i = k)}{\lambda} = \pi_k$$

As we have $\sum_{k=1}^{K} \pi_k = 1$ so,

$$\lambda = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}(z_i = k) = n$$
$$\pi_k = \frac{1}{n} \cdot \sum_{i=1}^{N} \mathbb{I}(z_i = k)$$

$$\pi_k = \frac{\sum_{i=1}^{n} \mathbb{I}[k = z_i]}{N}$$

Here, the numerator is simply the count of data points assigned to component k, and the denominator is the total number of data points.

Values of μ_k and π_k are :

$$\mu_k = \frac{\sum_{i}^{N} \mathbb{I}[z_i = k] x_i}{\sum_{i}^{N} \mathbb{I}[z_i = k]}$$
$$\pi_k = \frac{\sum_{i}^{N} \mathbb{I}[z_i = k]}{n}$$

But we do not have the information about $\mathbb{I}[z_i = k]$, so we compute the $E[\mathbb{I}[z_i = k]] = P(z_i = k|x_i)$ and use it in place of $\mathbb{I}[z_i = k]$.

This is how we go from hard k-means to soft k-means clustering.

$$P(z_i = k|x_i) = \frac{P(z_i)P(x_i = k|z_i)}{P(x_i)} \qquad \text{refer this as } \gamma_{i,k}$$
$$= \frac{\pi_k \mathcal{N}(\mathbf{x_i}; \mu_k, \mathbb{I})}{\sum_j \pi_j \mathcal{N}(\mathbf{x_i}; \mu_j, \mathbb{I})}$$

Then our objective becomes the following:

$$\max \sum_{x \in \mathcal{D}} \log P(x) = \max \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{i,k} \log \pi_k \mathcal{N}(\mathbf{x_i}; \mu_k, \mathbb{I})$$

We got the above equation from A assuming that $\gamma_{i,k}$ are fixed . Meaning they use current values of the parameter μ_k and π_k .
Once again we encounter a chicken and egg problem: We need μ_k and π_k to get $\gamma_{i,k}$ and vice versa.

This motivates us to **Expectation Minimization** Algorithm

2 Gaussian Mixture Models

As seen in section 1.2.2, Gaussian mixture models can be used to model approximate solutions of K-means clustering. However, in this section, we define Gaussian mixture models more formally. A Gaussian mixture model can be written as a linear superposition of Gaussians of the form,

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k})$$

We introduce a random variable $\mathbf{z}$ in $\mathbb{R}^K$, where a particular element in z_k is 1 and the rest are zeros. So, $\mathbf{z}$ satisfies $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. Now, we define marginal for $\mathbf{z}$ as $p(z_k = 1) = \pi_k$ where the parameters π_k satisfy

$$0 \leq \pi_k \leq 1, \sum_{k=1}^{K} \pi_k = 1$$

because $\mathbf{z}$ consists of "one-hot" vectors, this can be written as

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

along with,

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k})$$
$$\Rightarrow p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \Sigma_k)^{z_k}$$

Hence,

$$p(\mathbf{x}) = \sum_{z} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k})$$

2.1 Some terminology

If we have several observations $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}$ then, for every observed datapoint $\mathbf{x_n}$, there is a corresponding **latent** (or hidden) variable $\mathbf{z_n}$.
As seen in section 1.2.2, we can model the log-likelihood function as,

$$\log p(\mathbf{X}|\pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} \log\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k})\}$$

This can be represented succinctly using figure 1 (refer [1]).

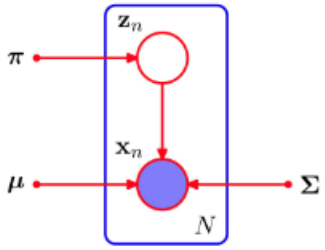

Figure 1: Graphical representation of a Gaussian mixture model for a set of N i.i.d data points $\{x_n\}$, with corresponding latent points $\{z_n\}$, where $n = 1, \ldots, N$.

The maximisation of this function however, is no easy business, and we run into a chicken and egg problem as described in section 1.2.2 because of presence of latent random variables $\mathbf{z}$ and presence of *singularities*[2].

2.2 The problem of singularity

Singularities prevent us from solving this as a maximum likelihood problem. In order to understand it, let us try our hand at formulating this as a maximum likelihood problem [3].
Suppose that one of the components of the mixture model says the j^{th} component has $\mu_\mathbf{j}$ exactly equal to one of the data points so that $\mu_\mathbf{j} = \mathbf{x_n}$ for some value of n. The datapoint will then

[2]We did not explicitly encounter this problem in 1.2.2 because of the assumption $\Sigma = \mathbb{I}$, however for a general Σ things get out of hand.

[3]Content borrowed heavily from [1] section 9.2.1. So ... just read that to gain a better intuition of what is really happening

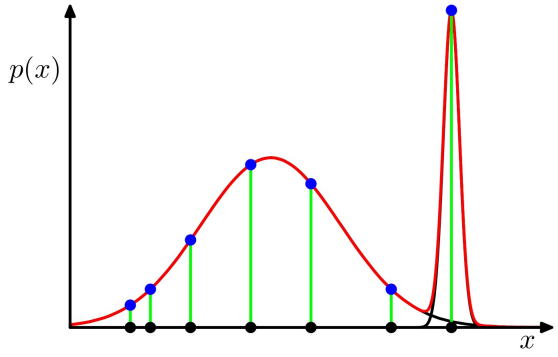

Figure 2: An illustration of how singularities in the likelihood function arise with mixtures of Gaussians

contribute a term in the likelihood function of the form[4],

$$\mathcal{N}(\mathbf{x_n}|\mathbf{x_n}, \sigma_j^2 \mathbb{I}) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_j}$$

In the limit $\sigma_j \to 0$, we see that this term goes to infinity (as seen in figure 2), so does the log-likelihood.

Hence, the Gaussian component *collapses* onto a single data point. This problem did not arise in single Gaussians because if it overfits to a point, it causes rather bad factors for the other datapoints. But, in the case of a mixture model, many components can *collapse* to respective datapoints, and their addition can be very large for each datapoint. Hence, this approach causes textbook *overfitting*.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4]Recall, we take $\mathbf{\Sigma} = \mathbb{I}$ or in this case, $\mathbf{\Sigma} = \sigma^2 \mathbb{I}$ for simplicity in subsequent calculations