As recompiled from `https://pillowlab.princeton.edu/teaching/statneuro2024/notes/notes17_MoG.pdf`

EM for the Mixture of Gaussians model

1 General form of EM

$$L = \overbrace{\log p(x \mid \theta)}^{\text{log likelihood}} = \log \int \overbrace{p(x, z \mid \theta)}^{\substack{\text{total-data} \\ \text{likelihood}}} dz \geq \int \underbrace{q(z \mid \phi)}_{\substack{\text{variational} \\ \text{distribution}}} \log \left[\frac{p(x, z \mid \theta)}{q(z \mid \phi)} \right] dz \triangleq \overbrace{F(\phi, \theta)}^{\text{ELBO}} \tag{1}$$

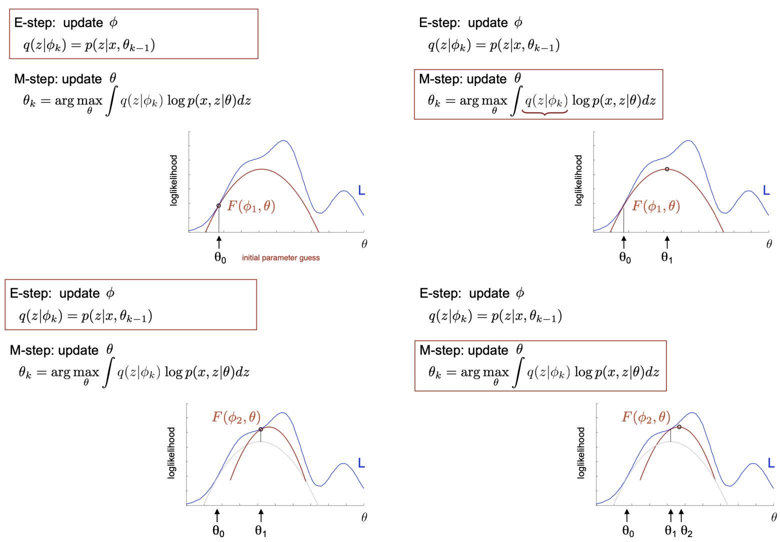
The ELBO, or negative free energy, has two convenient forms, which we exploit in the two alternating phases of EM:

$$F(\phi, \theta) = \log p(x \mid \theta) - KL(q(z \mid \phi) \| p(z \mid x, \theta)) \quad \text{(used in E-step)} \tag{2}$$

$$F(\phi, \theta) = \int q(z \mid \phi) \log p(x, z \mid \theta) \, dz + \underbrace{H[q(z \mid \phi)]}_{\text{indep of } \theta} \quad \text{(used in M-step)} \tag{3}$$

Specifically, EM involves alternating between:

- E-step: Update ϕ by setting $q(z \mid \phi) = p(z \mid x, \theta)$, with θ held fixed.
- M-step: Update θ by maximizing $\int q(z \mid \phi) \log p(x, z \mid \theta) dz$, with ϕ held fixed.



Note that for discrete latent variable models, where the latent z takes on finite or countably infinitely many discrete values, the integral over z is replaced by a sum:

$$F(\phi, \theta) = \sum_{j=1}^{m} q(z = z_j \mid \phi) \log p(x, z = z_j \mid \theta) \, dz + \underbrace{H[q(z \mid \phi)]}_{\text{indep of } \theta} \tag{4}$$

where $\{z_1, \ldots, z_m\}$ are the possible values of z.

1.1 Motivating the variational posterior, $q(z \mid \phi)$

Read this if the above is confusing

We sample some data, and think it emerges from a distribution with a latent variable, z – e.g. GMMs. We want to see what this distribution looks like, but we can only draw x. So what we need to find is $p(z|x)$. Our immediate instinct should be to use:

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

Unfortunately, $p(x)$ is an intractable integral as it marginalises x over all possible z. Bayes by itself is therefore no good. We instead resort to directly approximating the posterior $p(z \mid x)$ using EM. To do this, we assume there exists a distribution $q(z \mid \phi)$ that approximates the posterior well. Notice that we use ϕ to signal that we are looking at the particular distribution with parameters that does this best. We know that this distribution should closely resemble the posterior, so as to minimise the KL-divergence below.

$$q^*(z \mid \phi) = \arg\min \mathrm{KL}(q(z \mid \phi) \parallel p(z \mid x))$$

The intractable $p(x)$ again appears, hidden in the posterior. We can use Bayes on $p(x)$ to re-express the KL-divergence as:

$$\mathrm{KL}(q(z \mid \phi) \,||\, p(z \mid x)) = \underbrace{\mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q(z \mid \phi)]}_{\text{ELBO}} + \log p(x)$$

Notice that we have aggregated the messy expectations into a term we call ELBO. We can show using Jensen's inequality that log $p(x)$ is always greater than ELBO. So if we wish to minimise KL, we can only decrease ELBO, as log $p(x)$ is invariant of ϕ. The conclusion is that minimising ELBO is equivalent to minimising KL.

Looking at EM in the context of Mixture of Gaussians may clarify things further.

2 EM for Mixture of Gaussians

The model:

$$z \sim \mathrm{Ber}(p) \tag{5}$$

$$x \Big| z \sim \begin{cases} \mathcal{N}(\mu_0, C_0), & \text{if } z = 0 \\ \mathcal{N}(\mu_1, C_1), & \text{if } z = 1 \end{cases} \tag{6}$$

Suppose we have a dataset consisting of N samples $\{x_i\}_{i=1}^N$. Our model seeks to model these in terms of a set of pairs of iid random variables $\{(z_i, x_i)\}_{i=1}^N$, each consisting of a latent and an observation. These samples are independent under the model, meaning that the log-likelihood is given by a sum of independent terms, each of which is bounded by its ELBO:

$$\log p\left(\{x_i\}_{i=1}^N \mid \theta\right) = \sum_{i=1}^N \log p(x_i \mid \theta) \tag{7}$$

$$\geq \sum_{i=1}^N \mathrm{ELBO}(x_i \mid \theta) \tag{8}$$

Each of these ELBO terms is given by a sum over the discrete values of z :

$$\mathrm{ELBO}(x_i \mid \theta) = \sum_{z_i=0}^1 q(z_i \mid \phi_i) \frac{\log p(x_i, z_i \mid \theta)}{q(z_i \mid \phi)} \tag{9}$$

$$= q(z_i = 0 \mid \phi_i) \log \frac{p(x_i, z_i = 0 \mid \theta)}{q(z_i = 0 \mid \phi_i)} + q(z_i = 1 \mid \phi_i) \log \frac{p(x_i, z_i = 1 \mid \theta)}{q(z_i = 1 \mid \phi_i)} \tag{10}$$

Thus the ELBO for the entire model can be written as the sum over all datapoints:

$$ELBO = \sum_{i=1}^N \left[q(z_i = 0 \mid \phi_i) \log \frac{p(x_i, z_i = 0 \mid \theta)}{q(z_i = 0 \mid \phi_i)} + q(z_i = 1 \mid \phi_i) \log \frac{p(x_i, z_i = 1 \mid \theta)}{q(z_i = 1 \mid \phi_i)} \right] \tag{11}$$

Here ϕ_i is the variational parameter associated with the i 'th latent variable z_i.

2.1 E-step

The E step involves setting $q(z_i \mid \phi_i)$ equal to the conditional distribution of z_i given the data and current parameters θ. We will denote these binary probabilities by ϕ_{i0} and ϕ_{i1}, given by the recognition distribution of the z_i under the model:

$$\phi_{i0} = p(z_i = 0 \mid x_i, \theta) = \frac{(1-p)\mathcal{N}_0(x_i)}{(1-p)\mathcal{N}_0(x_i) + p\mathcal{N}_1(x_i)} \tag{12}$$

$$\phi_{i1} = p(z_i = 1 \mid x_i, \theta) = \frac{p\mathcal{N}_1(x_i)}{(1-p)\mathcal{N}_0(x_i) + p\mathcal{N}_1(x_i)} \tag{13}$$

where $\mathcal{N}_0(x_i) = \mathcal{N}(x_i \mid \mu_0, C_0)$ and $\mathcal{N}_1(x_i) = \mathcal{N}(x_i \mid \mu_1, C_1)$, and note that $\phi_{i0} + \phi_{i1} = 1$.
At the end of the E-step we have a pair of these probabilities for each sample, which can be represented as an $N \times 2$ matrix

$$\phi = \begin{bmatrix} \phi_{10} & \phi_{11} \\ \phi_{20} & \phi_{21} \\ \vdots & \vdots \\ \phi_{N0} & \phi_{N1} \end{bmatrix} \tag{14}$$

where each row sums to 1 . The values ϕ_{ij} are often described as the "soft assignments" of datapoints to classes, since they describe how likely it is that datapoint i came from class j, given the current model parameters θ.

2.2 M-step

The M-step involves updating the parameters $\theta = \{p, \mu_0, \mu_1, C_0, C_1\}$ using the current variational distribution $q(z \mid \phi)$. To do this, we plug in the assignment probabilities $\{\phi_{i0}, \phi_{i1}\}$ from the E-step into the ELBO (eq. 11) to obtain:

$$\begin{aligned} F &= \sum_{i=1}^{N} \left[\phi_{i0} \log p\left(x_i, z_i = 0 \mid \theta\right) + \phi_{i1} \log p\left(x_i, z_i = 1 \mid \theta\right)\right] && (15) \\ &= \sum_{i=1}^{N} \left[\phi_{i0}\left(\log(1-p) + \log \mathcal{N}\left(x_i \mid \mu_0, C_0\right)\right) + \phi_{i1}\left(\log p + \log \mathcal{N}\left(x_i \mid \mu_1, C_1\right)\right)\right] && (16) \end{aligned}$$

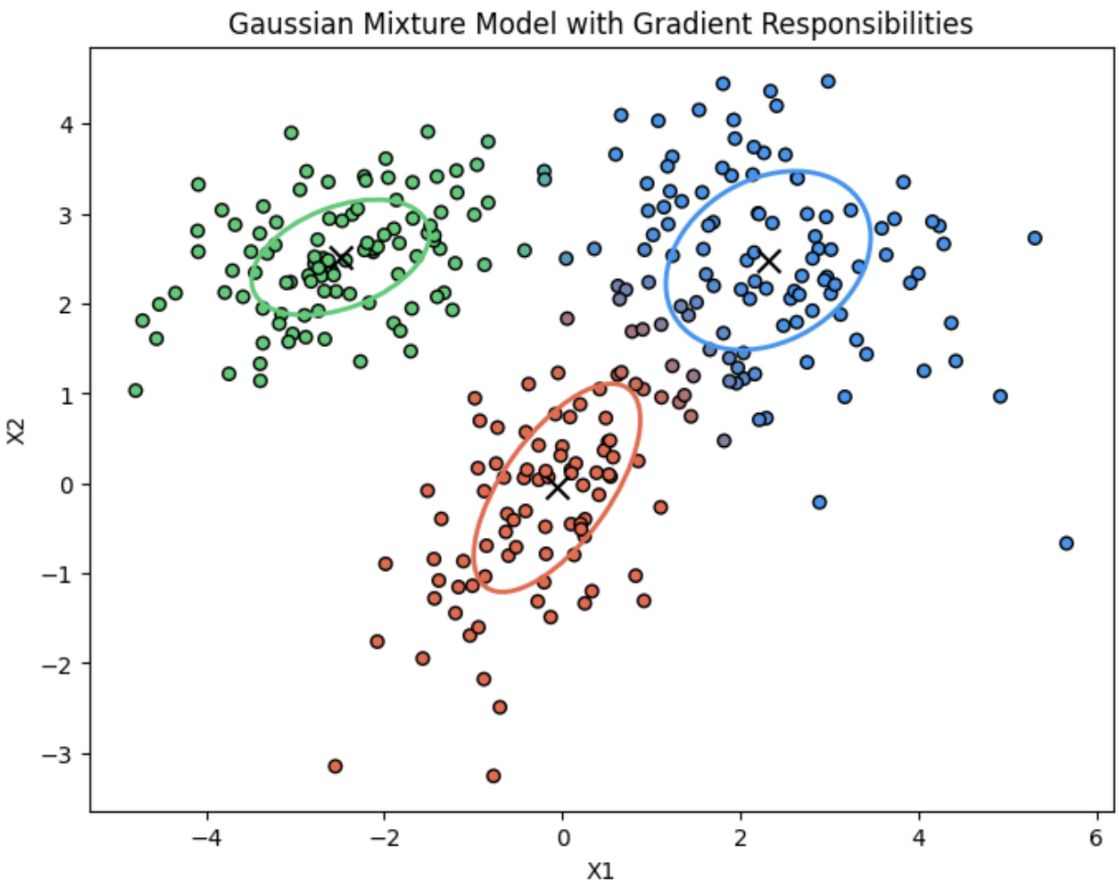
Maximizing this expression for the model parameters (see next section for derivations) gives updates:

$$\begin{aligned} \hat{\mu}_0 &= \left(\frac{1}{\sum \phi_{i0}}\right) \sum \phi_{i0} x_i && (17) \\ \hat{\mu}_1 &= \left(\frac{1}{\sum \phi_{i1}}\right) \sum \phi_{i1} x_i && (18) \\ \hat{C}_0 &= \left(\frac{1}{\sum \phi_{i0}}\right) \sum \phi_{i0}\left(x_i - \hat{\mu}_0\right)\left(x_i - \hat{\mu}_0\right)^{\top} && (19) \\ \hat{C}_1 &= \left(\frac{1}{\sum \phi_{i1}}\right) \sum \phi_{i1}\left(x_i - \hat{\mu}_1\right)\left(x_i - \hat{\mu}_1\right)^{\top} && (20) \\ p &= \frac{1}{N} \sum \phi_{i1} && (21) \end{aligned}$$

Note that the mean and covariance updates are formed by taking the weighted average and weighted covariance of the samples, with weights given by the assignment probabilities ϕ_{i0} and ϕ_{i1}.

3 GMM EM Algorithm

```python
import numpy as np

# Initialise parameters
def initialise_parameters(X, K):
    N, D = X.shape
    pi = np.ones(K) / K  # Initialise mixing coefficients
    mu = np.random.randn(K, D)  # Initialise means
    sigma = np.array([np.eye(D) for _ in range(K)])  # Initialise covariances
    return pi, mu, sigma

# E-step: Calculate responsibilities (posterior probabilities)
def e_step(X, pi, mu, sigma):
    N, K = X.shape[0], pi.shape[0]
    gamma = np.zeros((N, K))
    for k in range(K):
        gamma[:, k] = pi[k] * multivariate_gaussian(X, mu[k], sigma[k])
    return gamma / gamma.sum(axis=1, keepdims=True)  # Normalise

# M-step: Update parameters based on responsibilities
def m_step(X, gamma):
    N, D = X.shape
    K = gamma.shape[1]
    N_k = gamma.sum(axis=0)  # Effective number of samples for each component
    pi = N_k / N  # Update mixing coefficients
    mu = np.dot(gamma.T, X) / N_k[:, np.newaxis]  # Update means
    sigma = np.array([(gamma[:, k][:, np.newaxis] * (X - mu[k])).T @ (X - mu[k]) / N_k[k
        ]
                      for k in range(K)])  # Update covariances
    return pi, mu, sigma

# Gaussian likelihood function
def multivariate_gaussian(X, mu, sigma):
    D = X.shape[1]
    norm_const = 1 / (np.sqrt((2 * np.pi) ** D * np.linalg.det(sigma)))
    X_centered = X - mu
    return norm_const * np.exp(-0.5 * np.sum(X_centered @ np.linalg.inv(sigma) *
        X_centered, axis=1))

# Log likelihood calculation
def log_likelihood(X, pi, mu, sigma):
    return np.sum(np.log(np.sum([pi[k] * multivariate_gaussian(X, mu[k], sigma[k]) for k
         in range(len(pi))], axis=0)))

# EM Algorithm
def em_algorithm(X, K, max_iters=100, tol=1e-4):
    pi, mu, sigma = initialise_parameters(X, K)
    log_likelihood_values = []

    for _ in range(max_iters):
        # E-step
        gamma = e_step(X, pi, mu, sigma)

        # M-step
        pi, mu, sigma = m_step(X, gamma)

        # Log likelihood
        ll = log_likelihood(X, pi, mu, sigma)
        log_likelihood_values.append(ll)

        # Check for convergence
        if len(log_likelihood_values) > 1 and np.abs(log_likelihood_values[-1] -
            log_likelihood_values[-2]) < tol:
            break

    return pi, mu, sigma, log_likelihood_values
```



4 Derivation of M-step updates

4.1 Updates for $\mu 0, \mu_1$

To derive the updates for μ_0, we collect the terms from the ELBO (eq. 16) that involve μ_0, giving:

$$
\begin{aligned}
F\left(\mu_0\right) & =\sum_{i=1}^N \phi_{i 0} \log \mathcal{N}\left(x_i \mid \mu_0, C_0\right)+\text { const } \quad (22)\\
& =-\frac{1}{2} \sum_{i=1}^N \phi_{i 0}\left(x_i-\mu_0\right)^{\top} C_0^{-1}\left(x_i-\mu_0\right)+\text { const } \quad (23)
\end{aligned}
$$

$$
-\frac{1}{2} \sum_{i=1}^N \phi_{i 0}\left(-2 \mu_0^{\top} C_0^{-1} x_i+\mu_0^{\top} C_0^{-1} \mu_0\right)+\text{const} \quad (24)
$$

$$
=\mu_0^{\top} C_0^{-1}\left(\sum_{i=1}^N \phi_{i 0} x_i\right)-\frac{1}{2}\left(\sum_{i=1}^N \phi_{i 0}\right) \mu_0^{\top} C_0^{-1} \mu_0+\text{const.} \quad (25)
$$

Differentiating with respect to μ_0 and setting to zero gives:

$$\frac{\partial}{\partial \mu_0} F = C_0^{-1}\left(\sum_{i=1}^{N} \phi_{i0} x_i\right) - \left(\sum_{i=1}^{N} \phi_{i0}\right) C_0^{-1} \mu_0 = 0 \tag{26}$$

$$\implies \left(\sum_{i=1}^{N} \phi_{i0} x_i\right) = \left(\sum_{i=1}^{N} \phi_{i0}\right) \mu_0 \tag{27}$$

$$\implies \hat{\mu}_0 = \frac{\sum_{i=1}^{N} \phi_{i0} x_i}{\sum_{i=1}^{N} \phi_{i0}} \tag{28}$$

A similar approach leads to the update for μ_1, with weights ϕ_{i1} instead of ϕ_{i0}.

4.2 Updates for C_0, C_1

Matrix derivative identities:

Assume C is a symmetric, positive definite matrix. We have the following identities ([1]):

- log-determinant:

$$\frac{\partial}{\partial C} \log |C| = C^{-1} \tag{29}$$

- quadratic form:

$$\frac{\partial}{\partial C} x^\top C x = x x^\top \tag{30}$$

Derivation:

The simplest approach for deriving updates for C_0 is to differentiate the ELBO F with respect to C_0^{-1} and then solve for C_0. We assume we already have the updated mean $\hat{\mu}_0$ (which did not depend on C_0 or any other parameters).

The ELBO as a function of C_0 can be written:

$$F\left(C_0\right) = \sum_{i=1}^{N} \phi_{i0} \log \mathcal{N}\left(x_i \mid \hat{\mu}_0, C_0\right) + \text{ const} \tag{31}$$

$$= \sum_{i=1}^{N} \phi_{i0}\left(+\frac{1}{2} \log \left|C_0^{-1}\right| - \frac{1}{2}\left(x_i - \hat{\mu}_0\right)^\top C_0^{-1}\left(x_i - \hat{\mu}_0\right)\right) + \text{const} \tag{32}$$

Differentiating with respect to C_0^{-1} gives us:

$$\frac{\partial}{\partial C_0^{-1}} F = \frac{1}{2}\left(\sum_{i=1}^{N} \phi_{i0}\right) C_0 + \frac{1}{2}\left(\sum_{i=1}^{N} \phi_{i0}\left(x_i - \hat{\mu}_0\right)\left(x - \hat{\mu}_0\right)^\top\right) = 0 \tag{33}$$

$$\implies \hat{C}_0 = \frac{1}{\left(\sum_{i=1}^{N} \phi_{i0}\right)}\left(\sum_{i=1}^{N} \phi_{i0}\left(x_i - \hat{\mu}_0\right)\left(x - \hat{\mu}_0\right)^\top\right) \tag{34}$$

which as noted above is simply the covariance matrix of all stimuli weighted by their recognition weights. The same derivation can be used for C_1.

4.3 Mixing probability p update

Finally, updates for p are obtained by collecting terms involving p :

$$
\begin{aligned}
F(p) & =\sum_{i=1}^{N}\left(\phi_{i 0} \log (1-p)+\phi_{i 1} \log p\right)+\text { const } && (35)\\
& =\log (1-p)\left(\sum_{i=1}^{N} \phi_{i 0}\right)+(\log p)\left(\sum_{i=1}^{N} \phi_{i 0}\right)+\text { const } && (36)
\end{aligned}
$$

Differentiating and setting to zero gives

$$
\begin{aligned}
\frac{\partial}{\partial p} F=\frac{1}{p-1}\left(\sum_{i=1}^{N} \phi_{i 0}\right)+\frac{1}{p}\left(\sum_{i=1}^{N} \phi_{i 1}\right)=0 && (37)\\
\Longrightarrow p\left(\sum_{i=1}^{N} \phi_{i 0}\right)+(p-1)\left(\sum_{i=1}^{N} \phi_{i 1}\right)=0 && (38)\\
\Longrightarrow p\left(\sum_{i=1}^{N} \phi_{i 0}+\phi_{i 1}\right)=\left(\sum_{i=1}^{N} \phi_{i 1}\right) && (39)\\
\Longrightarrow p=\frac{1}{N} \sum_{i=1}^{N} \phi_{i 1} && (40)
\end{aligned}
$$

where note that we have used $p_{i 0}+p_{i 1}=1$ for all i. Thus the m -step estimate for p is simply the average probability assigned to cluster 1 .

5 K-means clustering

The K-means clustering algorithm can be seen as applying the EM algorithm to a mixture-of Gaussians latent variable with covariances $C_{0}=C_{1}=\epsilon I$ in the limit where $\epsilon \longrightarrow 0$. Note that in this limit the recognition probabilities go to 0 or 1 :

$$
p(z=1 \mid x)=\frac{p N\left(x \mid \mu_{1}, \epsilon I\right)}{p N\left(x \mid \mu_{0}, \epsilon I\right)+(1-p) N\left(x \mid \mu_{0}, \epsilon I\right)} \quad (41)
$$

$$
=\frac{1}{1+\frac{1-p}{p} \exp \left(\frac{1}{2 \epsilon}\left(\left\|x-\mu_{1}\right\|^{2}-\left\|x-\mu_{0}\right\|^{2}\right)\right)} \quad (42)
$$

$$
=\left\{\begin{array}{ll}
0, & \text { if }\left\|x-\mu_{1}\right\|^{2}>\left\|x-\mu_{0}\right\|^{2} \\
1, & \text { if }\left\|x-\mu_{1}\right\|^{2}<\left\|x-\mu_{0}\right\|^{2} .
\end{array}\right. \quad (43)
$$

The E-step for this model results in "hard assignments", since each datapoint is assigned definitively to one cluster or the other, and the M-step involves updating the means μ_{0} and μ_{1} to be the sample means of the points assigned to each cluster.

Note that the recognition distribution is independent of p, and we can therefore drop that parameter from the model. Thus, the only parameters of the K-means model are the means μ_{0} and μ_{1}.

References

[1] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. Technical University of Denmark, 7(15):510, 2008.