CS 181 Spring 2024 Section 6
Clustering and Mixture Models
Solution

1 Clustering

1.1 Motivation

We now move onto **unsupervised learning**, where the objective is to learn the structure of unlabeled data. In other words, we are looking for groups, or **clusters** among the data. Clustering algorithms are useful not only for finding groups in data, but also to extract features of the data that summarize the most important information about the data in a compressed way.

1.2 Setup

For most clustering algorithms, we need some kind of a metric to specify the notion of "distance" between the data points. If, for example, the points $\mathbf{x}$ and $\mathbf{x'}$ live in some Euclidean space $\mathbb{R}^m$, then the natural choice of such metric is the l_2 distance:

$$||\mathbf{x} - \mathbf{x'}||_2 = \sqrt{\sum_{i=1}^{n}(x_i - x_i')^2}$$

Now that the metric is well-defined, the next thing we need to do is to decide how many groups we want. Sometimes you know the ideal number of groups in advance (*e.g.* clustering the 26 letters in the alphabet). Other times, you need to decide if you'd like a more compressed representation with more information loss by having the number of groups small, or a less compressed representation with less information loss by having the number of groups large.

Suppose our data set is $\{\mathbf{x}_i\}_{i=1}^n$, then our objective is to find the ideal assignment of the data set to the clusters, by assigning to each of the n data points, a binary **responsibility vector** $\mathbf{r}_i$, which is all zeros except one component, corresponding to the assigned cluster.

1.3 K-Means Algorithm

The idea is to represent each cluster by the point in data space that is the average of the data assigned to it. For some choice of K and random initialization of clusters, the K-Means Algorithm (also called Lloyd's algorithm) is:

Repeat until convergence (none of the responsibility vectors change):

1. For each data point, update its responsibility vector by assigning it to the cluster with the closest mean.
2. For each cluster, $\{\boldsymbol{\mu}_k\}_{k=1}^K$, update its mean to be the mean of the data points currently assigned to that cluster.

1.3.1 Derivation

We begin by defining a loss function that the K-Means Algorithm minimizes via coordinate descent:

$$\mathcal{L}(\{\mathbf{r}_i\}_{i=1}^n, \{\boldsymbol{\mu}_k\}_{k=1}^K) = \sum_{i=1}^n \sum_{k=1}^K r_{ik} ||\mathbf{x}_i - \boldsymbol{\mu}_k||_2^2$$

First, we want to choose r_i that minimizes the loss, holding all else constant. This is when we assign data points to the clusters with means closest to them:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_{k'} ||\mathbf{x}_i - \boldsymbol{\mu}_{k'}||_2 \\ 0 & \text{otherwise} \end{cases}$$

This is the first step of each iteration of the K-means algorithm!

Second, we want to choose μ_k that minimizes the loss, holding all else constant. For a given k, the squared loss is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}_k) &= \sum_{i=1}^n r_{ik} ||\mathbf{x}_i - \boldsymbol{\mu}_k||_2^2 \\ &= \sum_{i=1}^n r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)^T (\mathbf{x}_i - \boldsymbol{\mu}_k) \end{aligned}$$

Taking the derivative and setting it to zero,

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\mu}_k)}{\partial \boldsymbol{\mu}_k} &= -2 \sum_{i=1}^n r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k) = 0 \\ \boldsymbol{\mu}_k &= \frac{\sum_{i=1}^n r_{ik} \mathbf{x}_i}{\sum_{i=1}^n r_{ik}} \end{aligned}$$

This is the second step of each iteration of the K-means algorithm!

1.3.2 Number of Clusters

There is not an especially well justified method to choose the number of clusters when using K-means. One approach is to plot K vs the objective criterion, and look for a "knee" or "kink" where progress slows down.

An advanced method is to use the "gap statistic". But this is out of scope for the course.

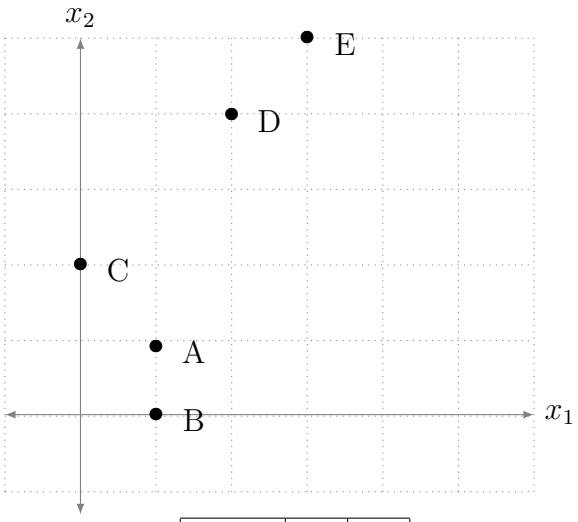
1.3.3 Notes

Lloyd's algorithm finds a locally optimal solution. Finding the globally optimal is NP-hard. A common strategy is to use random restarts. More recently, an algorithm called **K-Means++** has enjoyed popular usage as an alternative to random initialization. This is out of scope for the course, but the basic idea is to randomly select some of the data to be the first cluster centers. This is done by iteratively adding cluster centers, sampling them in proportion to the squared distance of each example from its nearest cluster center. Thus K-Means++ tends to favor points that are distant from the existing centers and produce a more diverse set of centers.

It is generally a good idea to **standardize** the data to account for unsatisfying result due to dimension mismatch. Lastly, when for the metric we are using for the given data set, a "mean" does not make sense, we might instead use a **K-Medoids Algorithm**. This algorithm requires the cluster centers to be a data point in the data set.

1.3.4 Exercise: K-Means (Di Cook)

Use K-means to cluster these examples in $\mathbb{R}^2$, looking for $K = 2$ clusters. Suppose that points A and C are randomly selected as the initial means.



Point	x_1	x_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

If we start with points A and C as our cluster, then let us set our cluster 1 mean $\boldsymbol{\mu}_1 = (1, 1)$ and cluster 2 mean to be $\boldsymbol{\mu}_2 = (0, 2)$. For each point, we will now calculate its distance from each cluster mean and assign it to the cluster where this distance is minimized.

For $\boldsymbol{\mu} = [1, 1]$, $\boldsymbol{\mu}_2 = [0, 2]$

Point	$distc_1$	$distc_2$	$cluster$
A	0	$\sqrt{2}$	1
B	1	$\sqrt{5}$	1
C	$\sqrt{2}$	0	2
D	$\sqrt{10}$	$\sqrt{8}$	2
E	$\sqrt{20}$	$\sqrt{18}$	2

Now we want to update our cluster means so that it takes the average of the coordinates for all points assigned to this cluster.

$$\boldsymbol{\mu}_1 = \frac{\sum_{i=1}^{n} r_{i1}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i1}} = [1, .5]$$
$$\boldsymbol{\mu}_2 = \frac{\sum_{i=1}^{n} r_{i2}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i2}} = [\frac{5}{3}, \frac{11}{3}]$$

For $\boldsymbol{\mu}_1 = [1, .5]$, $\boldsymbol{\mu}_2 = [\frac{5}{3}, \frac{11}{3}]$

Point	$distc_1$	$distc_2$	$cluster$
A	.5	$\sqrt{\frac{2}{3}^2 + \frac{8}{3}^2} = 2.7$	1
B	.5	$\sqrt{\frac{2}{3}^2 + \frac{11}{3}^2} = 3.7$	1
C	$\sqrt{1 + 1.5^2} = 1.8$	$\sqrt{\frac{5}{3}^2 + \frac{5}{3}^2} = 2.3$	1
D	$\sqrt{1 + 3.5^2} = 3.64$	$\sqrt{\frac{1}{3}^2 + \frac{1}{3}^2} = .4$	2
E	$\sqrt{2^2 + 4.5^2} = 4.9$	$\sqrt{\frac{4}{3}^2 + \frac{4}{3}^2} = 1.9$	2

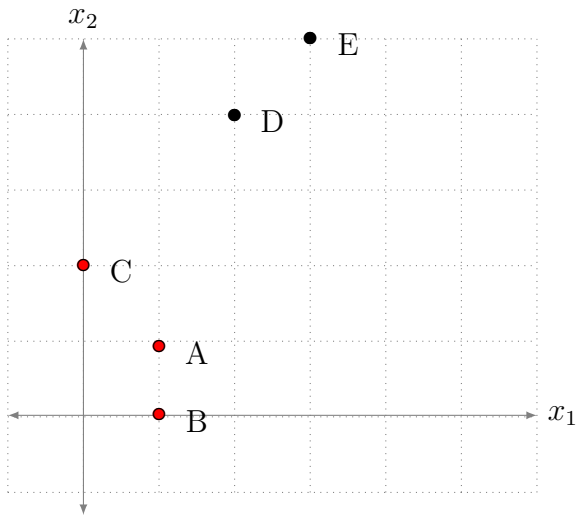
We will once again, recalculate our center means and distances from each point to the mean.

$$\boldsymbol{\mu}_1 = \frac{\sum_{i=1}^{n} r_{i1}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i1}} = [\frac{2}{3}, 1]$$
$$\boldsymbol{\mu}_2 = \frac{\sum_{i=1}^{n} r_{i2}\mathbf{x}_i}{\sum_{i=1}^{n} r_{i2}} = [\frac{5}{2}, \frac{9}{2}]$$

For $\boldsymbol{\mu}_1 = [\frac{2}{3}, 1]$, $\boldsymbol{\mu}_2 = [\frac{5}{2}, \frac{9}{2}]$

Point	$distc_1$	$distc_2$	cluster
A	$\sqrt{\frac{1}{3}^2} = .33$	$\sqrt{\frac{3}{2}^2 + \frac{7}{2}^2} = 3.8$	1
B	$\sqrt{\frac{2}{3}^2 + 1} = 1.24$	$\sqrt{\frac{3}{2}^2 + \frac{9}{2}^2} = 4.7$	1
C	$\sqrt{\frac{2}{3}^2 + 1} = 1.24$	$\sqrt{\frac{5}{2}^2 + \frac{5}{2}^2} = 3.5$	1
D	$\sqrt{\frac{2}{3}^2 + 3^2} = 3.07$	$\sqrt{\frac{1}{2}^2 + \frac{1}{2}^2} = .7$	2
E	$\sqrt{\frac{7}{3}^2 + 4^2} = 4.63$	$\sqrt{\frac{1}{2}^2 + \frac{1}{2}^2} = .7$	2

We see that the r_{ik} assignments have not changed, and therefore the means of the clusters do not change and the algorithm has converged. The final cluster assignments are shown below.



1.4 Hierarchical Agglomerative Clustering (HAC)

Hierarchical clustering constructs a tree over the data, where the leaves are individual data items, while the root is a single cluster that contains all of the data. When drawing the dendrogram, for the clustering to be valid, the distances between the two groups being merged should be monotonically increasing. The algorithm is as follows:

1. Start with n clusters, one for each data point.
2. Measure the distance between clusters. This will require an inter-cluster distance measurement that we will define shortly.
3. Merge the two 'closest' clusters together, reducing the number of clusters by 1. Record the distance between these two merged clusters.
4. Repeat step 2 until we're left with only a single cluster.

The main decision in using HAC is what the distance criterion should be between groups.

A few examples are given below.

1.4.1 The Min-Linkage Criterion

For two groups indexed by i and i', the idea is to merge groups based on the shortest distance over all possible pairs:

$$\mathrm{d}_{\min}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = \min_{i,i'} ||\mathbf{x}_i - \mathbf{x}_{i'}||_2.$$

1.4.2 The Max-Linkage Criterion

For two groups indexed by i and i', the idea is to merge groups based on the largest distance over all possible pairs:

$$\mathrm{d}_{\max}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = \max_{i,i'} ||\mathbf{x}_i - \mathbf{x}_{i'}||_2$$

1.4.3 The Average-Linkage Criterion

For two groups indexed by i and i', the idea is to average over all possible pairs between the groups:

$$\mathrm{d}_{\mathrm{avg}}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = \frac{1}{nn'} \sum_{i=1}^{n} \sum_{i'=1}^{n'} ||\mathbf{x}_i - \mathbf{x}_{i'}||_2$$

1.4.4 The Centroid-Linkage Criterion

For two groups indexed by i and i', the idea is to look at the difference between the groups' centroids:

$$\mathrm{d}_{\mathrm{cent}}(\{\mathbf{x}_i\}_{i=1}^{n}, \{\mathbf{x}_{i'}\}_{i'=1}^{n'}) = || \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \right) - \left(\frac{1}{n'} \sum_{i'=1}^{n'} \mathbf{x}_{i'} \right) ||_2 = ||\bar{\mathbf{x}}_n - \bar{\mathbf{x}}'_{n'}||_2$$

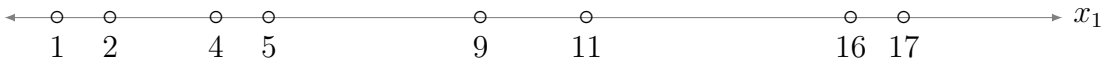
1.4.5 Exercise: K-means and HAC

What are three important differences between K-means and HAC?

K-Means has exactly K clusters whereas HAC can be used in a way where the number of clusters are determined after the fact, via inspecting the dendrogram. K-Means is randomized: the final clusters depend on the initial random centers whereas HAC is deterministic. HAC forms a hierarchy, which can provide additional understanding relative to a flat clustering.

1.4.6 Exercise: Min-Linkage and Max-Linkage Criterion

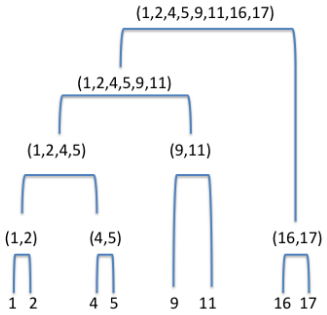
Assume the following examples lie in $\mathbb{R}$. Each example is initially in its own cluster.



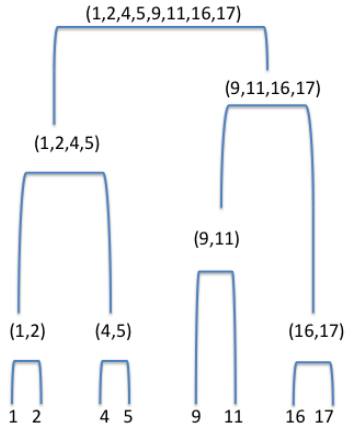
{1}{2}{4}{5}{9}{11}{16}{17}

1. Using the Min-Linkage Criterion for the HAC Algorithm, what is the clustering sequence? Draw the dendrogram.
2. Using the Max-Linkage Criterion for the HAC Algorithm, what is the clustering sequence? Draw the dendrogram.

a)
Step 1: {1} {2} {4} {5} {9} {11} {16} {17}
Step 2: {1, 2} {4, 5} {9} {11} {16, 17}
Step 3: {1, 2, 4, 5} {9, 11} {16, 17}
Step 4: {1, 2, 4, 5, 9, 11} {16, 17}
Step 5: {1, 2, 4, 5, 9, 11, 16, 17}



Min-Linkage (a)



Max-Linkage (b)

b)
Step 1: {1} {2} {4} {5} {9} {11} {16} {17}
Step 2: {1, 2} {4, 5} {9} {11} {16, 17}
Step 3: {1, 2} {4, 5} {9, 11} {16, 17}
Step 4: {1, 2, 4, 5} {9, 11} {16, 17}
Step 5: {1, 2, 4, 5} {9, 11, 16, 17}

Step 6: {1, 2, 4, 5, 9, 11, 16, 17}

1.4.7 Exercise: Clustering Complexity

What is the "big-O" complexity of HAC? What is the "big-O" complexity of K-means? Compare these.

Let's look at what each algorithm does. For K-means, we have two steps, the assignment step and the centroid update. For the assignment step, we have $O(nKm)$, where n is the size of data and m is your dimensions because you have to compare each example with each of the centroids. Then, we have $O(nm)$ for the centroid update, since we will be calculating different averages for each of the centroids. Overall, with T iterations we are left with $O(nKmT)$.

For HAC, we start with each point in its own cluster, and combine the two closest clusters until we are left with one big cluster. We get a runtime of $O(n^2m)$ because we need to calculate the pairwise distances for all n examples. In each of the T_h clustering steps, we then have $O(n^2)$ steps to use these pairwise distances and whatever linkage we're using to compute the distances between existing clusters. Generally, T_h is smaller than m and the overall complexity is $O(n^2m)$.

Comparing K-means and HAC, for large data sets we'd expect KT in K-means to be smaller than n, and thus K-means to be faster than HAC.

1.4.8 Exercise: Scaling to Large Dimensions

Explain the 'curse of dimensionality' and how it is related to HAC.

The curse of dimensionality refers to the problem where distances become meaningless in very large dimensional spaces. The problem is that the distance between two examples with some informative features but lots and lots of random features will be approximately the same (try this out in simulation if you don't see why!).

This means that non-parametric ('instance based') methods such as HAC that use pairwise distances between examples (vs between examples and prototypes in K-means) become less useful in higher dimensions.

2 Part II: Mixture Models

2.1 Motivation

Textbook sections 9.1, 9.2.

A *mixture model* is a type of probabilistic model for unsupervised learning.

Suppose you have some observed data $\{\mathbf{x}_n\}_{n=1}^N$.

Mixture models are used when you have reason to believe that each individual observation has a discrete *latent variable* $\mathbf{z}_n$ that determines the data generating process. A latent variable is some piece of data that is unknown, but influences the observed data.

Say there are K possible values for each $\mathbf{z}_n$, denoted $\{C_k\}_{k=1}^K$ where each C_k is a one-hot encoded vector of length K.

Consider the following data-generating process for each data point $\mathbf{x}_n$:

- Sample latent class $\mathbf{z}_n$ from θ, the categorical distribution over $\{C_k\}_{k=1}^K$ s.t. $p(\mathbf{z} = C_k; \theta) = \theta_k$. Call this sampled latent class C_S.
- Given that $\mathbf{z}_n = C_S$, sample $\mathbf{x}_n$ from the distribution

  $$p(\mathbf{x}|\mathbf{z} = C_S; \mathbf{w})$$

  This conditional distribution is a modeling assumption (which means we will give it to you in this class), and is specified using unknown parameters $\mathbf{w}$.

  For example, we may assume that $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z} = C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the unknown mean and covariance of the k-th cluster. (See Section 2.4 for more about Gaussian mixture models!)

Example: Say you have a dataset containing weights from a random sample of animals in a pet store. Each x_n is the animal's weight. The latent variables z_n represent what kind of animal is being weighed, so the possible values $\{C_1, C_2, \ldots, C_K\}$ may represent the categories cat, dog, bird, etc. In your model, you also use the assumption that $p(x|z = C_k; \mathbf{w}) \sim N(\mu_k, \sigma_k)$.

Exercise:

In this example, can you give an intuitive explanation of what vector θ represents? What does it mean that $p(x|z = C_k; \mathbf{w}) \sim N(\mu_k, \sigma_k)$?

θ represents the proportion of each type of animal in the pet store. For example, if $\theta_k = 0.2$ and C_k represents dogs, then 20% of all animals in the pet stores are dogs (or, if an animal is chosen at random, there is a 20% chance it will be a dog).

This class-conditional distribution means that for each type of animal, the weights of the animal are distributed normally with a fixed mean and variance for that animal.

2.2 Expectation Maximization

Textbook sections 9.3, 9.4.

Expectation maximization is a general technique for maximum-likelihood estimation used primarily for models with latent variables. Here we will show how to use EM to train a mixture model, but EM is also used for a variety of other models!

Consider a generative mixture model consisting of a latent variable $\mathbf{z}$ from a distribution $p(\mathbf{z}; \theta)$ and an observed variable $\mathbf{x}$, such that we draw $\mathbf{x}$ from a distribution $p(\mathbf{x}|\mathbf{z}; \mathbf{w})$.

We have 2 goals:

1. To compute the MLE for $\mathbf{w}$ and θ, i.e. the values of $\mathbf{w}$, θ that maximize $p(\mathbf{x}; \mathbf{w}, \theta)$.
2. To estimate the latent variable $\mathbf{z}_n$ corresponding to a particular $\mathbf{x}_n$, which in this case means maximize the distribution $p(\mathbf{z}_n|\mathbf{x}_n; \mathbf{w}, \theta)$.

Goal 2 is easy once we have an estimate of the MLE for $\mathbf{w}, \theta$, because we can apply Bayes' rule:

$$p(\mathbf{z}|\mathbf{x}; \mathbf{w}, \theta) \propto p(\mathbf{x}|\mathbf{z}; \mathbf{w}, \theta)p(\mathbf{z}; \mathbf{w}, \theta)$$

$$p(\mathbf{z}|\mathbf{x}; \mathbf{w}, \theta) \propto p(\mathbf{x}|\mathbf{z}; \mathbf{w})p(\mathbf{z}; \theta) \tag{1}$$

2.2.1 Why EM?

The likelihood of the data can be written as

$$p(\mathbf{x}; \mathbf{w}, \theta) = \sum_{\mathbf{z} \in Z} p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \theta)$$

Unfortunately calculating the MLE is often computationally intractable, because the log-likelihood is:

$$\log p(\mathbf{x}; \mathbf{w}, \theta) = \log \sum_{\mathbf{z} \in Z} p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \theta) \tag{2}$$

There is no closed form for the MLE of the log-likelihood because it is the log of a sum of expressions. We know the form of the model $p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \theta)$, but in general we cannot solve for the $(\mathbf{w}, \theta)$ which maximize the likelihood $p(\mathbf{x}; \mathbf{w}, \theta)$ in closed form.

2.2.2 The EM Algorithm

Since finding the MLE directly is difficult, we will use expectation maximization: an approximate iterative approach. The steps of the algorithm are:

1. Initialize $\mathbf{w}^{(0)}, \theta^{(0)}$ randomly.

2. *(E-step)* Use the parameters to predict the distribution $\mathbf{q}$ for each example. The vector $\mathbf{q}_n$ represents how likely it is that the latent variable $\mathbf{z}_n$ comes from each class, given our current setting for the model parameters:

$$q_{n,k} := p(\mathbf{z}_n = C_k|\mathbf{x}_n; \mathbf{w}^{(t)}, \theta^{(t)}) \propto p(\mathbf{x}_n|\mathbf{z}_n = C_k; \mathbf{w}^{(t)})p(\mathbf{z}_n = C_k; \theta^{(t)}) \tag{3}$$

3. *(M-step)* Update parameters: Choose the value of $\mathbf{w}^{(t+1)}, \theta^{(t+1)}$ that maximizes the expected complete data log likelihood (where the expectation is over the distribution calculated above):

$$\mathbf{w}^{(t+1)}, \theta^{(t+1)} = \underset{\mathbf{w},\theta}{\operatorname{argmax}} \, _{\mathbf{Z}|\mathbf{X}} \left[\sum_{n=1}^{N} \log p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \theta) \right] \tag{4}$$

4. Go back to step 2 until the log-likelihood estimate in step 3 converges.

2.2.3 Example: Gaussian Mixture Model

Lecture 14 and textbook section 9.5.

Recall from lecture the following setup:

We have data $\mathbf{x}_n \in \mathbb{R}^D$ and a latent variable $\mathbf{z}_n$ (corresponding to the cluster that the point is drawn from) such that $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z} = C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the mean and covariance of the k-th cluster. The choice of cluster is drawn from a categorical distribution with probabilities $\boldsymbol{\pi} \in [0, 1]^K$. We are able to observe the data $\mathbf{x}_n$ and want to find the cluster centers and their covariances.

The steps of EM inference applied to this problem are:

1. Randomly initialize $\boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.
2. Next, calculate the new distribution of each $\mathbf{z}_n$:

$$q_{n,k} = p(z_n = C_k|\mathbf{x}_n) \propto \pi_k \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{5}$$

   This is our new estimate of the distribution of $\mathbf{z}_n$ given the data and our estimate for $\boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_k$.

3. Find the expected complete data log-likelihood:

$$_{\mathbf{Z}|\mathbf{X}} [\mathcal{L}] =_{\mathbf{Z}|\mathbf{X}} [\sum_{n=1}^{N} \ln(p(\mathbf{x}_n, \mathbf{z}_n; \boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_k))] \tag{6}$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} q_{n,k} \ln \pi_k + q_{n,k} \ln \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{7}$$

and then optimize it for each of the parameters $\boldsymbol{\pi}, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. However, we need to be careful to remember constraints: since $\sum_k \pi_k = 1$, we must use Lagrange multipliers to optimize the parameters. We get the following update equations:

$$\pi_k^{(t+1)} = \frac{\sum_{n=1}^N q_{n,k}}{N} \tag{8}$$

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{n=1}^N q_{n,k} \mathbf{x}_n}{\sum_{n=1}^N q_{n,k}} \tag{9}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_{n=1}^N q_{n,k} (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_{n=1}^N q_{n,k}} \tag{10}$$