The Expectation-Maximization Algorithm - A Closeup

Alex Kirillova

January 30, 2019

Abstract

This paper discusses the motivation for, and the derivation of the expectation-maximization algorithm. In addition, it considers two simulation studies for the 2-dimensional case and more general 5-dimensional case as an insight to the computational application of the algorithm. As well, their purpose is to aid the development of the reader's intuition of the subject. The simulation studies are coded in Python. This text includes the structure of the code as well as visual outputs produced by the code. In case the reader wishes to reproduce the results on their own, the code is available at https://github.com/Ale-xK/EM-Algorithm. The conception of the Expectation-Maximization algorithm was inspired by a group of practical problems. As such we begin the discussion by considering an example problem and then work through the steps that lead to a solution. We will find that these steps are indeed the successive states that the algorithm takes to perform the necessary calculations.

Motivation

Consider the following made-up problem. You work at the United Nations Statistics Division and your job is to track neonatal baby weights. At the moment four countries are set up to participate in this survey:

Table 1: Population Data by Country

	Population	Population Growth Rate
Canada	36.3 M	1.2 %
China	1.379 B	0.5 %
Denmark	5.7 M	0.8 %
Ethiopia	102.4 M	2.5 %

Hospitals from each country send you a data point each time a baby is born: each data point consists of the new baby's weight in kilograms (kg). Note:

- baby weights are normally distributed,[1]
- different population levels and different population growth rates means countries will be sending data points at different rates – i.e. some countries send-in more frequently than others.

For the data to be maximally useful to your organization, you would like to know which data points come from which countries. Unfortunately this information is not included in the data that is sent to you.[2]

The ultimate goal is to fit a distribution to the available data in such a way as to obtain a (relatively accurate) estimate of the sources of your data points. Eventually this will allow us to identify subpopulations (countries) within our data for our various potential purposes of inference or prediction. At this point it is important to understand what is a Gaussian Mixture Model (GMM) in order to proceed. We include the following brief paragraph for the purposes of continuity of this section. A more thorough and rigorous explanation is provided later on.

The Gaussian Mixture Model is a weighted sum of gaussian distributions. Unlike a simple (unweighted) sum of distributions which has one agglomerate mean and one agglomerate covariance, the GMM allows for separately defined parameters (mean and covariance) per component Gaussian within the sum. The GMM is represented as a sum of its parts, wherein each part is distinguishable from the other. For this section, this explanation of the GMM should suffice – we move on.

If we can model the likelihood of the source country for each data point, and then record the most probable country for each point, then we can determine the most likely distribution for our data in conglomerate. That is, attribute each observed data point to it's most likely source so that in all, your data will appear to have subpopulations, with data points being grouped by their source country. A GMM neatly models this situation.

Since distributions are determined by their parameter values, the crux of the problem is estimating these parameters (weights, means, covariances) in order to discover the unknown (component) source distributions within the GMM.

This format of a problem is called an incomplete-data problem (not surprisingly) since we are trying to infer missing data. Solving such problems relies on iterative processes of estimating parameters. One such process is call the Expectation-Maximization Algorithm, which is the focal point of this paper.

[1] In this text, we will work with normally distributed data as our focus is on fitting Gaussian Mixture Models.

[2] This may seem like an unrealistic situation. That is because this is a contrived problem designed to demonstrate the use of the EM algorithm. The problem that the EM algorithm solves is that of estimating missing information - so we created a problem with missing information.

In our example, the *weights* of the babies are observed data and the *source country* for the newborn babies is missing - this is what we are trying to infer.

The EM algorithm works to find the maximum likelihood estimates of parameters. For this reason we review maximum likelihood estimation.

Section 1 - Maximum Likelihood Estimation (MLE)

Our goal is to fit a distribution to the available data in such a way as to obtain a (relatively accurate) estimate of the source of our data points.

To do this we estimate the parameters of each component source in the GMM.

One might wonder: why not apply the methods of maximum likelihood estimation to produce the desired estimates? This section provides that explanation - beginning with a review of MLE.

Likelihood

Discrete Case

Let X be a random variable. If its PDF f depends on a parameter θ, we write

$$\mathcal{L}(y|\theta) := f_\theta(y).$$

We call it the **likelihood function** of Y, and interpret it as the probability that an observation of y is made on Y.

X takes the value x

Note, here the notation "$|\theta$" does not refer to a conditional density. As well, at times we may consider it as a function of θ, given the observed outcome y of random variable Y.

Continuous Case

Given a random vector $\mathbf{Y} = (Y_1, \ldots, Y_n)$, we can similarly define the likelihood function as their joint density function:

$$\mathcal{L}(\mathbf{y}|\theta) := p_\theta(\mathbf{y}), \quad \mathbf{y} \in \mathbb{R}^n,$$

also interpreted as the probability that $Y_i = y_i$, $i = 1, \ldots, n$.

The $MLE(\theta)$ is that value of θ that maximizes the likelihood function - that is makes the observed data "most probable".

$$MLE(\theta) = \hat{\theta} = \arg\max_{\theta \in \Theta} \mathcal{L}(\mathbf{x}|\theta).$$

If X_i, $i = 1, ..., n$, are iid with density f, then for $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$:

$$\mathcal{L}(\mathbf{x}|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

$$\ell(\theta) = \log \mathcal{L}(\mathbf{x}|\theta) = \sum_{i=1}^{n} \log[f(x_i|\theta)]$$

$$MLE(\theta) = \hat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta).$$

Basic Example

We find the MLE for the normal distribution.

The normal distribution involves 2 params: μ, Σ.

$$\text{I.e., } \theta = (\mu, \sigma).$$

If $X_1, ..., X_n$ are iid $\mathcal{N}(\mu, \sigma^2)$, then their joint distribution is the product of their marginal distributions.

$$\begin{aligned}\log f(x_1, ..., x_n|\theta) &= \sum_{i=1}^{n} \log f(x_i|\theta) = \log\left\{\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left[\frac{x_i - \mu}{\sigma}\right]^2\right)\right\} \\ &= \ell(\mu, \sigma) \\ &= \ell(\theta) \longleftarrow \textbf{We maximize this}.\end{aligned}$$

We maximize $\ell(\theta)$.

$$\ell(\theta) = -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2}\sum_{i=1}^{n}\left[\frac{x_i - \mu}{\sigma}\right]^2$$

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu)$$

$$\frac{\partial \ell}{\partial \sigma} = -\frac{n}{\sigma} + \sigma^{-3}\sum_{i=1}^{n}(x_i - \mu)^2.$$

Setting $\frac{\partial \ell}{\partial \mu} = 0$ and $\frac{\partial \ell}{\partial \sigma} = 0$, we solve:

$$\begin{aligned}\hat{\mu} &= \overline{x} \\ \hat{\sigma} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}.\end{aligned}$$

Thus, $\hat{\theta} = (\hat{\mu}, \hat{\sigma}) = MLE(\theta)$ will give the value of θ which maximizes the probability of the data $\mathbf{x} = (x_1, \ldots, x_n)$ of being observed.

Section 2 - Gaussian Mixture Models (GMM)

We briefly considered the purpose of gaussian mixture models in the motivation section. Now, as promised, we take a closer look.

Recall the motivating example from earlier. Your job is to track neonatal baby weights.At the moment four countries are set up to participate in this survey:

Table 2: Population Data by Country

	Population	Population Growth Rate
Canada	36.3 M	1.2 %
China	1.379 B	0.5 %
Denmark	5.7 M	0.8 %
Ethiopia	102.4 M	2.5 %

Hospitals from each country send you a data point each time a baby is born: each data point consists of the new baby's weight in kilograms (kg). Note:

- baby weights are normally distributed,
- different population levels and different population growth rates means countries will be sending data points at different rates – i.e. some countries send-in more frequently than others.

For the data to be maximally useful to your organization, you would like to know which data points come from which countries. Unfortunately this information is not included in the data that is sent to you.

The ultimate goal is to fit a distribution to the available data in such a way as to obtain a (relatively accurate) estimate of the source of your data points.

We want to model a density for complete information. We don't have the complete information but we will attempt to estimate the missing information to get a model of the complete information. See tables 3 and 4 below.

Table 3: Complete Information - General Case

$X = (Y, Z)$	
Y: observed	Z: missing
y_1	z_1
$\vdots$	$\vdots$
y_n	z_n

Table 4: Complete Information - Our Example

$X = (Y, Z)$	
Y: Baby Weights	Z: Source Country
y_1	z_1
$\vdots$	$\vdots$
y_n	z_n

What is the density of the **complete** information X?

$$\begin{aligned} P(X = (y, j) \mid \theta) = P(X = (y, j)) &= P(Y = y, Z = j) \\ &\overset{*}{=} P(Y = y | Z = j) \cdot P(Z = j) \quad \textit{by Bayes' Rule } [3] \\ &\overset{**}{=} \phi(y \mid \mu_j, \Sigma_j) \cdot w_j \end{aligned}$$

$$\overset{*}{P(A \cap B) = P(A)P(B)}$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \quad \Longleftrightarrow \quad P(A|B) = \frac{P(A)P(B)}{P(B)} = P(A)$$

$$\therefore \quad \frac{P(Y_i = y_i, Z_i = j \mid \theta)}{P(Z_i = j \mid \theta)} = P(Y_i = y_i \mid Z_i = j, \theta)$$

$$\overset{**}{\because \quad P(Z_i = j \mid \theta) = P(Z_i = j) = w_j}$$

$$\text{and} \quad P(Y_i = y_i \mid Z_i = j, \theta) = P(Y_i = y_i \mid Z_i = j) = \phi(y_i \mid \mu_j, \Sigma_j)$$

The density of the complete information, X, is the probability of observing both Y and Z.
This is what we are trying to "learn", or in other words, this is what we are trying to **model as accurately as possible**.

What information have we **observed** (is *not* missing)? What is the density?

$$\begin{aligned} P(Y_i = y_i) &= \sum_{i=1}^{n} P(Y_i = y_i, Z_i = j) \\ &= \sum_{j=1}^{k} w_j \cdot \phi(y_i | \mu_j, \Sigma_j) \end{aligned}$$

The probability of any point, i, is the sum of its respective probabilities of being observed from each source, j, as shown above.

The probability of a particular source, j, generating some specified point, i, is $P(Z_i = j \mid y_i, \theta^{(m)})$.

From Bayes' Rule,

$$\begin{aligned} P(Z_i = j \mid y_i, \theta^{(m)}) &= \frac{P(Z_i = j, Y_i = y_i \mid \theta^{(m)})}{P(Y_i = y_i \mid \theta^{(m)})} \\ &\overset{**}{=} \frac{w_j \cdot \phi(y_i | \mu_j, \Sigma_j)}{\sum_{l=1}^{k} w_l \cdot \phi(y_i | \mu_l, \Sigma_l)}. \end{aligned}$$

We write

$$\gamma_{ij}^{(m)} \triangleq P(Z_i = j \mid y_i, \theta^{(m)}),$$

where $\gamma_{ij}^{(m)}$ is the probability of the j^{th} source being the one for the i^{th} point, and $\sum_{j=1}^{k} \gamma_{ij}^{(m)}$.

[3] Bayes' Rule: $P(A|B) = \frac{P(A \cap B)}{P(B)}$

MLE for GMM

Now that we have seen the gaussian mixture model up close, we apply maximum likelihood estimation to estimate the parameters.

A GMM has 3 parameters: μ_j, Σ_j, w_j.

Given observations $\mathbf{Y} = (Y_1, ..., Y_n)$, we estimate $\theta = \{(\mu_j, \Sigma_j, w_j)\}_{j=1}^{k}$, for a GMM with k sources.

As we did with the simple e.g. we employ the MLE to find the parameters theta that make the observed Y_i's most likely. Recall: $\ell(\theta) = \log \mathcal{L}(\mathbf{Y}|\theta) = \sum_{i=1}^{n} \log[P(Y_i = y_i|\theta)]$.

$$\begin{aligned} MLE(\theta) &= arg\max_{\theta} \ell(\theta) \\ &= arg\max_{\theta} \log \mathcal{L}(\mathbf{Y}|\theta) \\ &= arg\max_{\theta} \sum_{i=1}^{n} \log[P(Y_i = y_i|\theta)] \\ &= arg\max_{\theta} \sum_{i=1}^{n} \log\Big[\sum_{l=1}^{k} w_l \cdot \phi(y_i|\mu_l, \Sigma_l)\Big] \end{aligned}$$

Conclusion: it is difficult!

Typically it is nearly impossible to generate parameter estimates using the maximum likelihood estimation method as above.

Our solution is to create an analog to the MLE method. We create a probabilistic model of all possible situations and take the average, or expectation over all of those. By 'situation' we mean probability assignment of a source to a particular point, across all points and all sources. This is analogous to generating a log-likelihood function. Finally we maximize this to produce the overall most likely probability assignment across all points and source.

This method applied iteratively is called the Expectation-Maximization Algorithm. This is the topic of our next section.

EM for GMM

Since we cannot exactly take the MLE of θ, as shown above, we use an analogous method as follows:

- Take the log-likelihood of the i^{th} point coming from the j^{th} source, over all points and all sources:

  $$\log p(Z_i = j, Y_i = y_i \mid \theta^{(m)}).$$

  Intuitively speaking, this expression represents the set of all possible scenarios of our n data points coming from the k sources, and the probability assigned to each scenario.

- Next, take the expectation of the log-likelihood:

  $$E_{Z_i|y_i,\theta^{(m)}}\Big[\log p(Z_i = j, Y_i = y_i \mid \theta^{(m)})\Big].$$

  Loosely, here we treat the log-likelihood scenarios as a normally distributed data set, and find the expression for its mean (expectation). We do this to obtain an expression for a single likelihood scenario, a opposed to as set, so that we have an expression to then maximize.

- We call the above expression the Q function and it is analogous to the likelihood of θ in the MLE method.

$$
\begin{aligned}
Q(\theta|\theta^{(m)}) &= \mathrm{E}_{Z_i|y_i,\theta^{(m)}}\Big[\log p(Z_i = j, Y_i = y_i \mid \theta^{(m)})\Big] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k} P\Big(Z_i \mid y_i, \theta^{(m)}\Big) \log p\Big(y_i, Z_i \mid \theta\Big) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(m)} \log\Big(w_j\, \phi(y_i, Z_i \mid \theta)\Big) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(m)} \Big(\log w_j - \frac{1}{2}\log|\Sigma_j| - \frac{1}{2}(y_i-\mu_j)^T\Sigma_j^{-1}(y_i-\mu_j)\Big) + C \\
&= \sum_{j=1}^{k} n_j^{(m)}\Big(\log w_j - \frac{1}{2}\log|\Sigma_j|\Big) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\Big(y_i-\mu_j\Big)^T\Sigma_j^{-1}\Big(y_i-\mu_j\Big) + C \\
&= \Big(\sum_{j=1}^{k} n_j^{(m)}\log w_j\Big) \quad - \Big(\frac{1}{2}\sum_{j=1}^{k} n_j^{(m)}\log|\Sigma_j|\Big) \quad - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\Big(y_i-\mu_j\Big)^T\Sigma_j^{-1}\Big(y_i-\mu_j\Big),
\end{aligned}
$$

where $n_j^{(m)} \triangleq \sum_{i=1}^{n}\gamma_{ij}^{(m)}$.

- Next we maximize the Q function with respect to each variable: w_j, Σ_j, μ_j.

  We start with w_j.
  This is a constrained optimization problem since $\sum_{j=1}^{k} w_j = 1$. Therefore we apply the method of Lagrange multipliers.

  Let $\quad g(w_j) = \Big(\sum_{j=1}^{k} w_j\Big) - 1, \qquad$ and $\qquad \dfrac{\partial Q}{\partial w_j} = \lambda \cdot \dfrac{\partial g}{\partial w_j}.$

  Since only the first term in Q is a function of w_j, it follows:

$$
\frac{\partial Q}{\partial w_j} = \frac{\partial[\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\log w_j]}{\partial w_j} = \lambda \cdot \frac{\partial[\sum_{j=1}^{k} w_j - 1]}{\partial w_j} = \lambda \cdot \frac{\partial g}{\partial w_j}
$$

$$
\frac{\partial\{\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\log w_j + \lambda \cdot [\sum_{j=1}^{k} w_j - 1]\}}{\partial w_j} = 0
$$

$$
\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\left[\frac{\partial}{\partial w_j}\log w_j\right] + \lambda \cdot \sum_{j=1}^{k}\left[\frac{\partial}{\partial w_j} w_j\right] = 0
$$

$$
\frac{\sum_{i=1}^{n}\gamma_{ij}^{(m)}}{w_j} + \lambda = 0.
$$

  Solving for w_j :

$$
\boxed{w_j^{(m+1)} = \frac{\sum_{i=1}^{n}\gamma_{ij}^{(m)}}{\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}} = \frac{n_j^{(m)}}{n}.}
$$

Alternatively we can use Gibbs' inequality to show the same result in a much more concise way. First we take a look at the Gibbs' inequality. Then we will apply it to maximizing w_j.

Gibbs' Inequality

Suppose $P = \{p_1, ..., p_n\}$ is a probability distribution.
Then for any other probability distribution $Q = \{q_1, ..., q_n\}$ the following inequality between positive quantities (since the p_i and q_i are positive numbers less than one) holds,

$$-\sum_{i=1}^{n} p_i \log_2 p_i \leq -\sum_{i=1}^{n} p_i \log_2 q_i$$

with equality if and only if $p_i = q_i$ for all i.

Proof

For simplicity we prove the result for $\ln a$. We can do this since $\log_2 a = \dfrac{\ln a}{\ln 2}$.

Let I denote the set of all i for which $p_i \neq 0$.
Since $\ln x \leq x - 1$ for all $x > 0$, with equality if and only if $x = 1$, we have:

$\diamond$

If C is an invertible matrix then,

$$\begin{aligned} -\sum_{i\in I} p_i \ln \frac{q_i}{p_i} &\geq -\sum_{i\in I} p_i \left(\frac{q_i}{p_i} - 1\right) \\ &= -\sum_{i\in I} q_i + \sum_{i\in I} p_i \\ &= 0. \end{aligned}$$

Then,

$$\begin{aligned} -\sum_{i\in I} p_i \ln \frac{q_i}{p_i} &\geq 0 \longleftarrow \text{By } \diamond \\ -\sum_{i\in I} p_i (\ln q_i - \ln p_i) &\geq 0 \\ -\sum_{i\in I} p_i \ln q_i + \sum_{i\in I} p_i \ln p_i &\geq 0 \\ -\sum_{i\in I} p_i \ln q_i &\geq -\sum_{i\in I} p_i \ln p_i. \end{aligned}$$ $\square$

Now we show how to use the Gibbs' inequality to maximize w_j to obtain $w_j^{(m+1)}$.

Recall:
$Q(\theta|\theta^{(m)}) = \left(\sum_{j=1}^{k} n_j^{(m)} \log w_j\right) - \left(\frac{1}{2}\sum_{j=1}^{k} n_j^{(m)} \log|\Sigma_j|\right) - \left(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(m)} \left(y_i - \mu_j\right)^T \Sigma_j^{-1} \left(y_i - \mu_j\right)\right)$, and

$$n_j^{(m)} = \sum_{i=1}^{n} \frac{w_j \cdot \phi(y_i \mid \mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^{k} w_l^{(m)} \cdot \phi(y_i \mid \mu_l^{(m)}, \Sigma_l^{(m)})} = \sum_{i=1}^{n} p(Z_i = j \mid y_i, \theta^{(m)}).$$

Define $Q^*(\theta|\theta^{(m)}) \triangleq \sum_{j=1}^{k} n_j^{(m)} \log w_j$.

That is, let Q^* be the first term in $Q(\theta|\theta^{(m)})$. We want to work with the first term only because the other two terms do not depend on our variable of interest, w_j.

Now let $p_j = \dfrac{n_j^{(m)}}{\sum_{l=1}^{k} n_l^{(m)}}$, and let $q_j = w_j$.

From the Gibbs' inequality it follows that the maximum of $\sum_{j=1}^{k} n_j^{(m)} \log w_j$ is achieved when $q_j = p_j$. I.e. when $w_j^{(m+1)} = \dfrac{n_j^{(m)}}{\sum_{l=1}^{k} n_l^{(m)}} = \dfrac{n_j^{(m)}}{n}$ for all $j = 1, ..., k$.

And so we arrive at the same result as we did previously when we used the method of Lagrange multipliers:

$$\boxed{w_j^{(m+1)} = \frac{n_j^{(m)}}{n}\,.}$$

Therefore the expectation maximization for learning an optimal mixture of k fixed models reduced to iteratively solving for the k estimated weights $w_j^{(m+1)}$.

Now we maximize $Q(\theta|\theta^{(m)})$ with respect to μ_j.

Recall: $$Q(\theta|\theta^{(m)}) = \left(\sum_{j=1}^{k} n_j^{(m)} \log w_j\right) - \left(\frac{1}{2}\sum_{j=1}^{k} n_j^{(m)} \log|\Sigma_j|\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(m)} \left(y_i - \mu_j\right)^T \Sigma_j^{-1} \left(y_i - \mu_j\right).$$

Then the partial of Q with respect to μ_j is:

$$\begin{aligned}
\frac{\partial Q(\theta|\theta^{(m)})}{\partial \mu_j} &= \sum_{i=1}^{n}\sum_{j=1}^{k} \gamma_{ij}^{(m)} \Sigma_j^{-1}(y_i - \mu_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k} \Sigma_j^{-1}(\gamma_{ij}^{(m)} y_i - \gamma_{ij}^{(m)} \mu_j) \\
&= \sum_{j=1}^{k} \Sigma_j^{-1}\left(\sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i - \sum_{i=1}^{n} \gamma_{ij}^{(m)} \mu_j\right). \qquad n_j^{(m)}
\end{aligned}$$

Maximizing,

$$0 = \Sigma_j^{-1}\Big(\sum_{i=1}^{n}\gamma_{ij}^{(m)}y_i - n_j^{(m)}\mu_j\Big)$$

$$\cancel{\Sigma_j^{-1}}\, n_j^{(m)}\mu_j^{(m+1)} = \cancel{\Sigma_j^{-1}}\sum_{i=1}^{n}\gamma_{ij}^{(m)}y_i$$

$$n_j^{(m)}\ \mu_j^{(m)} = \sum_{i=1}^{n}\gamma_{ij}^{(m)}y_i$$

$$\mu_j^{(m)} = \frac{1}{n_j^{(m)}}\sum_{i=1}^{n}\gamma_{ij}^{(m)}y_i \qquad \text{for all } j = 1, ..., k.$$

$$\boxed{\mu_j^{(m+1)} = \tfrac{1}{n_j^{(m)}}\sum_{i=1}^{n}\gamma_{ij}^{(m)}y_i \qquad \text{for all } j = 1, ..., k}$$

Now we maximize $Q(\theta|\theta^{(m)})$ with respect to Σ_j.

Recall:

$$Q(\theta|\theta^{(m)}) \ = \Big(\sum_{j=1}^{k} n_j^{(m)}\log w_j\Big) \ - \ \Big(\frac{1}{2}\sum_{j=1}^{k} n_j^{(m)}\log|\Sigma_j|\Big) \ - \ \Big(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\Big(y_i-\mu_j\Big)^T\Sigma_j^{-1}\Big(y_i-\mu_j\Big)\Big).$$

Notice that the first term in the equation above does not depend on Σ_j, therefore we need only consider the other two terms when maximizing with respect to Σ_j.

Then,

$$\frac{\partial Q(\theta|\theta^{(m)})}{\partial\Sigma_j} = -\frac{\partial\Big(\frac{1}{2}\sum_{j=1}^{k} n_j^{(m)}\log|\Sigma_j|\Big)}{\partial\Sigma_j} \quad - \frac{\partial\Big(\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}^{(m)}\ (y_i-\mu_j)^T\Sigma_j^{-1}(y_i-\mu_j)\Big)}{\partial\Sigma_j}$$

$$= -\frac{1}{2}\left[\frac{\partial\Big(\sum_{j=1}^{k} n_j^{(m)}\log|\Sigma_j|\Big)}{\partial\Sigma_j} + \frac{\partial\Big(\sum_{i=1}^{n}\gamma_{ij}^{(m)}(y_i-\mu_j)^T\Sigma_j^{-1}(y_i-\mu_j)\Big)}{\partial\Sigma_j}\right]$$

$$\overset{*}{=} -\frac{1}{2}\left[\qquad n_j^{(m)}\Sigma_j^{-T} \qquad + \frac{\partial\Big(\sum_{i=1}^{n}\gamma_{ij}^{(m)}(y_i-\mu_j)^T\Sigma_j^{-1}(y_i-\mu_j)\Big)}{\partial\Sigma_j}\right]$$

$$\overset{**}{=} -\frac{1}{2}\left[\qquad n_j^{(m)}\Sigma_j^{-T} \qquad + \quad \sum_{i=1}^{n}\gamma_{ij}^{(m)}(\Sigma_j^{-1})(y_i-\mu_j)(y_i-\mu_j)^T(\Sigma_j^{-1})\right].$$

*

If C is an invertible matrix then,

$$\begin{aligned}\frac{\partial(\log|C|)}{\partial C} &= \frac{\partial(\log|C^{-1}|)}{\partial C} \\ &= (C^T)^{-1} \\ &= C^{-T}.\end{aligned}$$

**

Let X be any matrix. Then from the identity,

$$\frac{\partial X^{-1}}{\partial x} = -X\frac{\partial X}{\partial x}X^{-1}$$

we have,

$$\frac{\partial(a^TC^{-1}b)}{\partial C} = (-C)^{-1}a\ b\ C^{-1}.$$

Maximizing,

$$\begin{aligned}0 &= -\frac{1}{2}\left[n_j^{(m)}\Sigma_j^{-T} + \sum_{i=1}^{n}\gamma_{ij}^{(m)}(\Sigma_j^{-1})(y_i-\mu_j)(y_i-\mu_j)^T(\Sigma_j^{-1})\right] \\ &= n_j^{(m)}\Sigma_j^{-T} + \sum_{i=1}^{n}\gamma_{ij}^{(m)}(\Sigma_j^{-1})(y_i-\mu_j)(y_i-\mu_j)^T(\Sigma_j^{-1}) \\ &= \sum_{i=1}^{n}\gamma_{ij}^{(m)}\left(\Sigma_j^{-1} - (\Sigma_j^{-1})(y_i-\mu_j)(y_i-\mu_j)^T(\Sigma_j^{-1})\right) \\ &= \sum_{i=1}^{n}\gamma_{ij}^{(m)}\,\Sigma_j\left(\Sigma_j^{-1} - (\Sigma_j^{-1})(y_i-\mu_j)(y_i-\mu_j)^T(\Sigma_j^{-1})\right) \\ &= \sum_{i=1}^{n}\gamma_{ij}^{(m)}\left(I - (y_i-\mu_j)(y_i-\mu_j)^T(\Sigma_j^{-1})\right)\Sigma_j \\ &= \sum_{i=1}^{n}\gamma_{ij}^{(m)}\left(\Sigma_j - (y_i-\mu_j)(y_i-\mu_j)^T\right) \\ &= \sum_{i=1}^{n}\cancel{\gamma_{ij}^{(m)}}\,\Sigma_j - \sum_{i=1}^{n}\gamma_{ij}^{(m)}\left(y_i-\mu_j^{(m+1)}\right)\left(y_i-\mu_j^{(m+1)}\right)^T \quad (\cancel{\gamma_{ij}^{(m)}} \to n_j^{(m)})\end{aligned}$$

$$\Sigma_j^{(m+1)} = \frac{1}{n_j^{(m)}}\sum_{i=1}^{n}\gamma_{ij}^{(m)}\left(y_i-\mu_j^{(m+1)}\right)\left(y_i-\mu_j^{(m+1)}\right)^T \qquad \text{for all } j=1,...,k$$

This whole process is summarized in the next section.

Section 3 - Expectation-Maximization (EM) Algorithm

This section is comprised of two parts. First we discuss the implementation of the algorithm, second we explore examples with visuals.

Implementation

The following table summarizes the steps taken by the EM algorithm.

Recall that the EM algorithm is an iterative process which re-estimates the parameter $\theta = \{w, \mu, \Sigma\}$, where w represents the weights parameter of the k distributions, μ represents the means parameter of the k distributions, and Σ represents the covariance parameter of the k distributions.

We denote by $\theta^{(m)} = \{w^{(m)}, \mu^{(m)}, \Sigma^{(m)}\}$ the parameter estimates in the m-th iteration.

Table 5: EM algorithm iterative steps for estimating GMM parameters

Step 1	**Initialization** - Choose initial parameter estimates $w_j^{(0)}$, $\mu_j^{(0)}$, $\Sigma_j^{(0)}$, $j = 1, ..., k$. Compute initial log-likelihood: $$\ell^{(0)} = \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} w_j^{(0)} \phi(y_i | \mu_j^{(0)}, \Sigma_j^{(0)}) \right)$$.
Step 2	**E-Step** - For $j = 1, ..., k$ compute $$\gamma_{ij}^{(0)} = \frac{w_j^{(m)} \phi(y_i | \mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^{k} w_l^{(m)} \phi(y_i | \mu_l^{(m)}, \Sigma_l^{(m)})}, \; i = 1, ..., n,$$ and $$n_j^{(m)} = \sum_{i=1}^{n} \gamma_{ij}^{(m)}.$$
Step 3	**M-Step** - For $j = 1, ..., k$ compute the new estimates of parameters $$w_j^{(m+1)} = \frac{n_j^{(m)}}{n},$$ $$\mu_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i,$$ $$\Sigma_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} (y_i - \mu_j^{(m+1)})(y_i - \mu_j^{(m+1)})^T.$$
Step 4	**Convergence Check** - Compute the new log-likelihood $$\ell^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} w_j^{(m+1)} \phi(y_i | \mu_j^{(m+1)}, \Sigma_j^{(m+1)}) \right)$$
Iteration Step	**Proceed to Step 2** if $\lvert l^{(m+1)} - l^{(m)} \rvert > \delta$ for some predefined threshold δ. Otherwise the algorithm ends.

Initialization - A remark on Step 1 of Table 5

In order to start, the EM algorithm must begin with some initial values of the parameters. It uses these initial values to then create new estimates in its iterations. Finding these initial parameter values is called *initialization*.

For the weights parameter, it is common practice to make them equal for the k sources. The k means can be chosen randomly. That is, choose k random data points from the data set. Finally the covarainces parameter is commonly initialized as the identity.

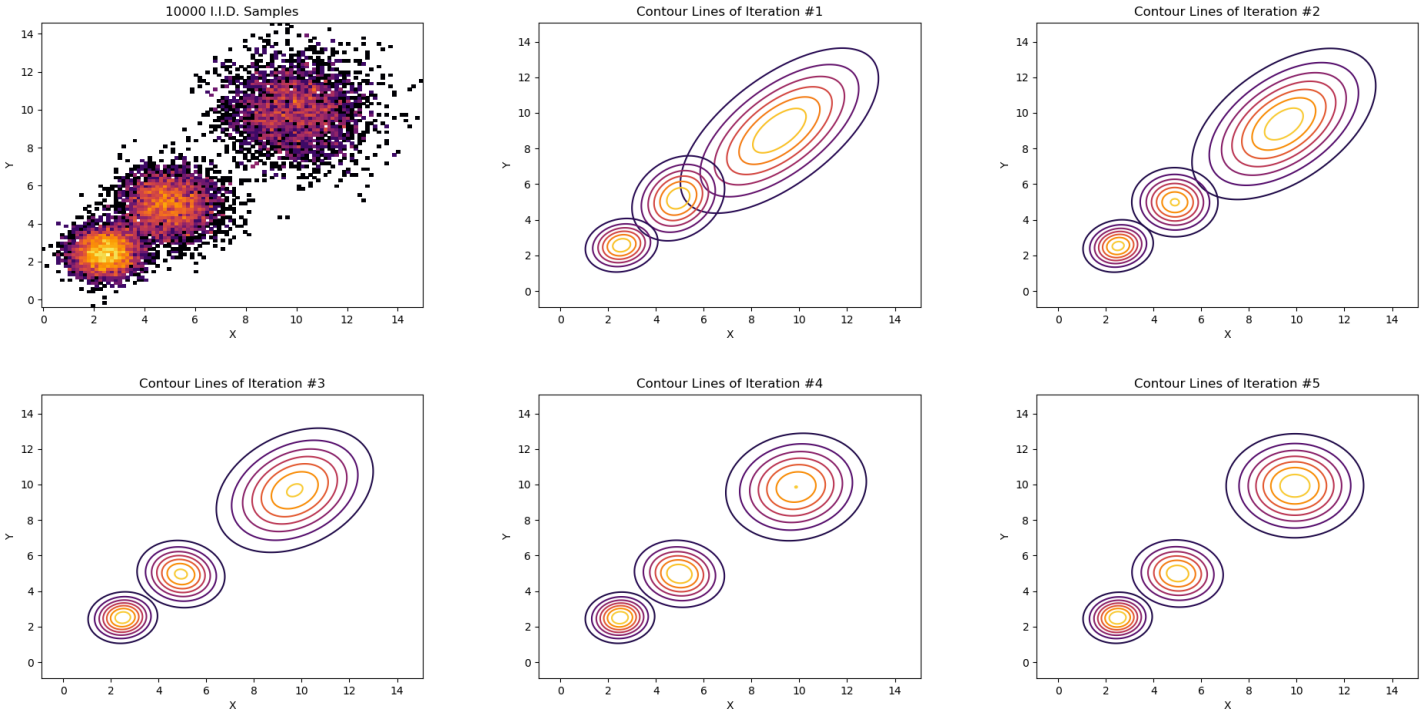
Examples

We explore two examples in this section. The first is the case where two-dimensional data and k=3 (three sources of data). The second is five-dimensional and k=4. Both examples have been arbitrarily generated for the sake of visual intuition and to demonstrate the efficiency of the EM algorithm.

The code which generates these examples is available at https://github.com/Ale-xK/EM-Algorithm.

Disclaimer. Running this code will not produce the exact same results as in the examples below since the code generates an entirely new data set every time it is run. The code is written wholly from scratch - that is, no Scikit-learn or other machine learning libraries have been used. As such, the code runs slowly. The more efficient way of implementing the algorithm would be to use the Scikit-learn library, however in the context of this paper, that would defeat the purpose of demonstrating how the algorithm iterates.

Example 1 - 2D



Example 2 - 5D

Unlike in the previous two-dimensional example, it is not possible to visualize data in five dimensions. In the absence of neat graphs, we can still make use of the algorithm to get a sense of how the data is distributed by using displayed parameter values.

Here, we generate 100 data points; they are 5-dimensional and come from 4 sources, i.e. n=100, k=4. The tolerance has been set to 0.05 and the maximum number of allowed iterations is set to 10.

Below is the output of the code:

```
GENERATING DATA SET...

Number of data points per source:
[24. 32. 25. 19.]

------------------------------------------------------------
INITIALIZING PARAMETERS...

Initial weights:
[0.25, 0.25, 0.25, 0.25]

Initial covariance:
[[[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 1. 0. 0.]
[0. 0. 0. 1. 0.]
[0. 0. 0. 0. 1.]]

[[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 1. 0. 0.]
[0. 0. 0. 1. 0.]
[0. 0. 0. 0. 1.]]

[[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 1. 0. 0.]
[0. 0. 0. 1. 0.]
[0. 0. 0. 0. 1.]]

[[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 1. 0. 0.]
[0. 0. 0. 1. 0.]
[0. 0. 0. 0. 1.]]]

Initial means:
[[ 2.18191292 2.77934976 4.22140408 1.49908169 1.5974763 ]
[ 6.41764894 3.85289076 6.18672566 5.29880409 4.86292395]
[ 0.94624947 1.47322441 3.08465079 3.46667458 1.5981541 ]
[-0.00826912 -2.55400342 5.84639849 2.95189482 4.97269828]]

------------------------------------------------------------
Iteration: 1

E-step:
Constructing phi matrix...
Initial log likelihood = 13.255139194546334
Constructing gamma matrix...
Constructing n matrix...

M-Step:
Constructing weights matrix...

[0.297309433110798, 0.2002740362206904, 0.41731748593076745, 0.08509904473774414]
Constructing means matrix...
```

```
[[ 3.16515017 3.04311859 3.16758348 2.37031825 2.77133353]
[ 4.53685831 4.61749529 4.42531969 4.34619225 4.21629728]
[ 0.77187334 1.12505992 2.10277187 3.43249791 3.24419598]
[ 0.41437483 -1.15052308 4.19046326 2.75798012 5.04767193]]
Constructing covariance matrix...

[[[ 1.63588092 0.19449542 -0.02663128 0.47611177 0.12786157]
[ 0.19449542 1.35291213 0.11733669 0.36239174 0.10024185]
[-0.02663128 0.11733669 1.50891509 0.04911976 0.00889388]
[ 0.47611177 0.36239174 0.04911976 0.69347609 0.19118001]
[ 0.12786157 0.10024185 0.00889388 0.19118001 1.1881292 ]]

[[ 0.73620282 -0.57704597 -0.20857418 0.07063184 -0.24724549]
[-0.57704597 1.47066034 -0.20885811 0.4793799 0.24251568]
[-0.20857418 -0.20885811 2.0846135 0.10477755 0.28396676]
[ 0.07063184 0.4793799  0.10477755 1.24684406 -0.68728474]
[-0.24724549 0.24251568 0.28396676 -0.68728474 2.15861669]]

[[ 4.27973931 0.69751055 0.85570612 0.75081115 -0.81214843]
[ 0.69751055 2.50094634 0.06115647 0.08166386 0.05983026]
[ 0.85570612 0.06115647 2.35499385 0.45768571 -1.03974291]
[ 0.75081115 0.08166386 0.45768571 1.9368195 0.05444664]
[-0.81214843 0.05983026 -1.03974291 0.05444664 2.0304633 ]]

[[ 2.52298374 -0.43381005 0.71170677 1.65997231 -1.51575798]
[-0.43381005 1.41657248 0.02005171 0.42852531 0.72534423]
[ 0.71170677 0.02005171 2.16484031 1.03113617 -0.88341502]
[ 1.65997231 0.42852531 1.03113617 4.2065106 -1.56601085]
[-1.51575798 0.72534423 -0.88341502 -1.56601085 2.78551691]]]

Convergence Step:
Recalculating phi matrix...
Convergence = 4.13260548535799

------------------------------------------------------------
Iteration: 2

E-step:
Constructing phi matrix...
Constructing gamma matrix...
Constructing n matrix...

M-Step:
Constructing weights matrix...

[0.30421255667589886, 0.20848289615624044, 0.40535612564768875, 0.08194842152017211]
Constructing means matrix...

[[ 3.15735632 2.92541799 3.02193218 2.34035619 2.76868727]
[ 4.48020869 4.64128786 4.37075999 4.25619506 4.20212804]
[ 0.62068894 1.07978185 2.13740583 3.44401065 3.28546357]
[ 0.74276827 -1.14905402 4.45655944 3.01323538 4.90120732]]
Constructing covariance matrix...

[[[ 1.35939787 0.29951022 -0.06327805 0.43118314 0.15575271]
[ 0.29951022 1.17468457 0.18493367 0.34915217 0.07266691]
[-0.06327805 0.18493367 1.42036387 -0.07353722 0.02573925]
[ 0.43118314 0.34915217 -0.07353722 0.55289626 0.24764041]
[ 0.15575271 0.07266691 0.02573925 0.24764041 1.12724678]]

[[ 0.73226116 -0.60288179 -0.11107286 0.11662327 -0.20921428]
```

```
[-0.60288179 1.36901004 -0.11750507 0.54064578 0.12453145]
[-0.11107286 -0.11750507 2.0479868 0.14292883 0.35770228]
[ 0.11662327 0.54064578 0.14292883 1.29811752 -0.60390209]
[-0.20921428 0.12453145 0.35770228 -0.60390209 2.09967304]]

[[ 4.42117164 0.70154658 0.94272068 0.95016738 -0.86351627]
[ 0.70154658 2.67165642 0.17338886 0.21340096 -0.01087728]
[ 0.94272068 0.17338886 2.58292814 0.38777979 -1.19276732]
[ 0.95016738 0.21340096 0.38777979 1.949844  0.00611593]
[-0.86351627 -0.01087728 -1.19276732 0.00611593 2.19429707]]

[[ 2.7016292 -0.233764   0.74352868 2.18157102 -2.20298316]
[-0.233764   1.51500359 0.23107004 0.60096823 0.61254667]
[ 0.74352868 0.23107004 1.60413981 1.40629506 -0.80269818]
[ 2.18157102 0.60096823 1.40629506 5.0701606 -2.2614036 ]
[-2.20298316 0.61254667 -0.80269818 -2.2614036 3.00845655]]]

Convergence Step:
Recalculating phi matrix...
Convergence = 0.08616720378248566

------------------------------------------------------------
Iteration: 3

E-step:
Constructing phi matrix...
Constructing gamma matrix...
Constructing n matrix...

M-Step:
Constructing weights matrix...

[0.2991284026217261, 0.21871020316434925, 0.40268297085807275, 0.07947842335585198]
Constructing means matrix...

[[ 3.12925339 2.88589317 2.94936223 2.31020817 2.75037928]
[ 4.47456047 4.6302967  4.31098136 4.20567788 4.15844926]
[ 0.54743787 1.02853116 2.16080366 3.45238518 3.30335617]
[ 0.90463003 -1.11988488 4.61690731 3.03478651 4.89885314]]
Constructing covariance matrix...

[[[ 1.25224275 0.32289573 -0.09766276 0.38345651 0.14713602]
[ 0.32289573 0.97008368 0.23422936 0.27657826 0.13094622]
[-0.09766276 0.23422936 1.37709335 -0.09741373 -0.01156503]
[ 0.38345651 0.27657826 -0.09741373 0.48548113 0.27360185]
[ 0.14713602 0.13094622 -0.01156503 0.27360185 1.09586379]]

[[ 0.73558869 -0.64348682 -0.09752183 0.10481501 -0.17933654]
[-0.64348682 1.44254756 -0.105153  0.57670711 0.0833522 ]
[-0.09752183 -0.105153  2.13980283 0.1797092  0.39374606]
[ 0.10481501 0.57670711 0.1797092  1.30036326 -0.53781408]
[-0.17933654 0.0833522  0.39374606 -0.53781408 2.09838521]]

[[ 4.31073019 0.62835593 0.97622906 1.01991205 -0.85646026]
[ 0.62835593 2.7090214  0.20427314 0.26517704 -0.03368292]
[ 0.97622906 0.20427314 2.58352003 0.33488642 -1.1881093 ]
[ 1.01991205 0.26517704 0.33488642 1.94905056 -0.007156 ]
[-0.85646026 -0.03368292 -1.1881093 -0.007156 2.21534501]]

[[ 2.93660745 -0.13015747 0.80924395 2.35155303 -2.54068227]
[-0.13015747 1.54355664 0.38291405 0.63952258 0.49072569]
```

```
[ 0.80924395 0.38291405 1.46681015 1.61077443 -0.99927347]
[ 2.35155303 0.63952258 1.61077443 5.39466279 -2.485529 ]
[-2.54068227 0.49072569 -0.99927347 -2.485529 3.13139357]]]

Convergence Step:
Recalculating phi matrix...
Convergence = 0.04146725140603458

------------------------------------------------------------
Computation Complete!
Success after 3 iterations!
```

Notice that the output below follows the steps in Table 5.