EM (Expectation-Maximization) Algorithm with An Application for Gaussian Mixture Model

Frances Lin

Fall 2020

Abstract

We first discuss the conditions under which EM algorithm can be used to find (local) maximum likelihood estimate (MLE) of parameter, as compared to other iterative method such as Newton-Raphson, Fisher's scoring, and IRLS (iteratively reweighted least squares). Then, we briefly review MLE and posterior probability. Next, we introduce the EM algorithm in the context of the Gaussian mixture model and expand the E-step and the M-step of the algorithm in details. Finally, we apply the algorithm to a simulated dataset that follows a two-component Gaussian mixture distribution and evaluate its performance.

Introduction

Optimization problems occur in a variety of fields such as mechanics, economics and finance, various engineering fields, operations research, and machine learning. Many optimization methods have been proposed in math, computer science and statistics as a result. Some of the methods include Newton-Raphson, Fisher scoring, IRLS (iteratively reweighted least squares) for glm (generalized linear model), etc. Among them, EM algorithm, short for "Expectation-Maximization" algorithm, first invented by computer scientist and later generalized by Dempster, Laird, and Rubin in a classic 1977 paper, is an iterative method that is particular useful for finding MLE for incomplete or missing data or when maximizing the target likelihood function is challenging.

As the name implies, "E" for "Expectation" and "M" for "Maximization", once initialized, EM algorithm iterate between the E-step and the M-step untill convergence. Comparing to method such as Newton-Raphson, EM algorithm requires simple implementation and has stable convergence. Comparing to IRLS, EM algorithm becomes IRLS in certain models. Note that EM algorithm is an umbrella term for a class of algorithm that iterates between expectation and maximization. Some of the applications include EM algorithm for missing data, for grouped, censored, or truncated data, for finite mixture models, for factor analysis, etc. For motivating example, we choose to focus on EM algorithm for Gaussian mixture models. For illustration, we apply EM algorithm to a simulated dataset that follows a two-component Gaussian mixture distribution.

Review

MLE

MLE (maximum likelihood estimation) is a commonly used method in statistics for finding estimate of parameter, as maximum likelihood estimator has a lot of nice properties. Estimate is found by maximizing the likelihood function

$$\mathcal{L}(\theta|X) = \prod_{i=1}^{n} p(X|\theta)$$

Posterior Probability

Give that the data X has a likelihood $p(X|Z)$ and a prior belief that the probability distribution is $p(Z)$, then the posterior probability is defined as

$$p(Z|X) = \frac{p(X|Z)p(Z)}{p(X)}$$

A key feature of the EM algorithm is that it makes use of the posterior probability $p(Z|X)$ (as supposed to $p(Z)$) in the E-step.

Setup

Gaussian mixture model

Suppose that the data X_i comes from a mixture model with K components, then the complete data $\{X, Z\}$ is made up of the incomplete (or observed) data X and the latent (or missing) variable $Z \in \{1, 2, ..., k\}$.

The mixture model is defined as

$$p(X, Z) = \sum_{k=1}^{k} p(Z)p(X|Z)$$

The distribution of Z, $p(Z_i = k)$ (i.e. the probability associated with k^{th} component,) is called mixture weight, where $p(Z_i = k) = \pi_k$ and $\sum_k \pi_k = 1$.

On the other hand, the conditional distribution of X, $p(X_i|Z_i = k)$ (i.e the probability of x assuming that it comes from the k^{th} component,) is called mixture component.

If the mixture component follows normal distribution, then it is called the Gaussian mixture model, and it has the form of

$$p(X, Z) = \sum_{k=1}^{k} \pi_k \ \mathcal{N}(x_i|\mu_k, \sigma_k)$$

Together, we also say that

$$X_i|Z_i = k \sim \mathcal{N}(\mu_k, \sigma_k) \ with \ p(Z_i = k) = \pi_k$$

The complete likelihood of Gaussian mixture model is

$$p(X, Z|\theta = \mu, \sigma, \pi) = \prod_{i=1}^{n} \prod_{k=1}^{k} \pi_k^{I(Z_i=k)} \ \mathcal{N}(x_i|\mu_k, \sigma_k)^{I(Z_i=k)}$$

The complete log-likelihood of Gaussian mixture model is then

$$\log p(X, Z|\theta = \mu, \sigma, \pi) = \sum_{i=1}^{n} \sum_{k=1}^{k} I(Z_i = k) \log \pi_k + I(Z_i = k) \log \mathcal{N}(x_i|\mu_k, \sigma_k)$$

We cannot evaluate the complete log-likelihood because of the latent (or missing) variable Z. However, we can make use of that the posterior distribution of Z, $p(Z|X)$ contains the information we need about Z.

EM algorithm for Gaussian mixture

During the initialization, we set the initial estimates. This can be done in various ways, but a common practice is to use K-means clustering.

In the E-step, we first calculate the posterior probability $p(Z|X, \theta^0)$ using data X and current estimates of parameters θ^0. Then, we take expectation of the complete log-likelihood with respect to (w.r.t) this posterior probability. This expectation is expressed in literature as:

$$Q(\theta, \theta^0) = E_{Z|X,\theta^0} \log(p(X, Z|\theta)) = \sum_Z p(Z|X, \theta^0) \log(p(X, Z|\theta))$$

In the M-step, we re-estimate the next (or current + 1) parameters by maximizing Q

$$\hat{\theta} = \theta^{0+1} = argmax_\theta \; Q(\theta, \theta^0)$$

A Rough Sketch of the Implementation

Recall that the algorithm consists of three steps:

1. Initialization
2. The E-step
3. The M-step

```
for (i in 1:iterations){
   if (i = 1){
    # initialization
    # E-step:
```

1. pass data & initial estimates from `k-means` to `e_step` and store as `e_out`

   return posterior probability & log-likelihood

```
    # M-step:
```

2. pass the posterior probablity from `e_out` to `m_step` and store as `m_out`

   return estimates (e.g. μ, σ, π)

```
    # current log-likelihood
```

3. store the `log-likelihood` from `e_out` to `current log-likelihood`

```
   } else {
    # E-step
```

1. pass data & current estimates obtained from `m_out` to `e_step` and store as `e_out`

```
    # M-step
```

2. pass the posterior probability from `e_out` to `m_step` and store as `m_out`

```
    # check
```

3. calculate the difference between `current log-likelihood` & `current + 1 log-likelihood`

```
    if (convergence (i.e. change is minimal) ){
     break
    } else {
     1. set current + 1 log-likelihood to current log-likelihood
     2. repeat E-step and M-step
    }
    }
}
```

Application

For illustration, we apply EM algorithm to a simulated dataset that follows a two-component Gaussian mixture distribution.

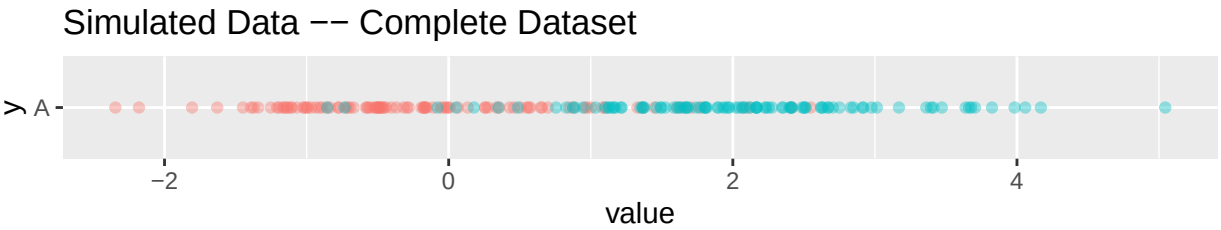
Simulated data

We use the `rnorm` function from the `stats` package to simulate 100 data points that follow $\mathcal{N}(0,1)$ and 100 data points that follow $\mathcal{N}(2,1)$.

The first 3 values of each distribution are:

component	value
A	-1.207
A	0.2774
A	1.084
B	2.415
B	1.525
B	2.066

Here is a graph:



We obtain the estimates of parameters using MLE

$$\mu_k = \frac{\sum x_{i,k}}{N_k}, \sigma_k^2 = \frac{\sum (x_{i,k} - \mu_k)^2}{N_k}$$

The estimates of parameters (i.e. μ, σ) are:

component	mean	sd
A	-0.1568	1.004
B	2.041	1.032

Now suppose that we do not know which distribution each data point is from (i.e. the latent (or missing) variable Z_i is involved.) This is when the EM algorithm can be of use.

Here is a graph:

Initialization using K-means clustering

Recall from class and lab that we discuss several ways such as plotting or finding the median or interquartile range (IQR) to set up initial values. Here, we use the `kmeans` function from the `stats` package to obtain initial values for the first EM initial iteration, as it is a common practice to use k-means clustering to find such values. For reference, k-means clustering is considered a method of hard labelling, as the data point is labelled as either $cluster_1$ or $cluster_2$ using the Euclidean distance.

The output of the first 3 values of each distribution are:

value	cluster
-1.207	2
0.2774	2
1.084	1
2.415	1
1.525	1
2.066	1

Here is a graph:

We obtain the estimates of parameters using MLE

$$\mu_k = \frac{\sum x_{i,k}}{N_k}, \sigma_k^2 = \frac{\sum (x_{i,k} - \mu_k)^2}{N_k}, \pi_k = \frac{N_k}{N}$$

The estimates of parameters (i.e. μ, σ, π) obtained from K-means are:

cluster	mean	sd	pi
1	2.099	0.8587	0.55
2	-0.4712	0.6671	0.45

E-step: Calculate posterior probability (or soft labelling) using Bayes Rule and pass it to M-step & store log likelihood to check for convergence

In the E-step of the first iteration, we calcuate the posterior probability of the latent variable $Z_i = k$ given the observations X_i (i.e. x_i belongs to the k^{th} cluster) using the estimates of parameters obtained from K-means. Specifically, given that X_i have an (incomplete) likelihood $p(X_i|Z_i = k) = \mathcal{N}(x_i|\mu_k, \sigma_k^2)$ and a prior belief that Z_i follows a probability distribution function $p(Z_i = k) = \pi_k$, the posterior probability of the latent variable $Z_i = k$ given the observations X_i (i.e. x_i belongs to the k^{th} cluster) is then given as follows

$$p(Z_i = k|X_i) = \frac{p(X_i|Z_i = k)p(Z_i = k)}{p(X_i)}$$

The output of the first 3 values of each distribution are:

value	post_1	post_2
-1.207	0.03252	0.9675
0.2774	0.5783	0.4217
1.084	0.9849	0.01509
2.415	1	1.54e-06
1.525	0.9989	0.001071
2.066	1	2.418e-05

In the E-step, we also compute and store the log likelihood to check for convergence.

M-step: Replace hard labelling with posterior probability (or soft labelling) and optimize the parameters using MLE & return the final estimates if convergence happens

In the M-step, we re-estimate the parameters replacing hard labelling N_k with posterior probability (or soft labelling) $p(Z_i = k|X_i)$.

Recall that in k-means, we have that

$$\mu_k = \frac{\sum_{i=1}^{N_k} x_{i,k}}{N_k}, \sigma_k^2 = \frac{\sum_{i=1}^{N_k}(x_{i,k} - \mu_k)^2}{N_k}, \pi_k = \frac{N_k}{N}$$

After replacing, we have that

$$\mu_k = \frac{\sum_{i=1}^{N} p(Z_i = k|X_i)x_i}{\sum_{i=1}^{N} p(Z_i = k|X_i)}, \sigma_k^2 = \frac{\sum_{i=1}^{N} p(Z_i = k|X_i)(x_i - \mu_k)^2}{\sum_{i=1}^{N} p(Z_i = k|X_i)}, \pi_k = \frac{\sum_{i=1}^{N} p(Z_i = k|X_i)}{N}$$

In the M-step, if convergence happens, we also return the final estimates that maximize the likelihood.

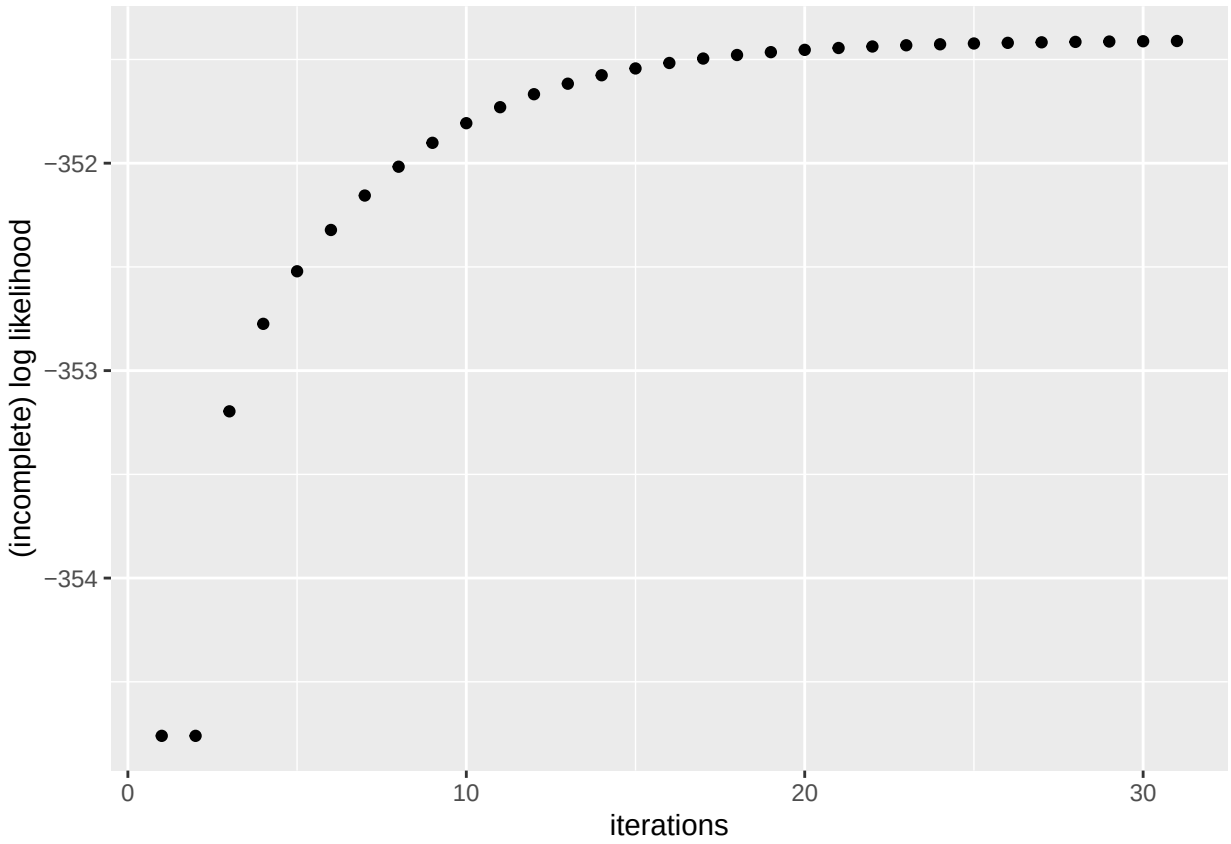
The final estimates of parameters (i.e. μ, σ, π) are:

cluster	mean	sd	pi
1	1.786	1.078	0.6553
2	-0.6611	0.591	0.3447

Convergence: Iterate between the E-step and the M-step untill convergence

Recall from calculus that given a function $f(x)$ that is continuous on an interval $I \in (c - \epsilon, c + \epsilon)$ for $\epsilon > 0$, optimization occurs at $x = c$ where $f'(x) \approx 0$ and we check $f''(x)$ to see if c is a (local) maximum or a minimum. Because of the latent (or missing) variable Z_i, we are unable to solve for the estimates. Instead, we check and update estimates by iterating between the E-step and the M-step untill convergence (i.e. change is minimal). Specifically, we compute and store the log likelihood at each EM iteration and compare this log likelihood to the log likehood of the previous iteration to see if the change is minimal. If the change is minimal (i.e. convergence), we stop and the estimates that the M-step returns are the final estimates. If it isn't, then we repeat another EM step.

Convergence occurs at 31st iteration with log likelihood = -351.4:



max_log_likelihood	#s of iterations
-351.4	31

Discussion

We see that the final estimates are not quite what we look for. This could be because of the initial values using K-means clustering, but the major reason is perhaps due to the simulated data. There is a lot of overlap between the 100 simulated data points that follow $\mathcal{N}(0,1)$ and another 100 data points that follow $\mathcal{N}(2,1)$. We expect EM algorithm to perform better if the overlap is not as wide as in our example, and it does (We re-simulate 100 data points that follow $\mathcal{N}(0,1)$ and another 100 data points that follow $\mathcal{N}(4,1)$ and find that the estimates are more accurate and convergence happens earlier now at 12th iteration.)

We also see how EM algorithm can be applied to two-component Gaussian mixture model. A logical next step is to expand the algorithm for k-component Gaussian mixture model and for other mixture model such as multinomial.

Reference

Fitting a Mixture Model Using the Expectation-Maximization Algorithm in R

Introduction to EM: Gaussian Mixture Models

Terminology

EM algorithm and GMM model

Maximum likelihood estimation

Mixture model

Posterior probability