EM Algorithm for Gaussian Mixture Models

Haocheng Hua
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
hh7@illinois.edu

May 10, 2019

1 Introduction of EM Algorithm

Expectation Maximization Algorithm is a numerical method to approximate maximum likelihood estimates when there are not only observed data and unknown parameters, but also hidden random variables, which are typically composed of a sequence of events. Thus, it is often hard to enumerate all of them to get the exact likelihood equation. It has been sucessfully applied in many fields such as Data Mining, Information Retrieval, Bioinformatics. The idea can be dated back to a paper by Dempster[1].

Throughout many applications, Gaussian Mixture Models is one of the most famous problems to which EM Algorithm has been successfully applied. In this report, I will dive deeply into it, derive it, study its implementation, convergence property with relevant speedup method and show the experiment result.

Before going into the details of Gaussian Mixture Models, Let's first take a look at the general idea of EM Algorithm. The EM Algorithm is composed of the following ingredients:

Θ: A set of unknown parameters needed to be estimated.
$Y = (X, Z)$: The complete data set, where X is the observed data set and Z is often called the hidden or latent data set which is not observed but it will be really helpful for us to estimate the unknown parameters if we know it.
$lnP(X, Z|\Theta)$: The log-likelihood of the complete data, which is a function of Θ is we know X and Z.

The intuition behind this algorithm is that if we know both X and Z, i,e, the complete dataset, we can get the maximum likelihood estimator by finding Θ that maximize the log-likelihood. But in many application, we can only get the observed data X and have no idea about what Z is, sometimes Z is composed of a set of sequence with many possible values and the number of the possible result is increasing exponentially, thus, it is always difficult to enumerate all of them. Thus, this conventional maximum likelihood approach is not feasible any more and the idea is to esimate the complete data log-likelihood by the conditional expectation of it given $X = x$ and current estimation of $\Theta^{(t)}$, which is often refered to as E step. And then we choose a new Θ to maximize this conditional mean, which is M step, expressing them in a nice mathematical form, we have the following:

E Step:

$$Q(\Theta|\Theta^{(t)}) = E\{lnP((X,Z)|\Theta)|X=x,\Theta^{(t)}\} \tag{1}$$

M Step:

$$\Theta^{(t+1)} = \arg sup_{\Theta} Q(\Theta|\Theta^{(t)}) \tag{2}$$

To implement EM Algorithm, we have to first set up the initial parameters in Θ, denoted as $\Theta^{(0)}$, and then repeat EM step till convergence.

The convergence properties of EM Algorithm have been studied by Dempster [1] and by Wu [2] and [3] has a well conclusion of it, which basically states the following:(it will be verifed in Gaussian Mixture Models)

Suppose that the complete data pdf $P(X,Z|\Theta)$ meets the assumption below.
- *It can be factored as* $P(X,Z|\Theta) = P(X|\Theta)K(Z|X,\Theta)$
- $lnP(X|\Theta)$ *is differentiable in* Θ
- $E\{lnK(Z|X,\hat{\Theta})|X=x,\hat{\Theta}\}$ *is finite for all* $\hat{\Theta}$
- $D(K(\cdot|X,\Theta)||K(\cdot|X,\Theta^{'}))$ *is differentiable with respect to* $\Theta^{'}$ *for fixed* Θ.
- $D(K(\cdot|X,\Theta)||K(\cdot|X,\Theta^{'}))$ *is continuous in* Θ *for fixed* $\Theta^{'}$

Then the incomplete data likelihood $P(X|\Theta^{(t)})$ is non-decreasing and will converge to a stationary point Θ^* which satisfies:

$$\frac{\partial P(X|\Theta)}{\partial \Theta}|_{\Theta=\Theta^*} = \mathbf{0} \tag{3}$$

2 Application in Gaussian Mixture Models

In Gaussian Mixture Models, we have K clusters in total, each cluster is a Gaussian distribution, which is characterized as:

$$P(x|z=i,\Theta) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)} \tag{4}$$

The unknown parameter Θ is a set of parameters:

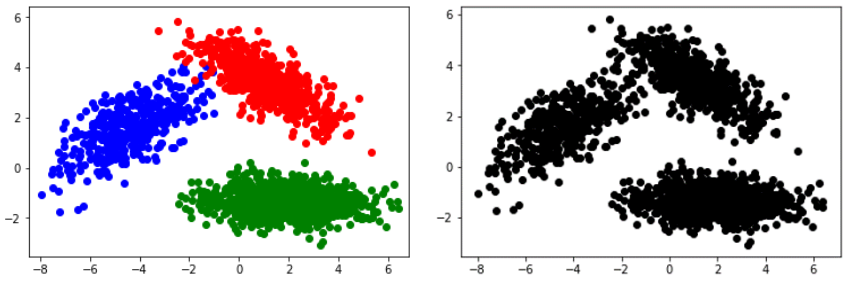
$$\Theta = \{\alpha_1,...,\alpha_K,\mu_1,...,\mu_K,\Sigma_1,...,\Sigma_K\}$$

where $\{\alpha_1,...,\alpha_K\}$ is the pick up probability for each cluster, and they meet the constraint: $\sum_{i=1}^{K}\alpha_i = 1$. and the second list and the third list is the mean vector and covariance matrix for each cluster Gaussian distribution respectively.

The observed data pdf given Θ can be formulated as:

$$P(x|\Theta) = \sum_{i=1}^{K}\alpha_i P(x|z=i,\Theta) \tag{5}$$

Each time when we samples the distribution, we will first pick up one of the distribution with the corresponding pick up probability, and then use that distribution to generate the data. Note that here the generated data $X = \{x_j\}_{j=1}^{N}$ is the observed data, where each $x_j \in R^d$ and N is the number of samples we get. z is the latent data which we do not observe. Our goal is to estimate



(a) Data generated by 3 distribution (b) Data that we see

Figure 1: Figures used to explain the goal of GM Models

Θ based on observed data X using the EM algorithm. A 2D data set with K=3 is shown above to specify our goal.

Now, since we have a clear understanding of what the problem is, we should focus on the exact step of how we implement EM algortihm on this problem. Below, I will derive the EM algorithm for Gaussian Mixture Models from scratches.

We should first get the equation in E step. Recall that:

$$Q(\Theta|\Theta^{(t)}) = E\{lnP(X, Z|\Theta)|X = \{x_j\}_{j=1}^N, \Theta^{(t)}\} \tag{6}$$

Since each data sample x_j is generated independently, we have:

$$lnP(X, Z|\Theta) = \sum_{i=1}^{N} lnP(x_i|z_i, \Theta)P(z_i|\Theta) \tag{7}$$

Combine the above two terms, we have:

$$Q(\Theta|\Theta^{(t)}) = \sum_{i=1}^{N} E\{lnP(x_i|z_i, \Theta)P(z_i|\Theta)|X = \{x_j\}_{j=1}^N, \Theta^{(t)}\} \tag{8}$$

By the definition of the conditional expectation, we can get:

$$Q(\Theta|\Theta^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} lnP(z_i = k|\Theta)P(x_i|z_i = k, \Theta)P(z_i = k|X = \{x_j\}_{j=1}^N, \Theta^{(t)}) \tag{9}$$

For simplicity, let's denote $P(z_i = k|X = \{x_j\}_{j=1}^N, \Theta^{(t)})$ as w_{ik}, which is the probability of data sample x_i belonging to cluster k given $\Theta^{(t)}$ and x_i and is often refered to as posterior membership probability. Notice that $P(z_i = k|\Theta)$ is nothing but α_k. and by plugging equation (4) into equation (9), we get:

$$\sum_{i=1}^{N}\sum_{k=1}^{K}(ln\alpha_k)w_{ik} + \sum_{i=1}^{N}\sum_{k=1}^{K}\{-\frac{1}{2}(x_i - \mu_k)^T\Sigma_k^{-1}(x_i - \mu_k) - \frac{1}{2}ln|\Sigma_k| - \frac{d}{2}ln(2\pi)\}w_{ik} \tag{10}$$

$$s.t. \sum_{k=1}^{K}\alpha_k = 1$$

Now before we finish the E step, we need to come up with the formula to compute w_{ik}, which is quite intuitive. But I will also derive it based on simple probability knowledge. It is easy to see that it can be computed efficiently based on the previous estimate of Θ and given data samples.

$$\begin{aligned} w_{ik} &= P(z_i = k | x = \{x_j\}_{j=1}^N, \Theta^{(t)}) \\ &\overset{(a)}{=} P(z_i = k | x = x_i, \Theta^{(t)}) \\ &\overset{(b)}{=} \frac{P(z_i = k|\Theta^{(t)}) P(x_i | z_i = k, \Theta^{(t)})}{P(x = x_i | \Theta^{(t)})} \\ &\overset{(c)}{=} \frac{\alpha_k^{(t)} P(x_i | z_i = k, \Theta^{(t)})}{\sum_{m=1}^K \alpha_m^{(t)} P(x_i | z_i = m, \Theta^{(t)})} \end{aligned} \tag{11}$$

(a) is due to the fact that the data samples are generated from identical independent distribution, knowing x_i is enough.

(b) is simply the definition of conditional probability.

(c) is obtained by the total probability formula.

Now we are ready to go through the M step. Let's first take the derivative with respect to the pick up probability α_k, we can divide the equation into several terms to simplify analysis, for any $1 \leq k \leq K$, we have:

$$Q(\Theta|\Theta^{(t)}) = ln(\alpha_k) \sum_{i=1}^N w_{ik} + ln(1 - \alpha_k - \sum_{i \neq c, i \neq k} \alpha_i) \sum_{i=1}^N w_{ic} + \text{term irrelevant to } \alpha_k \tag{12}$$

Take the derivative with respect to α_k and set to zero, we get(well, to be more precise, we also have to show that the second derivative of the term is smaller than 0. we can show that, but considering there is limited space for this report, I cut that part)

$$\begin{aligned} \frac{1}{\alpha_k} \sum_{i=1}^N w_{ik} &= \frac{1}{1 - \alpha_k - \sum_{i \neq c, i \neq k} \alpha_i} \sum_{i=1}^N w_{ic} \\ &= \frac{1}{\alpha_c} \sum_{i=1}^N w_{ic} \end{aligned} \tag{13}$$

Subject to the constraint $\sum_{k=1}^K \alpha_k = 1$, we can only set

$$\alpha_k^{(t+1)} = \frac{\sum_{i=1}^N w_{ik}}{N} \triangleq \frac{N_k}{N} \tag{14}$$

Take the derivative with respect to μ_k and set it to 0, we have

$$\Sigma_k^{-1} (\sum_{i=1}^N w_{ik}(x_i - \mu_k)) = \mathbf{0} \tag{15}$$

Since Σ_k^{-1} is invertible, the above equation implies that:

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^N w_{ik} x_i}{N_k} \tag{16}$$

Take the derivative with respect to Σ_k and set it to 0, here, we have to use some deeper matrix analysis formula:

$$\frac{\partial x^T \Sigma_k^{-1} x}{\partial \Sigma_k} = -\Sigma_k^{-1}(x_i - \mu_k)(x_i - \mu_k)^T \Sigma_k^{-1} \tag{17}$$

and

$$\frac{ln|\Sigma_k|}{\partial \Sigma_k} = \Sigma_k^{-1} \tag{18}$$

plugging them into the derivative

$$\sum_{i=1}^{N} w_{ik}(\Sigma_k^{-1}(x_i - \mu_k)(x_i - \mu_k)^T \Sigma_k^{-1} - I\Sigma_k^{-1}) = 0 \tag{19}$$

which is just

$$\Sigma_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^{N} w_{ik}(x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T \tag{20}$$

We can also calculate the observed data log-likelihood to track the convergence of the EM algorithm, which can be computed as:

$$lnP(X|\Theta^{(t+1)}) = \sum_{i=1}^{N} lnP(x_i|\Theta^{(t+1)}) = \sum_{i=1}^{N} ln(\sum_{k=1}^{K} \alpha_k^{(t+1)} P(x_i|z_i = k, \Theta^{(t+1)})) \tag{21}$$

3 EM Algorithm Experiment Results

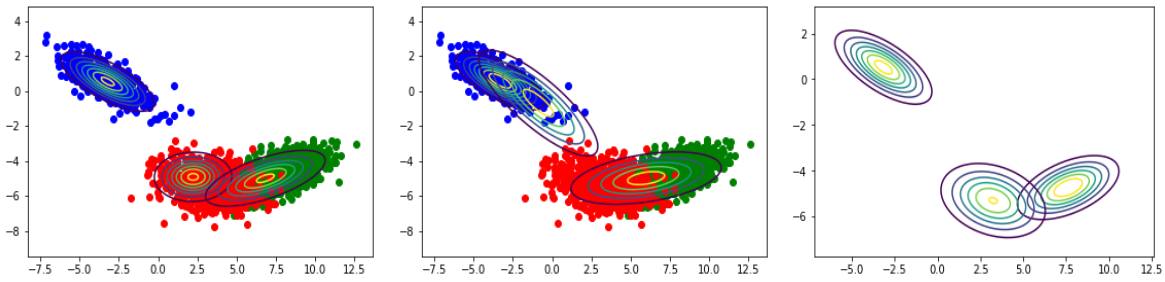
Serious Overlapping among the mixture components, bad parameter initilization, and unbalanced pick up probability are well known to make the final result trapped in local maximum more likely and slow down the convergence rate[5][6]. Here we implement the algorithm derived above in Python and show these effects concretely. We will also verifed the theorem of convergence property shown in section 1. The result shown below is obtained by using $K = 3$, $N = 2000$ and $d = 2$.

To elimiate the effect of unbalanced pick up probability so that we can take a close scrutiny at the other two effects, we first set the original pick up probability to $\{0.5, 0.2, 0.3\}$ and randomly generate the mean vector and the covariance matrices for all K clusters.

Figure 2 shows that with different initialized parameters, the algorithm might converge to a local maximum instead of a global maximum. Due to the serious overlap between red cluster and green cluster, it is hard for the algorithm to distiguish them from each other. Thus, even the parameters estimation obtained from the global maximum are not very accurate.

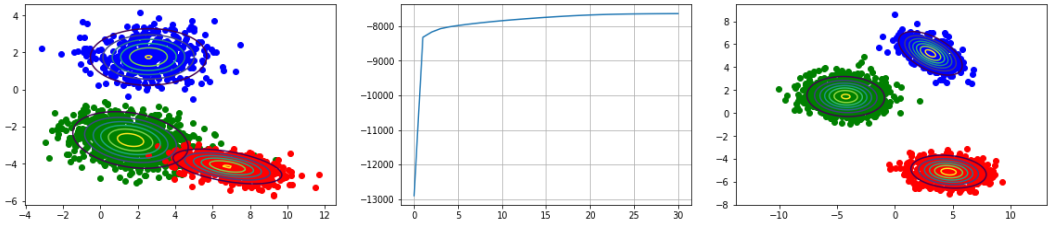
Keep setting the original pick up probability to $\{0.5, 0.2, 0.3\}$ with different level of overlapping dataset and distribution, we get the following result shown in Figure 3.

If we compare the result shown in Figure 3, we can see that overlapping indeed slow down the convergence rate. In the first setup, the algorithm takes almost 30 iteration to converge to the global maximum while in the second setup, it takes less than 20 iteration and if we change the initialization for the second setup, the algorithm converge to the global maximum in less than 5 iteration!

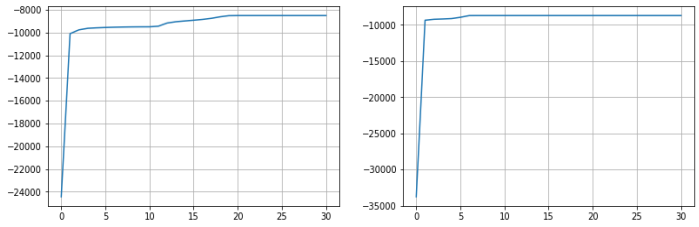


(a) Converge to global maximum (b) Trapped in local Maximum (c) Original Distribution

Figure 2: converged result with different initialized parameters(in 15 iteration)



(a) Overlapped distribution (b) Learning curves for (a) (c) non-overlapped distribution



(d) Learning curves for (c) with bad initialization (e) Learning curves for (c) with good initialization

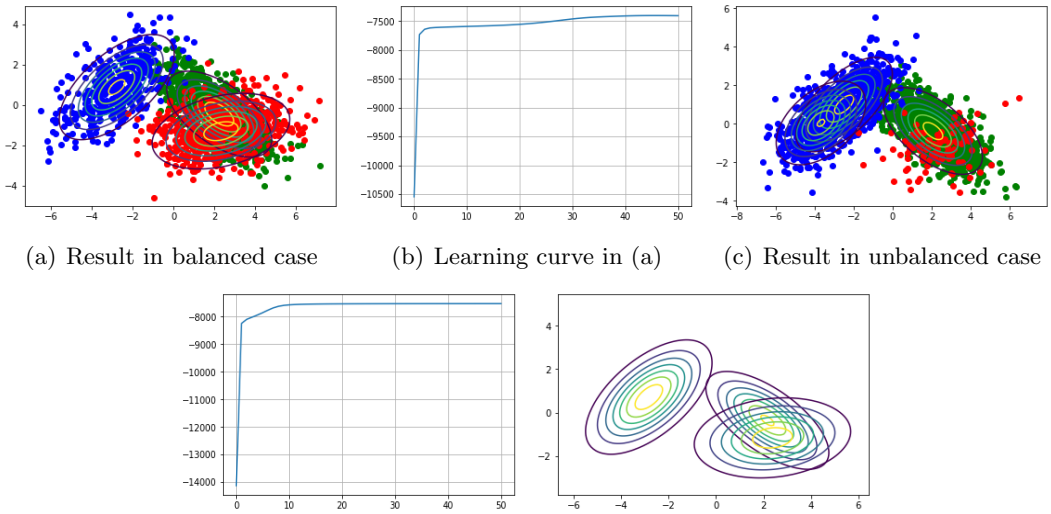
Figure 3: Comparison between overlapping and non-overlapping cases

Finally, let's examine the impact of unbalanced pick up probability on the convergence rate and convergence result.

We fix the original parameter and the initial $\Theta^{(0)}$ except the pick up probability vector, $\{0.5, 0.3, 0.2\}$ for balanced case and $\{0.5, 0.45, 0.05\}$ for unbalanced case. The result is shown in Figure 4. We can see that for the balanced case, the algorithm figure out the correct solution while for the unbalanced case, it just cannot pick up the distribution with small pick up probability.

Besides the fact that the observed data log-likelihood is always non-decreasing, several key conclusion based on the above analyse can be given below:

- *Initialization has great impact on both the convergence rate and the convergence result.*
- *Overlapping will slow down the convergence.*
- *The pick up probability vector also has great impact on the convergence result.*



(a) Result in balanced case (b) Learning curve in (a) (c) Result in unbalanced case

(d) Learning curve in (c) (e) Original distribution

Figure 4: converged result for dataset with different pick up probability

4 Convergence Improvement

The issues has been shown previously about EM algorithm in Gaussian Mixture Model. Over the past decades, researchers have proposed several improved methods to speed up convergence and make the algorithm coverge more likely to global maximum. [4] and [6] have proposed two methods which are called Deterministic Annealing EM Algorithm(DAEM) and Deterministic Anti-Annealing EM Algorithm(DAAEM) respectively. The basic idea of them is to modify the rule to calculate the posterior membership probability w_{ik} using the scheduling parameter β.

$$w_{ik} = \frac{(\alpha_k^{(t)} P(x_i|z_i = k, \Theta^{(t)}))^\beta}{\sum_{m=1}^{K} (\alpha_m^{(t)} P(x_i|z_i = m, \Theta^{(t)}))^\beta} \tag{22}$$

They first start with $\beta \approx 0$, to maintain more robust global convergence, and after each iteration, they gradully change β from ≈ 0 to 1(the normal case) and from ≈ 0 to $\beta_{max} (> 1)$ then back to 1 in DAEM and DAAEM, respectively.

Some intuition of these algorithm are as follows. When $\beta = 0$, the posterior membership probability becomes $\frac{1}{K}$, i,e, the membership for each data sample becomes uniformly distributed among all the Gaussian components. And the updated equation for the mean vector will just be simply the mean of all the data samples, i,e,

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{23}$$

This kind of update just eliminates the effects of those wired or bad initilization and prevent possible converging to the local maximum. The pick up probability is also uniformly distributed, thus, it is quite fair for each cluster. However, if we keep $\beta \approx 0$, the algorithm will not be able to distinguish these clusters from each other. So we should gradully incease β to 1 monotonically.

Unlike DAEM, DAAEM tends to increase β to some value larger than 1 and then go back to 1. Since when β becomes larger, the cluster becomes more separated from each other, which acclerates the convergence rate. Actually, when $\beta \to \infty$, it becomes a winner-takes-all strategy and is close to K-Mean algorithm in terms of the fact that each data sample can only be classified into one cluster, rather than a soft version provided by conventional EM algorithm.

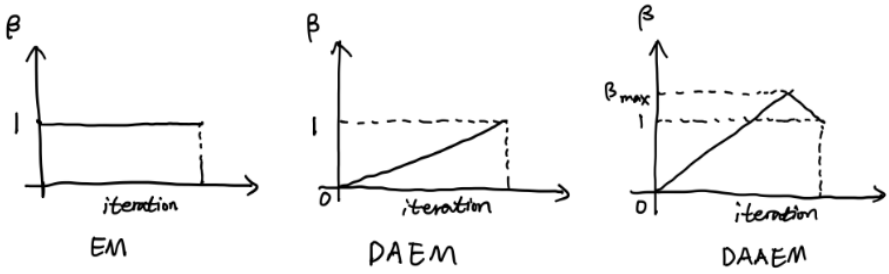
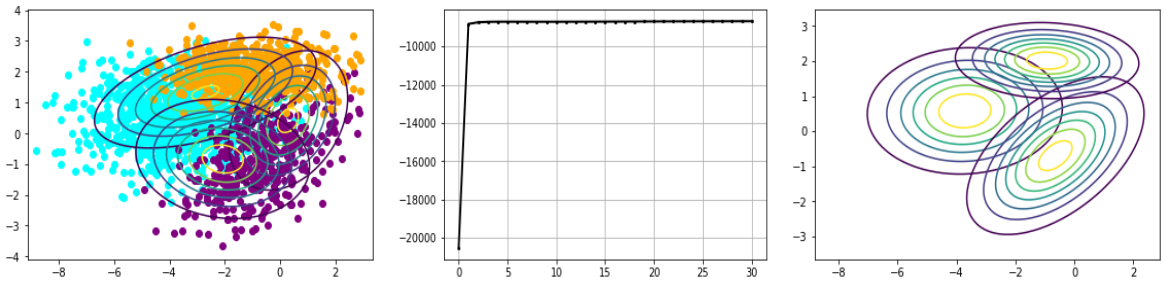


Figure 5: Comparison among three algorithms

The experiment result is shown as follows. In Figure 6, we show another overlapping distirbution and relevant EM algorithm converged result and log-likelihood curves. We can see that the converged result is incorrect and the algorithm fails to converge to the global maximum.

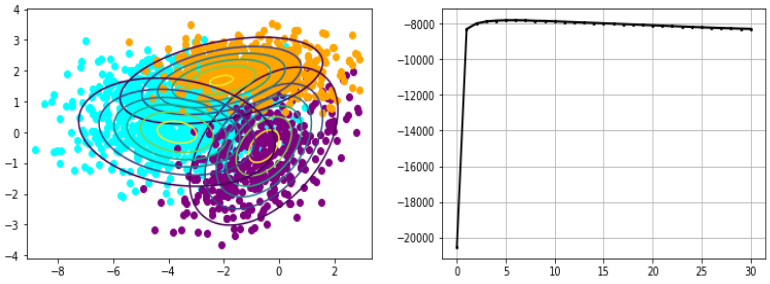


(a) EM converged result (b) Learning curves (c) Original Distribution

Figure 6: Experiment result using EM Algorithm

We then use DAEM algorithm, after tuning the parameters a little bit, we let the algorithm start with $\beta = 0.5$, then gradually incerases to 1 with a step size of 0.075, once it reaches 1, it will remains to be 1 ever since. The result is shown in Figure 7. we can see that it finally converges to the global maximum. Notice that the log-likelihood first increases and then decreases a little bit, the main reason is that when we calculate the membership probability, we rescale it with β, which is a little bit different from what we have derived and cause some discrepancy. Indeed, DAEM algorithm will provide more robust global maxima convergence in some scenario.

We now switch to DAAEM algorithm, we first begin with $\beta = 0.5$, then we gradually increase it with a step size 0.075, after it reaches the maximum possible value $\beta_{max} = 1.3$, then we gradually decrease it with a step size 0.075 until it reaches 1 again and it will remain to be 1 ever since. The result is shown in Figure 8. We can see that DAAEM algorithm converges more quickly than

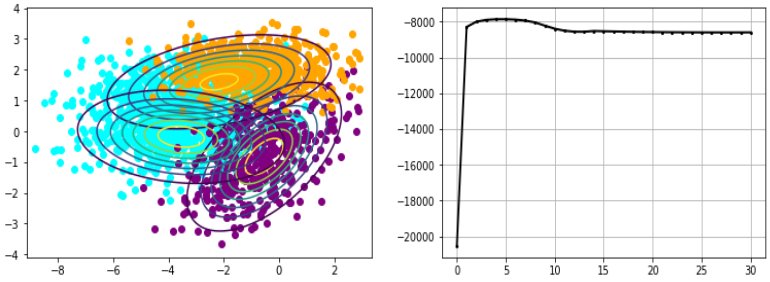


(a) DAEM converged result (b) Learning curves

Figure 7: Experiment result using DAEM Algorithm

DAEM algorithm since its log-likelihood remains unchanged after 10 iteration and also converges to the global maximum. The estimated pick up probability vector is also close to the real one(although DAEM outperforms DAAEM in terms of pick up probability accuracy) The result is shown below.

	real vector	estimated vector
DAAEM	$\{0.56, 0.18, 0.26\}$	$\{0.5, 0.2, 0.3\}$
DAEM	$\{0.49, 0.2, 0.31\}$	$\{0.5, 0.2, 0.3\}$
EM	$\{0.67, 0.1, 0.23\}$	$\{0.5, 0.2, 0.3\}$



(a) DAAEM converged result (b) Learning curves

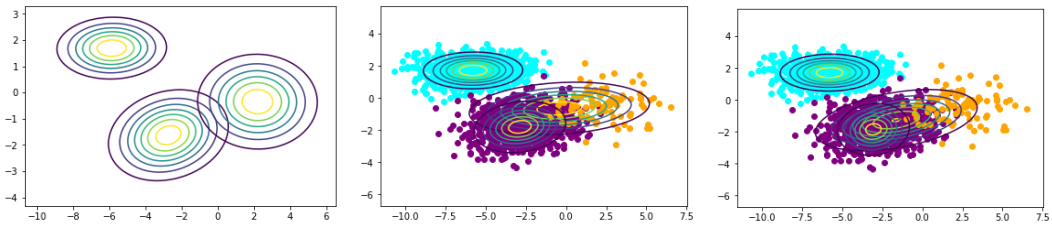
Figure 8: Experiment result using DAAEM Algorithm

As we see, these algorithms indeed speed up the convergence and makec converging to global maxima more likely when there are highly overlapped components. We finally examine these algorithms when there exists unbalanced pick up probability.

After 50 iteration, the final pick up vector result is shown below, it is easy to see that DAAEM gets more accurate results in terms of pick up probability.

	real vector	estimated vector
DAAEM	$\{0.50, 0.39, 0.11\}$	$\{0.5, 0.45, 0.05\}$
EM	$\{0.50, 0.34, 0.16\}$	$\{0.5, 0.45, 0.05\}$

The result for DAAEM remains nearly unchanged after 15 iteration while EM needs 20 iteration. The final result is close to each other but DAAEM still outperforms EM a little bit.



(a) Original Distribution (b) Result after 15 iteraion using DAAEM (c) Result after 20 iteraion using EM

Figure 9: Comparison between EM and DAAEM with unbalanced pick up probability

5 Conclusion and Further Work

In this report, the concept of EM algorithm and its application in Gaussian Mixture Models has been introduced. Then both E step and M step for it have been derived and implemented in Python. Experiment result has been shown and several issues have been pointed out including overlapping mixtures, bad initialization and unbalanced pick up probability. All of them have great impacts on the convergence rate and convergence result of EM algorithm. Then Two methods called DAEM and DAAEM has been addressed and also been implemented. The result shows that DAEM can make global maxima convergence more likely in some scenarios while DAEM not only maintains the desired property of DAEM, but also accelerates the convergence when there are small clusters, i,e, unbalanced pick up probability. It is also found that although these two algorithms give us better performance, it takes some time to tune the parameters of them. Further work includes EM online algorithm exploration, in which we start the algorithm once we get data samples rather than wait until we obtain all the available data.

The author would like to thank Pro.Bruce Hajek for his inspiring suggestions about the project.

References

[1] Dempster A P, Laird N M, Rubin D B. Maximum likelihood from incomplete data via the EM algorithm[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1977, 39(1): 1-22.

[2] Wu C F J. On the convergence properties of the EM algorithm[J]. The Annals of statistics, 1983, 11(1): 95-103.

[3] B. Hajek, Random Processes for Engineers, Cambridge University Press, 2015

[4] Naim I, Gildea D. Convergence of the em algorithm for gaussian mixtures with unbalanced mixing coefficients[J]. arXiv preprint arXiv:1206.6427, 2012.

[5] Xu L, Jordan M I. On convergence properties of the EM algorithm for Gaussian mixtures[J]. Neural computation, 1996, 8(1): 129-151.

[6] Ueda N, Nakano R. Deterministic annealing EM algorithm[J]. Neural networks, 1998, 11(2): 271-282.