Particle Swarm optimization of EM-algorithm for Gaussian Mixture Model

Sergei Kholkin [1] **Maxim Kurkin** [1] **Anastasia Batsheva** [1] **Anna Akhmatova** [1]

Abstract

GMM is optimized using EM algorithm. There were many attempts to beat EM algorithm (Hosseini & Sra, 2015) it is still mainly SOTA. Through EM algorithm find local minima and highly depends on initial data. The idea of the project is to optimize the initialization of EM algorithm using Particle Sworn Optimization (PSO) as it was already done with similar algorithm k-means (Li & Wang, 2022).

1. GMM parameters estimation

Provide statement for GMM: We consider a family of mixture of K multivariate Gaussian distributions in $\mathbb{R}^d$ that are parametrized as follows: $\theta = \{w_1, w_2, \dots, w_K, \mu_1, \Sigma_1, \dots, \mu_K, \Sigma_K\}$. Where $\{\mu_i, \Sigma_i\}$ is parametrization of ith multivariate Gaussian distribution, $\mu_k \in \mathbb{R}^d$ is vector of means and covariance matrix $\Sigma_k \in \mathbb{S}^d_{++}$ where $\mathbb{S}^d_{++}$ is a set of symmetric positive definite $d \times d$ matrices and $w_k \in [0, 1]$ is a weight of each gaussian distribution and $\sum_{k=1}^K w_k = 1$. Given a set of data $X = \{x_1, \dots, x_n\}, x_i \in \mathbb{R}^d$ which are i.i.d. Mixture probability desnity function is $p(x|\theta) = \sum_{k=1}^K w_k p_k(x|\mu_k, \Sigma_k)$. Our objective is to obtain maximum likelihood estimation of θ through maximizing log-likelihood function:

$$\log(X|\theta) = \sum_{i=1}^N \log\left(\sum_{k=1}^K w_k p_k(x_i|\mu_k, \Sigma_k)\right)$$

Since this functional is not convex we don't have covinient algorithm for obtaining global maxima.

2. Expectation Maximization

Conventional algorithm for maximizing log-likelihood is Expectation-Minimization(EM). We do add latent variables to the model $Z = \{z_1, \dots, z_n\}, z_i \in \mathbb{Z}[0, k]$, $z_i = k$ if x_i was generated from kth Gaussian component. Then we do add probability $z_{ik} = q(z_i = k) = p(z_i = k|x_i, \mu_k, \Sigma_k)$ that x_i was generated from kth Gaussian component and reformulate out model and log-likelihood function considering that varibales.

$$\log p(X|\theta) = \mathbb{E}_{q(z)}\left[\log \frac{p(X, Z|\theta)}{q(z)}\right] + KL(q||p)$$

Since $KL(q||p) \geq 0$, we get that

$$\log p(X|\theta) \geq \mathbb{E}_{q(z)}\left[\log \frac{p(X, Z|\theta)}{q(z)}\right];$$

$$\mathbb{E}_{q(z)}\left[\log \frac{p(X, Z|\theta)}{q(z)}\right] = \mathcal{L}(q, \theta)$$

is called Evidence Lower Bound (ELBO) and we do optimize it instead of $\log p(X|\theta)$. Then consider fixed point iteration:

For fixed $\theta^{(t)}$, whete t is the number of iteration:

$$q(Z)^{(t+1)} = \arg\max_q \mathcal{L}(q, \theta^{(t)}) = p(Z|X, \theta^{(t)})$$

$$z_{ik}^{(t+1)} = q(z_i = k)^{(t+1)} = \frac{w_k^{(t)} p_k(x_i, \theta_k^{(t)})}{\sum_{j=1}^K w_j^{(t)} p_j(x_i, \theta_j^{(t)})}$$

Then for fixed $q(Z)^{(t+1)}$

$$\theta^{(t+1)} = \arg\max_\theta \mathcal{L}(q^{(t+1)}, \theta) = \arg\max_\theta \mathbb{E}_{q(z)} \log p(X, Z|\theta)$$

That leads up to update of parameters:

$$w_k^{(t+1)} = \frac{1}{N}\sum_{i=1}^N z_{ik}^{(t+1)}$$

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

$$\Sigma_k^{(t+1)} = \frac{\sum_{i+1}^N z_{ik}(x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^\top}{\sum_{i+1}^N z_{ik}}$$

After a number of iteration EM converge and one the EM problems is that it strongly depend on initialization.

*Equal contribution [1]Skolkovo Institute of Science and Technology, Russia. Correspondence to: Sergei Kholkin <sergei.kholkin@skoltech.ru>, Anastasia Batsheva <anastasia.batsheva@skoltech.ru>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

3. Particle swarm optimization

Particle swarm optimization (PSO) is a population-based stochastic search algorithm that is inspired by the social interactions of swarm animals. In PSO each member of a polutaion is called particle. We have some function we want to optimize $f(X_\theta), X_\theta \in \mathbb{R}^d$. Each particle is represented as two vectors: $X_i = \{X_{(\theta,i)}, x_v\}, X_\theta, x_v \in \mathbb{R}^d$ parameters and velocity respectively. Parameters part is the candidate solution that particle does represent and velocity is the part used for defining optimization step. We can evaluate criterion value for each particle $f(X_{(\theta,i)})$. As the optimization prcodure we do iterative updates. At the each iteration we do remember the particle position that does have the best criterion among all the particle positions among all the iterations (global best) and is denoted as $X^{(GB,t)}$. Also through the search each particle remebers its personal best position $f(X_i^{(PB,t)}) \geq f(X_{\theta,i}), i \in 1, \ldots, i_{last}$

We start from initializing particles by some way, choosing $X^{(GB,1)}$ and $X_i^{(PB,1)}$ and then we do run optimization steps as follows:

$$
\begin{aligned}
X_{v,i}^{(t+1)} = \eta X_{v,i}^{(t)} + c_1 U_1^{(t)}(X_i^{(PB,t)} - X_{(\theta,i)}) + \\
+ c_2 U_2^{(t)}(X^{(GB,t)} - X_{(\theta,i)}), \\
X_{(\theta,i)}^{(t+1)} = X_{(\theta,i)}^{(t)} + X_{v,i}^{(t+1)}, \\
X_i^{(PB,t+1)} = X_i^{(PB,t^*)}, t* = \arg\max_t f(X_i^{(PB,t)}), \\
X^{(GB,t+1)} = X_{\theta,i}^{(GB,t^*)}, t* = \arg\max_t f(X_i^{(GB,t)})
\end{aligned}
$$

Where η is inertia coefficient, c_1 and c_2 are acceleration weights, $U_1, U_2 \sim U[0,1]$ random numbers and t is iteration number. We have some function we want to optimize.

4. PSO for GMM

We do want apply PSO for optimizing GMM. Every particle parameters part will be representing full GMM candidate solution $X_{(\theta,i)} = \{w_1, w_2, \ldots, w_K, \mu_1, \Sigma_1, \ldots, \mu_K, \Sigma_K\}_i$. But we would need to maintain positive definite constraint on covariance matrix $\Sigma \in \mathbb{S}_{++}^d$ and PSO update doesn't necessary do it. $\Sigma_{new} = \Sigma_1 + (\Sigma_2 - \Sigma_3)$ may not be positive definite. Then to perform optimization on the manifold of positive definite matrices lets consider some parametrization of covariance matrix $\Sigma = g(\vartheta)$ so that changing $g(\vartheta) \in \mathbb{S}_{++}^d, \forall \vartheta \in \mathbb{R}^m$. One possible parametrization is Cholesky decomposition $\Sigma = LL^\top$ but as a result we have $\frac{d(d+1)}{2}$ parameters which are unbounded, can have very diverse values $(-\infty, +\infty)$ and that doesn't allow us to properly optimze. Another parametrization through eigenvalues and givens rotation matrix angles was presented in (Çağlar Arı et al., 2012).

4.1. Eigenvalue and Givens rotation matrix angles parametrization

Theorem 1 *An arbitary covariance matrix with $\frac{d(d+1)}{2}$ degrees of freedom can be parametrized by using d eigenvalues in a particular order and $\frac{d(d-1)}{2}$ Givens rotation matrix angles $\phi^{pq} \in [-\pi/4, 3\pi/4]$ for $1 \leq p < q \leq d$ computed from the eigenvector matrix whose columns store the eigenvectors in the same order as the corresponding eigenvalues.*

Lemma 1 *An eigenvactor matrix $V \in \mathbb{R}^{d \times d}$ can be written as a product of $\frac{d(d-1)}{2}$ Givens rotation matrices with angles $\phi^{pq} \in [-\pi/4, 3\pi/4]$ and a diagonal matrix with ± 1 entries.*

For proof see (Çağlar Arı et al., 2012). $\Sigma \in \mathbb{S}_{++}^d$ is symmetrid hence it is diagonizable in orthogonal basis and we can represent it as $\Sigma = V\Lambda V^\top = \sum_{i=1}^d \lambda_i v_i v_i^\top$. Regarding Lemma 1 that we don't need to store the diagonal matrix ± 1 entries because $v_i v_i^\top = (-v_i)(-v_i)^\top$. Then we can construct our covariance matrix as $\Sigma = V\Lambda V^\top = \sum_{i=1}^d \lambda_i v_i v_i^\top = \sum_{i=1}^d \lambda_i \prod_{p=1}^d \prod_{q=p+1}^d G(p,q,\phi^{pq})$, where $G(p,q,\phi^{pq})$ is Givens rotaion matrix with for dimensions p and q with ϕ^{pq} rotation angle. Also notice since our eigendecomposition is unique up to eigenvalues/eigenvectors permutation so we need to maintain order of eigenvalues and corresponding Givens rotation angles after making the decomposition. To tackle this issue authors proposed algorithm of eigenvectors ordering w.r.t. reference eigenvectors matrix V_{ref} which happens to be the personal best for each particle.

You can see the algorithm in appendix 3Algorithm

But this parametrization still have a problem since it has $\frac{d(d+1)}{2}$ parameters, hence optimization in such a high dimensional space can be difficult and we need to compute SVD and QR factorication for each Gaussian covariance matrix for each particle for each iteration, which slows algorithm down.

4.2. Low rank delta parametrization

We propose novel method of parametrization, we paramatrize not the covariance matrix but it's inverse, precision matrix, which is also $\mathbb{S}_{++}^d$ and we parametrize not the matrix but the addition to it.

$$\Lambda_{new} = \Lambda + \Delta(\vartheta)$$

Using property that for $\Lambda_1, \Lambda_2 \in \mathbb{S}_{++}^d$ holds true $\Lambda_1 + \Lambda_2 \in \mathbb{S}_{++}^d$. We choose such Δ that $\Delta(\vartheta) \in \mathbb{S}_{++}^d, \forall \vartheta \in \mathbb{R}^m$. Than we get that that $\Lambda_1 + \Delta(\vartheta) \in \mathbb{S}_{++}^d$.

Let's try to avoid $\frac{d(d+1)}{2}$ parameters for precision matrix. We can do that by making low rank addition, in other words

making Δ such that $\text{rank}(\Delta(\vartheta)) < d$ and hence we can lower number of dimensions for ϑ so that $\vartheta \in \mathbb{R}^m, m < \frac{d(d+1)}{2}$.

We propose low rank delta parametrization satisfying all the above requirements as follows:

$$\Lambda_{new} = \Lambda + \text{diag}(a_1^2, \dots, a_d^2) + \sum_{i=1}^{R} b_i b_i^\top$$

Notice that $\text{diag}(a_1^2, \dots, a_d^2) \in \mathbb{S}^d_{++}, \forall a_i \in R$ and $b_i b_i^\top \in \mathbb{S}^d_{++}, \forall b_i \in \mathbb{R}^d$. $\text{rank}(\sum_{i=1}^{R} b_i b_i^\top) \leq R$ for approximating parameters of precision matrix and $\text{diag}(a_1^2, \dots, a_d^2)$ because we need special attention to diagonal elements. Total number of parameters is $(R+1)d$. For the initialization we run EM algorithm and get $\{w_1, w_2, \dots, w_K, \mu_1, \Lambda_1, \dots, \mu_K, \Lambda_K\}$ parameters, we freeze $\Lambda_{(i,base)}$, and set $w_1, w_2, \dots, w_K, \mu_1, \dots, \mu_K$ as particle initial parameters. Then we randomly initialize low rank addition $a_i \sim N(0, \sigma_1)$ and $b_i \sim N(0, \sigma_2 I_d)$ and compute precision matrices for each Gaussian for mth particle as $\Lambda_{(i,m)} = \Lambda_{(i,base)} + \text{diag}(a^2_{(1,m)}, \dots, a^2_{(d,m)}) + \sum_{i=1}^{R} b_{(i,m)} b^\top_{(i,m)}$.

Then PSO particle will contain parameters as follows: $\{w_1, w_2, \dots, w_K, \mu_1, \dots, \mu_K, a_1, \dots, a_d, b_1, \dots, b_r\}$

4.3. Low rank delta parametrization for PSO with EM reinit

EM is still very effictive algo to search for a local minima so let's try to utilize it inside our PSO framework. Unfortunately we are not able to extract our parametrization from precision matrix in a unique way so we will have an opportunity to init EM using constructed from PSO particles parameters weights, means and precision matrices, then run EM and but we won't have an opportunity to construct PSO particle back using resulting EM weights, means and precision matrices. So let's try to use LogLikelihood of PSO particles as a proxy metric to LogLikelihood of EM starting from PSO particles coordinates. Then we'd have criterion corresponding to each PSO particle parametrization itself, $\theta_{pso} = \{w_1, w_2, \dots, w_K, \mu_1, \dots, \mu_K, a_1, \dots, a_d, b_1, \dots, b_r\}$, from θ_{pso} we'll construct GMM parameters θ in a deterministic way $\theta_{pso} = \{w_1, w_2, \dots, w_K, \mu_1, \dots, \mu_K, a_1, \dots, a_d, b_1, \dots, b_r\} \rightarrow \{w_1, w_2, \dots, w_K, \mu_1, \Lambda_1, \dots, \mu_K, \Lambda_K\} = \theta$. $\log p(x|\theta)$ will be criterion corresponding to each PSO particle parametrization itself. And we are able to launch EM starting from θ and obtain new set of GMM parameters θ_{EM}. $EM(init = \theta) = \theta_{EM}$ and corresponding criterion will be $\log p(x|\theta_{EM})$. So we'll optimize using PSO w.r.t. first criterion but we'll evaluate second criterion and output as an algorithm result parameters θ_{EM}.

Also for economy of computations we can run EM reinitialization for particles each N iterations (for example 10).

Resulting algorithm will be as follows 1

5. Experiments

5.1. Experiments setup

We compare our algorith with vanilla EM either by execution time 1 or EM iterations budget (approach used in (Çağlar Arı et al., 2012)) 2 3 Comparison by EM iterations budget goes as follows. For PSO algorithm we have M particles which are being init based on one EM which runs T_{init} iterations with M random init samples, then each particle through running of PSO algorithm is being reinit T_1 times using EM with T_2 iteration budget. So in total number of EM iterations for PSO algorithm run is: $M(T_{init} + T_1 * T_2)$. Then lets use this EM iterations budget for EM with random init and run M random samples with $(T_{init} + T_1 * T_2)$ iterations budget for each EM run. Then for simmplicity we set $T_{init} = T_1 * T_2$. See the table 2

5.2. Synthetic Data

We'll use model of generatin synthtic data from (Çağlar Arı et al., 2012). Data sets will be generated from some Gaussian Mixture Model itself with various dimenstions $d \in \{30, 50, 70\}$, number of components will be $M \in \{15\}$, with sample size $N \in \{1000\}$. Also we add c separation as in (Dasgupta, 1999). Two Gaussians can be called c-separated if: $\|\mu_2 - \mu_1\|_2 \geq c\sqrt{d \max \lambda_{max}(\Sigma_1), \lambda_{max}(\Sigma_2)}$. We want our Gaussians to be less separated so optimization problem of learning GMM will be harder, so the lower c the harder optimization problem. Procedure of generating data sets will be as follows 2

5.3. Real world data

We'll be testing on the following real world datasets:

- Cloud Dataset: 10 dimensions, 1024 data points
- Breast Cancer Dataset: 22 dimensions, 569 data points

5.4. Experiments conclusions

As far as you can application of PSO algo in all of the featured experiments result in growth of LL criteria, especially in the case of comparison by iterations. In terms of comparison by time our algorithm is not optimized a lot (for example it can be parallelized) so we can fasten it up. But still in comparison by time out algo still gives LL growth.

A thing to mention is hyperparameters, the main of them are: σ_1, σ_2 used for randomly scatterring PSO particles around

Algorithm 1 Algorithm with reinitialization of EM

Input: d-dimensional dataset with N samples, number of iterations T, number of initial EM iterations T_{init}, number of components K, times of EM reinit T_1 , number of iteration for EM reinit T_2, number of particles M, rank of addition R, PSO hyperparameters η, c_1, c_2

Output: GMM solution $\{w_1, w_2, .., w_K, \mu_1, \Lambda_1, .., \mu_K, \Lambda_K\}$

run EM for T_{init} iterations
for $m = 1$ to M **do**
  $a^{(0)} \sim N(0, \sigma_1 I_d)$ ▷ Initialize parameters for particles
  for $r = 1$ to R **do**
    $b_r^{(0)} \sim N(0, \sigma_2 I_d)$
  end for
end for
for $t = 1$ to T **do**
  for $m = 1$ to M **do**
    Calculate precision matrices using low rank addition parameters
    Calculate log-likelihood
    Update personal best
  end for
  Update global best
  for $m = 1$ to M **do** ▷ PSO update
    Update particle's $\mu_m^{(t)}, w_m^{(t)}, a_m^{(t)}, b_m^{(t)}$
  end for
  if $t \mod (t \text{ div } T_1) = 0$ **then**
    for $m = 1$ to M **do**
      Calculate precision matrices using low rank addition parameters for each particle's personal best
      θ_{EM} = EM($\{w_{1,m}^{(PB)}, w_{2,m}^{(PB)}, .., w_{K,m}^{(PB)}, \mu_{1,m}^{(PB)}, \Lambda_{1,m}^{(PB)}, .., \mu_{K,m}^{(PB)}, \Lambda_{K,m}^{(PB)}\}$) ▷ Run EM for T_2 iterations
      Calculate log-likelihood ($\log p(\theta_{EM})$)
    end for
  end if
end for
Update global best EM reinit
return Global best EM reinit

initial GMM parameters. If you do choose them to be large then PSO will take a lot of time to converge and you will get bad LogLikelihood even after running EM algorithm if it was launched from bad point. If you choose them to be small then you won't get that much of a diversity in PSO particles and hence running EM will get similar results for all the points. From our experience it can be quite hard to guess right parameters and one of the future research directions is surely aims to find the way to automatically choose good hyperparameters.

6. Conclusions

Applying PSO for optimization of EM for GMM in general already showed itself as a good idea. Out contribution of initialization of particles through scattering across initial precision matrix via randomly sampling points in low rank subspace showed promising and beats vanilla EM in comparison by both budget of EM iterations and computation time through out algorithm is not optimized yet. Though it has some drawbacks in terms of choosing hyperparameters.

6.1. Future work

Hyperparameters is a huge part of our algorithm so we would work more on searching for optimal hyperparameters and how this optimality depends on data we are working on. Also we would like to try (Çağlar Arı et al., 2012) algorithm in out setting, scattering points accross the manifold near the gotten by inializing via vanilla EM GMM parameters, and trying to run PSO with parametrization by eigenvalues and Givens rotation. The main advantage this algorithm can bring to us is recovering PSO particles after doing EM, which we can't do in terms of our proposed parametrization and the main disadvantage will be $n(n+1)/2$ parameters we have to optimized on.

Algorithm 2 Algorithm for generation of synthetic dataset

Input: dimensionality d**, number of components** M**, number of samples** N**,** c **- separation coefficient**
Output: $\{x_1, x_2, .., x_N\}$**, where** $x_i \in \mathbb{R}^d$

$w \sim U[0,1]^d$
$w := w / \sum_{i=1}^{d} w_i$
$m := 0$
$GMM = \emptyset$
while $m < M$ **do**
  $\mu \sim U[0,20]^d$
  for $i = 1$ to d **do**
    $\lambda_j \sim U[1,16]$ ▷ Sample eigenvalues
    for $j = i+1$ to d **do**
      $\phi_{(p,q)} \sim U[-\pi/4, 3\pi/4]$ ▷ Sample Givens rotation angles
    end for

  $\Sigma = f(\lambda_1, .., \lambda_d, \phi_{(1,1)}, \phi_{(d,d)})$ ▷ Construct covariance matrix from Givens rotation angles and eigenvalues
$\mu_{ref}, \Sigma_{ref} \in$
  if $\|\mu - \mu_{ref}\|_2 \leq c\sqrt{d \max \lambda_{max}(\Sigma), \lambda_{max}(\Sigma_{ref})}$ **then**
    $GMM = GMM \cup \{\mu, \Sigma\}$
    $m = m + 1$
  end if
end for

Sample $x_1, .., x_N$ from M Gaussian Mixtures w.r.t. w
return $x_1, .., x_N$

References

Dasgupta, S. Learning mixtures of gaussians. *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pp. 634–644, 1999.

Hosseini, R. and Sra, S. Matrix manifold optimization for gaussian mixtures. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/dbe272bab69f8e13f14b405e038deb64-Paper.pdf.

Li, H. and Wang, J. Collaborative annealing power k-means++ clustering. *Knowledge-Based Systems*, 255:109593, 2022. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2022.109593. URL https://www.sciencedirect.com/science/article/pii/S0950705122008036.

Çağlar Arı, Aksoy, S., and Arıkan, O. Maximum likelihood estimation of gaussian mixture models using stochastic search. *Pattern Recognition*, 45(7):2804–2816, 2012. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2011.12.023. URL https://www.sciencedirect.com/science/article/pii/S0031320312000167.

A. Experiment results

B. Eigenvalues and Givens rotation matrix angles parametrization PSO algorithm

Comparison with random init EM by running time						
Dataset	EM iters PSO	EM iters EM	PSO LL	PSO Time, sec	EM LL	EM Time, sec
Breast Cancer	5000	12500	75.14 +- 0.07	57.3 +- 6	74.61 +- 0.19	75.9 +- 4
Synthetic (dim=50)	2000	5000	58.32 +- 0.21	96.53 +- 8.77	57.94 +- 0.3	104.6 +- 1.5

Table 1. PSO EM reinit and EM with random init algos comparison by time

Comparison with random init EM by EM iterations budget for real data						
Dataset	M	T_1	T_2	$T_1 \cdot T_2$	PSO LL	EM LL
Cloud Dataset	50	5	300	1500	25.065 +- 0.03	25.04 +- 0.015
Breast Cancer	100	EM	-	-	74.88 +- 0.47	74.64 +- 0.25

Table 2. M — number of particles and number of random inits for initial PSO EM, T_1 — Number of EM reinits from PSO particle coordinates, T_2 — max number of iterations of EM reinits, EM LL — log likelihood of best EM results from $2M$ random inited points and with max number of iterations equal to $T_1 \cdot T_2$, PSO LL — log likelihood of EM reinit from PSO particle coordinates. Recap that PSO algorithm in these experiments setup runs as follows:

1. Run EM with M random inits for $T_1 \cdot T_2$ iterations
2. Run PSO for $F \cdot T_1$ iterations, each F PSO iterations run GMM reinit that starts from particle coordinates and runs for T_2 iterations
3. Total number of EM iterations for this PSO algorithm will be $2 \cdot M \cdot T_1 \cdot T_2$.

Algorithm 3 Eigenvalue and Givens rotation angles parametrization algo (Çağlar Arı et al., 2012)

Input: d-dimensional dataset with N samples, number of iterations T_1, number of iteration for EM local convergence T_2, number of components K, number of particles M
Output: GMM solution $\{w_1, w_2, .., w_K, \mu_1, \Sigma_1, .., \mu_K, \Sigma_K\}$
for $t = 1$ to T_1 **do**
  for $m = 1$ to M **do**
    Construct K eigenvalue matrices
    Construct K eigenvector matrices by multiplying Givens rotation angles
    Run EM for local convergence for T_2 iterations T_2: number of EM iterations for each PSO iteration
    Compute K eigenvalue and eigenvector matrices via singular value decomposition of new covariance matrices
    Reorder eigenvalues and eigenvectors of each covariance matrix according to personal best
    Extract Givens rotation angles using QR factorization
    Replace particle's means, eigenvalues, and angles
    Calculate log-likelihood
    Update personal best
  end for
  Update global best
  for $m = 1$ to M **do**
    Update particle's means, eigenvalues, and angles
  end for
end for

Comparison with random init EM by EM iterations budget for synthetic data								
D	N_{comp}	C	M	T_1	T_2	$T_1 \cdot T_2$	PSO LL	EM LL
30	15	2	50	5	300	1500	32.32 +- 0.1	31.86 +- 0.05
50	15	2	50	5	300	1500	57.38 +- 1.9	56.628 +- 0.09
70	15	2	50	5	500	2500	97.06 +- 1.25	95.23 +- 0.56

Table 3. Comparison by iterations on synthetic data. For parameters discription of M, T_1, T_2, $T_1 \cdot T_2$, PSO LL, EM LL check 2. Synthetic dataset parameters: D – number of dimensions, N_{comp} – number of clusters in dataset, C – separation coefficient