The main idea of particle swarm optimization (PSO) algorithm is based on social intelligence when it simulates how a flock of birds search for food. Given a target function known as cost function *f*(***x***), recall that the optimization problem is to find out the minimum point ***x***\* known as minimizer or optimizer so that *f*(***x***\*) is minimal. As a convention, the optimization problem is global minimization problem. For maximization, it is simple to change a little bit our viewpoint.

Traditional local optimization methods such as Newton-Raphson and gradient descent along with global optimization methods require that *f*(***x***) is differentiable. Alternately, PSO does not require existence of differential. PSO scatters a population of candidate solutions (candidate optimizers) for ***x***\* and such population is called swarm whereas each candidate optimizer is called particle in the swarm. PSO is an iterative algorithm in which every particle is moved at each iteration so that it approaches the global optimizer ***x***\*. Movement of all particles is attracted by ***x***\*. In other words, such movement is attracted by minimizing *f*(***x***) so that *f*(***x***) is small enough. In PSO, ***x*** is considered as position of particle. It is focused that the movement of each particle is affected by its best position and the best position of the swarm. Note, the closer to ***x***\*, the better the position is.

As a formal definition, let be the swam of particles and let ***x****i* and ***p****i* be current position and best position of particle *i*. Moreover, the movement speed of particle *i* is specified by its velocity ***v****i*. The cost function *f*(***x***) is from real *n*-dimensional space ***R****n* to real space ***R***. Let ***p****g* is the best position of entire swarm. The closer to ***x***\*, the better the positions ***p****i* and ***p****g* are. It is expected that ***p****g* is equals to ***x***\* or is approximated to ***x***\*. The ultimate purpose of PSO is to determine ***p****g*.

Of course, ***x****i*, ***p****i*, and ***p****g* are *n*-dimensional points and ***v****i* is *n*-dimensional vector. Following is pseudo-code of PSO.

|  |
| --- |
| Input: the swam of particles along with their initialized positions and velocities.  Output: the best position ***p****g* of entire swarm with expectation that ***p****g* is equal or approximated to the global minimizer ***x***\*.  All current positions ***x****i* of all particles are initialized randomly. Moreover, their best positions are set to be their current positions such that ***p****i* = ***x****i*.  The global best position ***p****g* is assigned by the local best position ***p****i* such that *f*(***p****i*) is smallest among particles.  While terminated condition is not met do  For each particle *i* in *S*  Velocity of particle *i* is updated as follows:  Position of particle *i* is updated as follows:  If *f*(***x****i*) < *f*(***p****i*) then  The best position of particle *i* is updated: ***p****i* = ***x****i*  If *f*(***p****i*) < *f*(***p****g*) then  The best position of swarm is updated: ***p****g* = ***p****i*  End if  End if  End for  End while |

There are two most popular terminated conditions:

1. The cost function at ***p****g* which is evaluated as *f*(***p****g*) is small enough. For example, *f*(***p****g*) is smaller than a small threshold.
2. Or PSO ran over a large enough number of iterations.

Note, denotes component-wise multiplication of two points. For example, given two vectors ***x****i* = (*xi*1, *xi*2,…, *xin*)*T* and ***x****j* = (*xj*1, *xi*2,…, *xjn*)*T*, their component-wise multiplication is:

Function *U*(0, *ϕ*1) generates a random vector whose elements are random numbers in the range [0, *ϕ*1]. Similarly, function *U*(0, *ϕ*2) generates a random vector whose elements are random numbers in the range [0, *ϕ*2].

The convergence of GEM is based on the assumption that *Q*(Θ’ | Θ) is smooth enough but *Q*(Θ’ | Θ) may not be smooth in practice, for example, *f*(*X* | Θ) may be discrete probability function. For example, when *f*(*X* | Θ) and *k*(*X* | *Y*, Θ) are discrete, equation 2.8 becomes

This discussion section goes beyond traditional variants of GEM algorithm when *Q*(Θ’ | Θ) is not smooth. Therefore, heuristic optimization methods which simulate social behavior, such as particle swarm optimization (PSO) algorithm (Poli, Kennedy, & Blackwell, 2007) and artificial bee colony (ABC) algorithm, are useful in case that there is no requirement of existence of derivative. Moreover, these heuristic methods aim to find global optimizer. I propose an association of GEM and PSO which produces a so-called quasi-PSO-GEM algorithm in which M-step is implemented by one-time PSO (Wikipedia, Particle swarm optimization, 2017). Given current *t*th iteration, Θ(*t*) is modeled as swarm’s best position. Suppose there are *n* particles and each particle *i* has current velocity *Vi*(*t*), current positions Ψ*i*(*t*), and best position Φ*i*(*t*). At each iteration, it is expected that these particles move to swarm’s new best position which is the next parameter Θ(*t*+1). The swarm’s best position at the final iteration is expected as Θ*\**. Table 6.2 is the proposal of quasi-PSO-GEM algorithm.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *E-step*:  As usual, *Q*(Θ | Θ(*t*)) is determined based on current Θ(*t*) according to equation 2.8. Actually, *Q*(Θ | Θ(*t*)) is formulated as function of Θ.  *M-step* includes four sub-steps:   1. Calculating the next velocity *Vi*(*t*+1) of each particle based on its current velocity *Vi*(*t*), its current positions Ψ*i*(*t*), its best positions Φ*i*(*t*), and the swarm’s best position Θ(*t*):  |  |  | | --- | --- | |  | (6.1) |   Where *ω*, *ϕ*1, and *ϕ*2 are particular parameters of PSO (Poli, Kennedy, & Blackwell, 2007, pp. 3-4) whereas *r* is a random number such that 0 < *r* < 1 (Wikipedia, Particle swarm optimization, 2017).   1. Calculating the next position Ψ*i*(*t*+1) of each particle based on its current position Ψ*i*(*t*) and its current velocity *Vi*(*t*):  |  |  | | --- | --- | |  | (6.2) |  1. If *Q*(Φ*i*(*t*) | Θ(*t*)) < *Q*(Ψ*i*(*t*+1) | Θ(*t*)) then, the next best position of each particle *i* is re-assigned as Φ*i*(*t*+1) = Ψ*i*(*t*+1). Otherwise, such next best position is kept intact as Φ*i*(*t*+1) = Φ*i*(*t*). 2. The next parameter Θ(*t*+1) is the swarm’s new best position over the best positions of all particles:  |  |  | | --- | --- | |  | (6.3) |   If the bias |Θ(*t*+1) – Θ(*t*)| is small enough, the algorithm stops. Otherwise, Θ(*t*+1) and all *Vi*(*t*+1), Ψ*i*(*t*+1), Φ*i*(*t*+1) become current parameters in the next iteration. |

**Table 6.1.** E-step and M-step of the proposed quasi-PSO-GEM

At the first iteration, each particle is initialized with Ψ*i*(1) = Φ*i*(1) = Θ(1) and uniformly distributed velocity *Vi*(1). Note, Θ(1) is initialized arbitrarily. Other termination criteria can be used, for example, *Q*(Θ | Θ(*t*)) is large enough or the number of iterations is large enough.

We cannot prove mathematically convergence of quasi-PSO-GEM but we expect that Θ(*t*+1) resulted from equation 6.3 is an approximation of Θ*\** at the last iteration after a large enough number of iterations. However, quasi-PSO-GEM tendentiously approaches global maximizer of *L*(Θ), regardless of whether *L*(Θ) is concave. Hence, it is necessary to make experiment on quasi-PSO-GEM.

There are many other researches which combine EM and PSO but the proposed quasi-PSO-GEM algorithm has different ideology when it one-time PSO is embed into M-step to maximize *Q*(Θ | Θ(*t*)) and so the ideology of quasi-PSO-GEM is near to the ideology of Newton-Raphson process. With different viewpoint, some other researches combine EM and PSO in order to solving better a particular problem instead of improving EM itself. For example, Ari and Aksoy (Ari & Aksoy, 2010) used PSO to solve optimization problem of the clustering algorithm based on mixture model and EM. Rajeswari and Gunasundari (Rajeswari & Gunasundari, 2016) proposed EM for PSO based weighted clustering. Zhang, Zhuang, Gao, Luo, Ran, and Du (Zhang, et al., 2014) proposed a so-called PSO-EM algorithm to make optimum use of PSO in partial E-step in order solve the difficulty of integrals in normal compositional model. Golubovic, Olcan, and Kolundzija (Golubovic, Olcan, & Kolundzija, 2007) proposed a few modifications of the PSO algorithm which are applied to EM optimization of a broadside antenna array. Tang, Song, and Liu (Tang, Song, & Liu, 2014) proposed a hybrid clustering method based on improved PSO and EM clustering algorithm to overcome drawbacks of EM clustering algorithm. Tran, Vo, and Lee (Tran, Vo, & Lee, 2013) proposed a novel clustering algorithm for image segmentation by employing the arbitrary covariance matrices that uses PSO for the estimation of Gaussian mixture models.