Improved Particle Swarm Optimization and Applications to Hidden Markov Model and Ackley Function

Saeed Motiian[1] and Hamid Soltanian-Zadeh, [1,2] *Senior Member, IEEE*
[1] Control and Intelligent Processing Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran 14395 515 Iran
[2] Image Analysis Laboratory, Radiology Department, Henry Ford Hospital, Detroit, Michigan 48202, USA
Emails: saeed.motiian@gmail.com, hszadeh@ut.ac.ir

Abstract*—*Particle Swarm Optimization (PSO) is an algorithm based on social intelligence, utilized in many fields of optimization. In applications like speech recognition, due to existence of high dimensional matrices, the speed of standard PSO is very low. In addition, PSO may be trapped in a local optimum. In this paper, we introduce a novel algorithm that is faster and generates superior results than the standard PSO. Also, the probability of being trapped in a local optimum is decreased. To illustrate advantages of the proposed algorithm, we use it to train a Hidden Markov Model (HMM) and find the minimum of the Ackley function.

Key words: Particle Swarm Optimization (PSO), Hidden Markov Model, Optimization, Ackley function.

I. Introduction

The Particle Swarm Optimization (PSO) is an algorithm inspired by nature. This algorithm has been improved for specific applications [1], [2]. One of applications of the PSO is in Speech Processing. The Hidden Markov Model (HMM) is the most successful machine learning technique in Speech Processing [3]. There are several algorithms to train a HMM like Beum-Welch, e.g., PSO and genetic algorithm. In [4], it is illustrated that a technique based on PSO has been the best technique among other techniques. Since there are matrixes with high dimensions in the training of HMM, the standard PSO is not very fast. In this paper, we introduce an algorithm based on PSO that is faster than other techniques. This novel algorithm tries not to be trapped in a local optimum.

In next section, we describe the standard PSO and Hidden Markov Model and Ackley problem.

II. Particle Swarm Optimization and its applications

Particle swarm optimization (PSO) is inspired by social behavior of bird flocking or fish schooling, originally introduced by Eberhart and Kennedy [5] in 1995. PSO does not use the gradient of the objective function, therefore, PSO does not require for the objective function to be differentiable as is required by classic optimization methods. Therefore, PSO is suitable for optimization problems that are relatively irregular, noisy, dynamic, etc.

In PSO, there are some particles that search the space of the problem. In the next step, particles change their positions based on the knowledge of themselves and other particles. Thus, positions of particles in swarm influence each particle. After several steps, particles move toward an optimum. These particles are imaginary objects. The goal is to search possible solutions of a problem to find an optimum solution.

Each particle updates its position based on the following factors: its best solution (**Pbest**), a best solution of swarm (**gbest**), and a best solution of its neighbors (**nbest**).

Pbest is the best position that each particle found by the pervious step. Each particle has a special Pbest. gbest is the best position that all of particles found by pervious step. gbest is same for all of particles. We can consider some neighbors for each particle. nbest is the best position that all of neighbors of each particle found by pervious step. Each particle has a special nbest.

There are several means to initialize positions of particles. One of them used in this paper is random initialization. In random initialization, particles are placed in random positions in the space. The update equation of positions is:

$$V = c_0 X + c_1 r_1 \times (pbest - X) + c_2 r_2 \times (gbest - X) + c_3 r_3 \times (nbest - X) \quad (1)$$

$$X = X + V \quad (2)$$

where c_0 , c_1 , c_2 and c_3 are constants and r_1 , r_2 and r_3 are random numbers between 0 and 1.

In the next section, we introduce HMM training based on the PSO.

A. Hidden Markov Model and PSO

A Hidden Markov Model (HMM) is a statistical Markov model used for systems that are assumed to be a Markov process with hidden states.

A continuous HMM is characterized by its parameters vector $\lambda = \{\pi, A, C, U, \Sigma\}$, where π is the initial state probability matrix and $A = \{a_{ij}\}$ is the state transition

978-1-61284-925-6/11/$26.00 ©2011 IEEE

probability distribution in which a_{ij} is the transition probability from state i (at time t) to j (at time $t+1$) (N is the number of state) [6]:

$$a_{ij} = P[q_{t+1} = j | q_t = i], \qquad 1 \le i, j \le N \tag{3}$$

a_{ij} must satisfy 2 constraints:

1. $\sum_{j=1}^{N} a_{ij} = 1, \qquad 1 < i < N$
2. $a_{ij} > 0, \qquad 1 \le i, j \le N$ (4)

$b_j(o_t)$ is the probability density function (PDF) of producing observation vector o_t in state j. In the field of speech recognition, we use the following Gaussian Mixture Model (GMM):

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} N(o_t, U_{jm}, \Sigma_{jm}) \tag{5}$$

where c_{jm} is the mixture coefficient for the mth mixture in state j. c_{jm} is similar to a_{ij} and must satisfy the following 2 constraints:

1. $\sum_{m=1}^{M} c_{jm} = 1, \qquad 1 < j < N$
2. $c_{jm} > 0, \qquad 1 \le j \le N$ (6)

$U_{jm} = [U_{jmd}]_{d=1}^{D}$ is a mean vector and Σ_{jm} is a covariance matrix, where D is the dimension of feature vectors.

These parameters can be estimated by maximizing the model likelihood probability $P(0|\lambda)$. which can be calculated by the following forward procedure [6]:

Consider the forward variable $\alpha_t(i)$ defined as

$$a_t(i) = P(o_1 o_2 \dots o_t, q_t = i | \lambda) \tag{7}$$

which can be calculated as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(o_1), \tag{8}$$

2. Induction

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \qquad 1 \le t \le T-1 \quad 1 \le j \le N \tag{9}$$

3. Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{10}$$

As mentioned above, HMM is defined using the vector $\lambda = \{\pi, A, C, U, \Sigma\}$. The main goal in HMM training is to find the best λ that maximizes $P(o|\lambda)$. First, we must introduce a fitness function. In this case, the fitness function is:

$$f(\lambda) = logP(o|\lambda) = \frac{1}{L}\sum_{l=1}^{L} logP(o^l|\lambda) \tag{11}$$

where $o^l = [o_1^l, o_2^l, \dots, o_T^l]$ is the lth observation sequence and L is the number of observations. Then, we consider k particles with random positions and calculate their fitness function. Next, according to the previous section, we find pbest, gbest, and nbest.

In the following, we update positions according to Eq (1) and the algorithm moves to next iteration.

This procedure continues until a stopping condition is satisfied.

In this paper, the stopping condition is ignored and the algorithm repeats for an arbitrary number of iterations.

B. Ackley Problem and PSO

The ***Ackley Problem*** [7] is a minimization problem. This problem was originally defined for two dimensions, but recently the problem has been generalized to N dimensions [8].

The Ackley function is:

$$f(x_1, x_2, \dots, x_n) = -c_1 . exp\left(-c_2 \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(c_3 . x_i)\right) + c_1 + e \tag{12}$$

where $c_1 = 20$, $c_2 = 0.2$, and $c_3 = 2\pi$. Also, n is the number of dimensions. In this paper, we use n=100.

The 2 dimensions Ackley function is illustrated in Figure 1[9]. The minimum of Ackley function is 0 that occurs at the origin. Similar to the previous section, we consider k particles with random positions and calculate their fitness (Eq. (12)). Then, we find pbest, gbest, and nbest and update particles positions according to Eq. (1).

In next section we introduce our new algorithm.

III. Proposed Algorithm Based on PSO

In the standard PSO, some of particles may be trapped in a local optimum and these particles do not play enough roles in finding the global optimum. Therefore, in our algorithm, we try to discard these particles and replace them with new particles.

In our algorithm, after a number of iterations, we find particles that have relatively small fitness functions and eliminate them. Therefore, this increases the speed of the algorithm. To cover the entire search space, we add a new

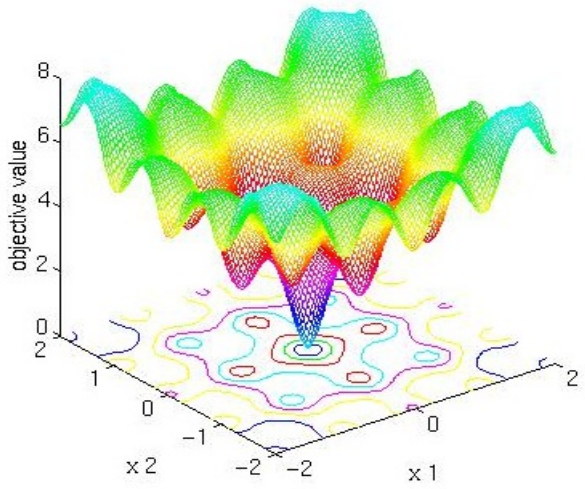

Fig1. Two-dimensional Ackley function.

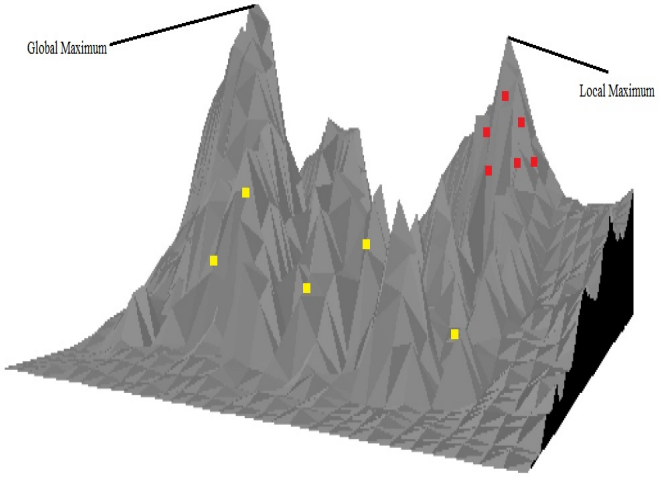

Fig2. Consider PSO algorithm is trapped in a local optimum (Red squares). By adding some new particles (Yellow squares) with random positions and considering them as a new swarm. Algorithm can escape from a local optimum and likely, it can reach a global optimum.

particle (or new particles) with random coordinates. We do not allow the new particles to be eliminated in the 2 next stages, even if they show worse fitness values. This is done to give them sufficient time to reach their optimum. In our work, we apply the elimination/addition stage once after every 20 iterations. The total number of stages is arbitrarily. Also, PSO greatly depends on initial positions of particles. In our algorithm, by adding some new particles, this problem is removed. The number of stages of elimination/addition and the number of particles of eliminated/added can be changed for different applications. For example, in complicated applications (applications with a lot of local optimums), the number of eliminated/added particles must be increased or in simple applications, it is not necessary that some particles were added and just must be eliminated.

In the other hand, due to existence of $c_2 r_2 \times (gbest - X)$ in Eq. (1) all new particles move toward the best optimum found by now. If we update positions of new particles according to Eq.1, new particles may reach a local optimum like other particles. Therefore, to escape from local optimums and obtain good and fast results in very complicated applications, it is necessary for new particles that $c_2 = 0$ and $c_3 \neq 0$ in Eq. (1) (neighbors of each particle are chosen another new particles). Actually, we create some isolated swarms in the space. By doing so, our algorithm can escapes from a local optimum. (Fig. 2[10])

The new algorithm can be used in all fields. The most important advantages of the algorithm are revealed in complicated applications such as HMM training.

IV. Experimental Results

A. Ackley Function

We compare the results of our new algorithm with the standard PSO (see Table 1 and 2). Both of the standard PSO and our algorithm are started with the same initial conditions. The constant in Eq. (12) are: $c_0 = 0.8$, $c_1 = 1.5$, $c_2 = 1.5$ and $c_3 = 0$ (for the initial swarm). To obtain reliable results, we performed both algorithms 5,000 times and they included 150 iterations in each time. In our algorithm, we eliminate 6 particles and add 1 particle in each stage of elimination/addition. We do not eliminate the new particles in the next two stages. If we do not consider a new swarm, the time of our algorithm decreases to around 59% compared to the time of standard PSO. If we do consider a new swarm, the time of our algorithm decreases to around 65% compared to the time of standard PSO. Nevertheless, the result of our algorithm is better than the result of the standard PSO. This is the main advantage of our new algorithm.

B. HMM Training

We performed HMM training with our algorithm and the standard PSO. In this case, like the previous case, our algorithm has a better solution and takes less time.

We use 1,000 digits utterance spoken by 10 speakers as a close set for the HMM training. Also, we use 500 digits utterance spoken by 5 speakers as a close set for the HMM testing.

In this paper, we use mel-frequency cepstral coefficients (MFCCs) as a features vector. It includes 12 MFCCs and their first order derivatives. There are some approaches to extract MFCCs. Recently new approaches have been designed [11].

Table 1. The comparison between the standard PSO and the new algorithm for n=100 without considering a new swarm (n is dimensions of Ackley function). in this experiment $c_0 = 0.8$, $c_1 = 1.5$, $c_2 = 1.5$ and $c_3 = 0$ for both algorithms.

Algorithm	Number of performances	Average Minimum
Standard PSO	5,000	1.0155×10^{-13}
Proposed PSO	5,000	3.2736×10^{-14}

In this experiment, $c_1 = 1.5\,, c_2 = 1.5\,, c_3 = 0$ and $c_0 = [0.8\ 0.3]$. Through the iterations, c_0 decreases due to forgetting factor. Similar to the previous section, all conditions are the same for both algorithms. The number of particles is 30. First we perform HMM training without considering a new swarm and just add some particles. The results are presented in Table 3.

Table 2. The comparison between the standard PSO and the new algorithm for n=100 with considering a new swarm (n is dimensions of Ackley function).

Algorithm	The Number of Performances	Average Minimum
Standard PSO	5,000	1.0155×10^{-13}
Proposed PSO	5,000	3.1348×10^{-14}

Secondly, we applied our algorithm considering a new swarm. The results are presented in Table 4.

Table 3. Comparison of HMM training between standard PSO and our algorithm when we do not consider a new swarm. In this experiments, N=5 and M=3.Our algorithm generates results that are superior to those of the standard PSO and the time of our algorithm is less, around 55% compared to the time of the standard PSO. in this experiment $c_0 = [0.8 - 0.2]\,, c_1 = 1.5\,, c_2 = 1.5$ and $c_3 = 0$ for both algorithms.

Algorithm	Average log probability [dB]
Standard PSO	-80.82
Proposed PSO	-76.22

This experiment proved our statements on superiority of our algorithm. According to Table 3 and 4, we can see that our algorithm is faster and generates superior results than the standard PSO. Also as mentioned before, in complicated applications like HMM training, if we consider a new swarm, we can reach a better result. But the time will be increased due to surplus computations rather than we do not consider a new swarm.

Table 4. Comparison of HMM training between standard PSO and our algorithm when we consider a new swarm. In this experiments, N=5 and M=3. Our algorithm generates results that are superior to those of the standard PSO and the time of our algorithm is less, around 60% compared to the time of the standard PSO.

Algorithm	Average log probability [dB]
Standard PSO	-80.82
Proposed PSO	-73.56

Finally, we compare PSO, GA, and Beum-Welch algorithms to show superiority of PSO. We perform HMM training under the previous conditions. The results are presented in Table 5. As mentioned before, PSO is the best among the others.

V. Conclusions

In this paper, we introduced a novel algorithm based on PSO that is faster and generates superior results than the standard PSO. We eliminate some of the particles that do not have good fitness values.

Table 5. Comparison between PSO, GA, and Beum-Welch algorithms.

Algorithm	Average log probability [dB]
PSO	-80.82
GA	-82.31
Beum-Welch	-85.76

With this elimination, the speed of the algorithm increases. To escape from local optima, we added new particles and give them enough time to reach their optimum. Also, by changing the number of particles that are eliminated/added in each stage, we find the best solution for each specific field. Also, the number of stages for elimination/addition can be changed. It has been mentioned that we can consider new particles as a new swarm for complicated applications. This new swarm is isolated from initial swarm. Therefore, our algorithm can escape from local optima. In this paper, we use 4 stages for 30 particles and 6 eliminations and 1 addition.

ACKNOWLEDGMENT

The authors would like to thank Dr. H. Sheikh-Zadeh for useful comments on Hidden Markov Model training. They would also like to thank M. Amian and V. Jabi, the members of bioinstrumentation lab at University of Tehran and M. Fraji, the member of speech processing lab at Amirkabir University of Technology, for useful discussions on PSO algorithm and speech processing.

REFERENCES

[1] J. Liu, X. Qiu, "A Novel Hybrid PSO-BP Algorithm for Neural Network Training," 2009 *International Joint Conference on Computational Sciences and Optimization,* vol. 1, pp.300-303.

[2] G. Zeng, Y. Jiang, "A Modified PSO Algorithm with Line Search," *Computational Intelligence and Software Engineering (CiSE)*, 2010 International, Wuhan.

[3] K. Watts, O. Yamagishi, J. King, S. Berkling, "Synthesis of Child Speech With HMM Adaptation and Voice Conversion", *Audio, Speech, and Language Processing*. vol.18, pp.1005-1015, July 2010.

[4] L. Xue, J. Yin, Z. Jiang, "A particle swarm optimization for hidden Markov model training,", *ICSP 2006 proceedings of IEEE.*

[5] J. Kennedy, R.C. Eberhart. "Particle Swarm Optimization", *In Proc .of the IEEE Int Conf. Neural Networks*, 1995.

[6] L. Rabiner, B.H. Juang, "*Fundamentals of speech recognition*",Prentice-Hall Inc., 1993.

[7] D.H. Ackley. "*A connectionist machine for genetic hillclimbing*". Boston: Kluwer Academic Publishers, 1987.

[8] T. Bäck. "*Evolutionary algorithms in theory and practice*". Oxford University Press, 1996.

[9] D.Izzo, "The Ackley problem", http://pagmo.sourceforge.net/doc/classpagmo_1_1problem_1_1ackley.html, 2010.

[10] J.Evershed, k.Fith, K.Peadon, "Particle Swarm Optimization (PSO) Visualisation (or "PSO Visualization")," http://www.projectcomputing.com/index.html, 2004.

[11] M.A. Hossan, S. Memon, M.A. Gregory, "A novel approach for MFCC feature extraction," *Signal Processing and Communication Systems (ICSPCS)*, 2010.