PSO Tutorial

A Chinese version is also available.

1. Introduction

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. The detailed information will be given in following sections.

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

The remaining of the report includes six sections:

Background: artificial life.
The Algorithm
Comparisons between Genetic algorithm and PSO
Artificial neural network and PSO
PSO parameter control
Online resources of PSO

2. Background: Artificial life

The term "Artificial Life" (ALife) is used to describe research into human-made systems that possess some of the essential properties of life. ALife includes two-folded research topic: (http://www.alife.org)

1. ALife studies how computational techniques can help when studying biological phenomena
2. ALife studies how biological techniques can help out with computational problems

The focus of this report is on the second topic. Actually, there are already lots of computational techniques inspired by biological systems. For example, artificial neural network is a simplified model of human brain; genetic algorithm is inspired by the human evolution.

Here we discuss another type of biological system - social system, more specifically, the collective behaviors of simple individuals interacting with their environment and each other. Someone called it as swarm intelligence. All of the simulations utilized local processes, such as those modeled by cellular automata, and might underlie the unpredictable group dynamics of social behavior.

Some popular examples are floys and boids. Both of the simulations were created to interpret the movement of organisms in a bird flock or fish school. These simulations are normally used in computer animation or computer aided design.

There are two popular swarm inspired methods in computational intelligence areas: Ant colony optimization (ACO) and particle swarm optimization (PSO). ACO was inspired by the behaviors of ants and has many successful applications in discrete optimization problems. (http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html)

The particle swarm concept originated as a simulation of simplified social system. The original intent was to graphically simulate the choreography of bird of a bird block or fish school. However, it was found that particle swarm model can be used as an optimizer. (http://www.engr.iupui.edu/~shi/Coference/psopap4.html)

3. The algorithm

As stated before, PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (a)
present[] = persent[] + v[] (b)

v[] is the particle velocity, persent[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2.

The pseudo code of the procedure is as follows

```
For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest) in history
            set current value as the new pBest
    End

    Choose the particle with the best fitness value of all the particles as the gBest
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End
While maximum iterations or minimum error criteria is not attained
```

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.

4. Comparisons between Genetic Algorithm and PSO

Most of evolutionary techniques have the following procedure:
1. Random generation of an initial population
2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to 2.

From the procedure, we can learn that PSO shares many common points with GA. Both algorithms start with a group of a randomly generated population, both have fitness values to evaluate the population. Both update the population and search for the optimium with random techniques. Both systems do not guarantee success.

However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm.

Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gBest (or lBest) gives out the information to others. It is a one -way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

5. Artificial neural network and PSO

An artificial neural network (ANN) is an analysis paradigm that is a simple model of the brain and the back-propagation algorithm is the one of the most popular method to train the artificial neural

network. Recently there have been significant research efforts to apply evolutionary computation (EC) techniques for the purposes of evolving one or more aspects of artificial neural networks.

Evolutionary computation methodologies have been applied to three main attributes of neural networks: network connection weights, network architecture (network topology, transfer function), and network learning algorithms.

Most of the work involving the evolution of ANN has focused on the network weights and topological structure. Usually the weights and/or topological structure are encoded as a chromosome in GA. The selection of fitness function depends on the research goals. For a classification problem, the rate of mis-classified patterns can be viewed as the fitness value.

The advantage of the EC is that EC can be used in cases with non-differentiable PE transfer functions and no gradient information available. The disadvantages are 1. The performance is not competitive in some problems. 2. representation of the weights is difficult and the genetic operators have to be carefully selected or developed.

There are several papers reported using PSO to replace the back-propagation learning algorithm in ANN in the past several years. It showed PSO is a promising method to train ANN. It is faster and gets better results in most cases. It also avoids some of the problems GA met.

Here we show a simple example of evolving ANN with PSO. The problem is a benchmark function of classification problem: iris data set. Measurements of four attributes of iris flowers are provided in each data set record: sepal length, sepal width, petal length, and petal width. Fifty sets of measurements are present for each of three varieties of iris flowers, for a total of 150 records, or patterns.

A 3-layer ANN is used to do the classification. There are 4 inputs and 3 outputs. So the input layer has 4 neurons and the output layer has 3 neurons. One can evolve the number of hidden neurons. However, for demonstration only, here we suppose the hidden layer has 6 neurons. We can evolve other parameters in the feed-forward network. Here we only evolve the network weights. So the particle will be a group of weights, there are 4*6+6*3 = 42 weights, so the particle consists of 42 real numbers. The range of weights can be set to [-100, 100] (this is just a example, in real cases, one might try different ranges). After encoding the particles, we need to determine the fitness function. For the classification problem, we feed all the patterns to the network whose weights is determined by the particle, get the outputs and compare it the standard outputs. Then we record the number of misclassified patterns as the fitness value of that particle. Now we can apply PSO to train the ANN to get lower number of misclassified patterns as possible. There are not many parameters in PSO need to be adjusted. We only need to adjust the number of hidden layers and the range of the weights to get better results in different trials.

6. PSO parameter control

From the above case, we can learn that there are two key steps when applying PSO to optimization problems: the representation of the solution and the fitness function. One of the advantages of PSO is that PSO take real numbers as particles. It is not like GA, which needs to change to binary encoding, or special genetic operators have to be used. For example, we try to find the solution for f(x) = x1^2 + x2^2+x3^2, the particle can be set as (x1, x2, x3), and fitness function is f(x). Then we can use the standard procedure to find the optimum. The searching is a repeat process, and the stop criteria are that the maximum iteration number is reached or the minimum error condition is satisfied.

There are not many parameter need to be tuned in PSO. Here is a list of the parameters and their typical values.

The number of particles: the typical range is 20 - 40. Actually for most of the problems 10 particles is large enough to get good results. For some difficult or special problems, one can try 100 or 200 particles as well.

Dimension of particles: It is determined by the problem to be optimized,

Range of particles: It is also determined by the problem to be optimized, you can specify different ranges for different dimension of particles.

Vmax: it determines the maximum change one particle can take during one iteration. Usually we set the range of the particle as the Vmax for example, the particle (x1, x2, x3)
X1 belongs [-10, 10], then Vmax = 20

Learning factors: c1 and c2 usually equal to 2. However, other settings were also used in different papers. But usually c1 equals to c2 and ranges from [0, 4]

The stop condition: the maximum number of iterations the PSO execute and the minimum error requirement. for example, for ANN training in previous section, we can set the minimum error requirement is one mis-classified pattern. the maximum number of iterations is set to 2000. this stop condition depends on the problem to be optimized.

Global version vs. local version: we introduced two versions of PSO. global and local version. global version is faster but might converge to local optimum for some problems. local version is a little bit slower but not easy to be trapped into local optimim. One can use global version to get quick result and use local version to refine the search.

Another factor is inertia weight, which is introduced by Shi and Eberhart. If you are interested in it, please refer to their paper in 1998. (Title: A modified particle swarm optimizer)

7. Online Resources of PSO

The development of PSO is still ongoing. And there are still many unknown areas in PSO research such as the mathematical validation of particle swarm theory.

One can find much information from the internet. Following are some information you can get online:

http://www.particleswarm.net lots of information about Particle Swarms and, particularly, Particle Swarm Optimization. Lots of Particle Swarm Links.

http://icdweb.cc.purdue.edu/~hux/PSO.shtml lists an updated bibliography of particle swarm optimization and some online paper links

http://www.researchindex.com/ you can search particle swarm related papers and references.

References:

http://www.engr.iupui.edu/~eberhart/

http://users.erols.com/cathyk/jimk.html

http://www.alife.org

http://www.aridolan.com

http://www.red3d.com/cwr/boids/

http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html

http://www.engr.iupui.edu/~shi/Coference/psopap4.html

Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proc. IEEE int'l conf. on neural networks Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.

Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan, 1995.

Eberhart, R. C. and Shi, Y. Particle swarm optimization: developments, applications and resources. Proc. congress on evolutionary computation 2001 IEEE service center, Piscataway, NJ., Seoul, Korea., 2001.

Eberhart, R. C. and Shi, Y. Evolving artificial neural networks. Proc. 1998 Int'l Conf. on neural networks and brain pp. PL5-PL13. Beijing, P. R. China, 1998.

Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization. Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf., Springer-Verlag, Berlin, San Diego, CA., 1998.

Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization. Evolutionary Programming VII: Proc. EP 98 pp. 591-600. Springer-Verlag, New York, 1998.

Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation pp. 69-73. IEEE Press, Piscataway, NJ, 1998