UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES
Département d'informatique

MASTER OF COMPUTER SCIENCE

MÉTAHEURISTIQUES POUR L'OPTIMISATION
HOMEWORK ASSIGNMENT REPORT

Series 5 : Network learning using PSO

Author :
Mohammad Oday DARWICH

Professors :
M. Bastien CHOPARD
Mme. Aziza MERZOUKI
M. Gregor CHLIAMOVITCH

14 novembre 2015

Introduction

In this TP we will implement the PSO algorithm to learn the weights of a Neural Netwrok model to classify hand written digit, so we are talking about pattern recognition.

PSO utilizes a very simplified model of social behavior to solve the optimization problems, in a co-operative and intelligent framework. PSO is one of the most useful and famous metaheuristics and it is successfully applied to various optimization problems.

Note :In the code you will see that all parameters are described very well in a comment.

1 Implementation and results

In the implementation i did the execution varying the *Number of Particles* ***nPop*** and the *Maximum number of iteration* ***MaxIT*** to see the difference with these statistics. Below our defined parameters :

```
%% PSO Parameters
nPop=10;              % Number of particles
MaxIt = 100;          % Maximum Number of Iterations
c1=2.2;                 % Personal Learning Coefficient
c2=2;                 % Global Learning Coefficient
w =0.9;               % Inertia Weight
Pv = 0.9;             % Particule sp

r1=rand(1,nPop);      % Random for the PSO particules
r2=rand(1,nPop);      % Random for the PSO particules
```

Listing 1 – PSO param

1.1 Experiment 1.1 with nPop = 30 and MaxIt = 100

```
%% PSO Parameters
nPop=30;              % Number of particles
MaxIt = 100;          % Maximum Number of Iterations
```

Listing 2 – Exp with nPop

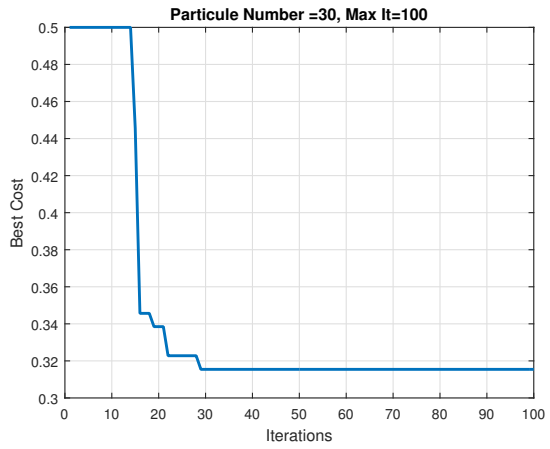

FIGURE 1

Exec	Fitness Value	Prediction error
1	0.224530737025400	0.270000000000000
2	0.296091873819556	0.300000000000000
3	0.253013247456068	0.500000000000000
4	0.250229805297489	0.500000000000000
5	0.265996917259017	0.500000000000000
6	0.259154805665461	0.260000000000000
7	0.250892291967106	0.500000000000000
8	0.259340958433906	0.500000000000000
9	0.273909563174873	0.495000000000000
10	0.250116076608497	0.500000000000000
Mean	0.2583	0.4325

FIGURE 2

1.2 Experiment 1.2 with nPop = 30 and MaxIt = 300

```
%% PSO Parameters
nPop=30;              % Number of particles
MaxIt = 1000;          % Maximum Number of Iterations
```

Listing 3 – Exp with nPop

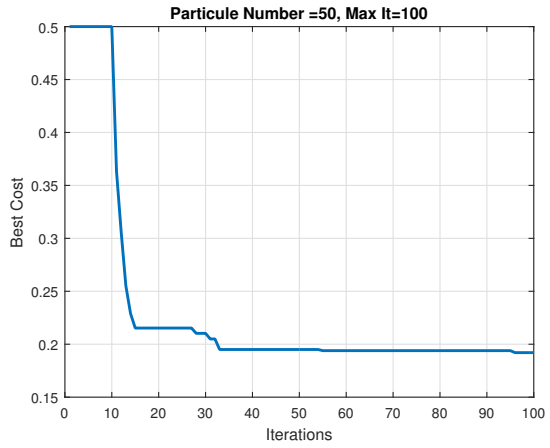

FIGURE 3

Exec	Fitness Value	Prediction error
1	0.324869418219014	0.530000000000000
2	0.257075971073757	0.260000000000000
3	0.167485446086770	0.175000000000000
4	0.271571786704650	0.495000000000000
5	0.233432880169853	0.235000000000000
6	0.287162631107825	0.510000000000000
7	0.222811446451344	0.230000000000000
8	0.227401382878207	0.345000000000000
9	0.149962454240649	0.150000000000000
10	0.171024560383294	0.175000000000000
Mean	0.2313	0.3105

FIGURE 4

1.3 Experiment 1.3 with nPop = 10 and MaxIt = 100

```
%% PSO Parameters
nPop=10;              % Number of particles
MaxIt = 100;          % Maximum Number of Iterations
```

Listing 4 – Exp with nPop

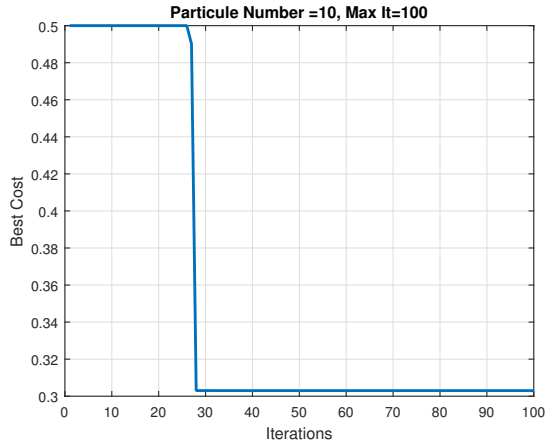

FIGURE 5

Exec	Fitness Value	Prediction error
1	0.234047867361561	0.240000000000000
2	0.309969555076566	0.500000000000000
3	0.242414230565715	0.295000000000000
4	0.260000000000000	0.500000000000000
5	0.121444015487047	0.495000000000000
6	0.280650339549293	0.340000000000000
7	0.264599931807714	0.330000000000000
8	0.254418819164213	0.465000000000000
9	0.327201391129398	0.480000000000000
10	0.373258342588827	0.205000000000000
Mean	0.2771	0.3850

FIGURE 6

1.4 Experiment 1.4 with nPop = 10 and MaxIt = 300

```
%% PSO Parameters
nPop=10;            % Number of particles
MaxIt = 10100;        % Maximum Number of Iterations
```

Listing 5 – Exp with nPop

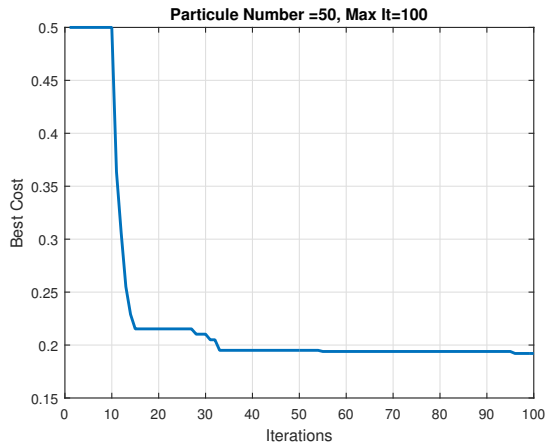

FIGURE 7

Exec	Fitness Value	Prediction error
1	0.234047867361561	0.240000000000000
2	0.309969555076566	0.310000000000000
3	0.242414230565715	0.260000000000000
4	0.260000000000000	0.260000000000000
5	0.121444015487047	0.125000000000000
6	0.280650339549293	0.300000000000000
7	0.264599931807714	0.270000000000000
8	0.254418819164213	0.475000000000000
9	0.327201391129398	0.505000000000000
10	0.373258342588827	0.380000000000000
Mean	0.2668	0.3125

FIGURE 8

2 Interpretation

We see from the experiences of 1.3 and 1.4 that there are no big difference in the prediction, it's 1 % , instead there are a little influence of the **MaxIt**, the fitness increase for 4.2 %.
Comparing the experience 1.3 and the 1.1 see that the *Perror* less then exp 1.3 by 0.88 %.

so that mean increasing the number of particles may the system learn more the behavior and so we get better fitness by 4.2 % more then a 1/3 number of particles of the exp 1.1.

So, from the 4th experiments, 1.1, 1.2, 1.3 and 1.4, we can deduce that more the number of iteration increase more we will have a well training but we have to do also the trade off for choosing the number of particles because decreasing the value of *nPop* may give a bad result, and increasing it do not mean we will get good results.

3 Conclusion

We see that particles are independent so each one is like a software and there are informations to be communicated between each other, so it's kind of *parallelism algorithm*. So, it has no crossover between individual particles.

PSO We see that the fitness shut down quickly then we go to a local minimum and this escape some area of research, but according the results above we see that the P_{error} doesn't exceed the 0.27 and the smallest min attend 0.23 so that mean that our choice off parameter was fairly valued and also the fitness is more and more closer to zero and so more desirable.

In **PSO**, Velocities of particles are typically limited to a maximum velocity V_{max}. We can change the maximum velocity as we want and after changing it many times during an execution for number of iteration between 50 and 300 we see that the larger V_{max} helps towards global exploration, and conversely a smaller Vmax promotes local exploitation.

The trade off between chosen parameter can make a different result, better or bad it's depend and that appear in the fitness and the *Perror* values. So the parameter must be carefully set since too little velocity means the swarm will converge on the first good solution that is found, while too much means the swarm may never converge on a solution.