Dr B R Ambedkar National
Institute of Technology , Jalandhar

Neural Network and Fuzzy Logic

Assignment

Particle Swarm Optimization

Submitted By:
Divya Kumari
18104026
ECE

Submitted To:
Dr Sandeep Verma

Introduction

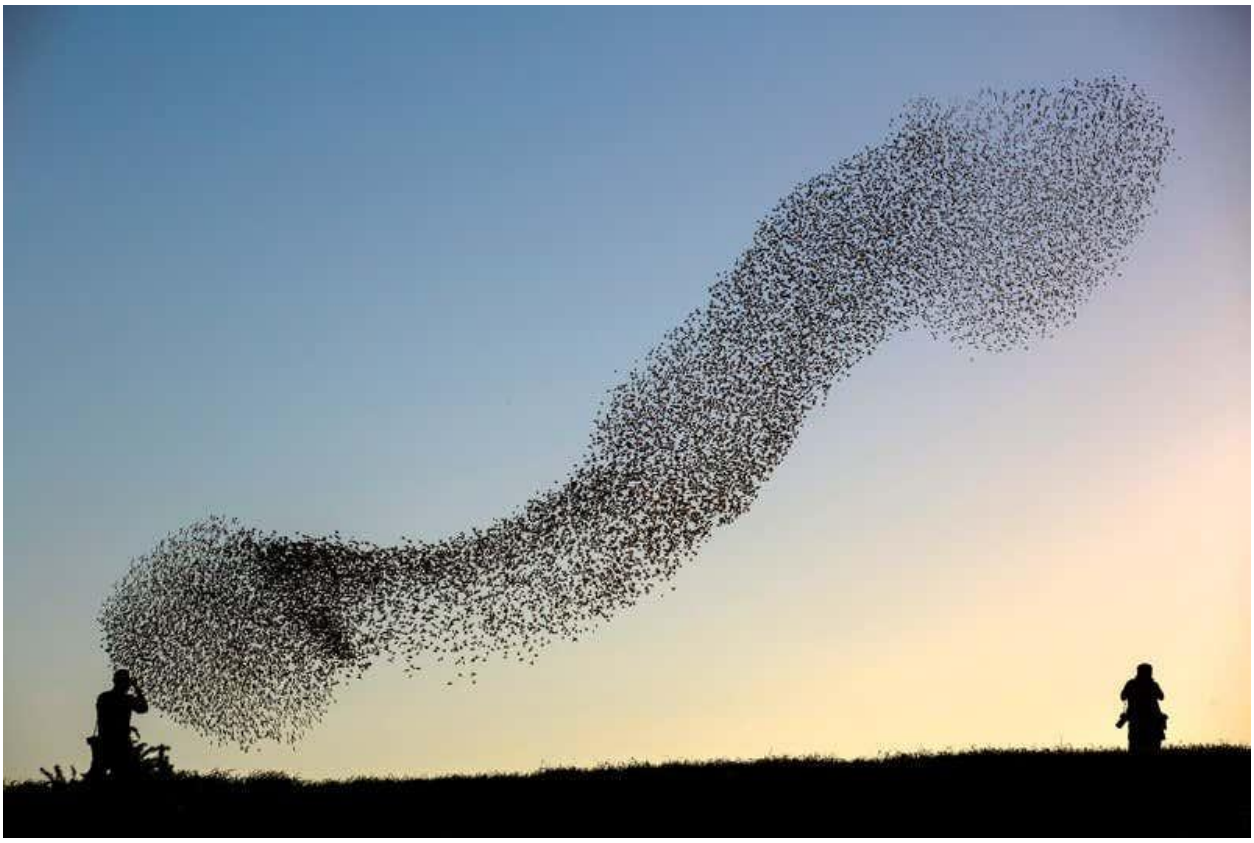
- In 1995, J.Kennedy and R.Eberhart developed a PSO algorithm inspired by social behaviour of organisms such as fish schooling and bird flocking.
- The PSO algorithm is based on the group of individuals (population based) to find the optimum solution.
- An individual in a swarm approaches the optimum through the previous experience, present velocity and experience of its neighbours.
- This algorithm involves taking a set of candidate solutions (particles) with random initial position, and the particles are set to move around the space to search for the best solution.

fig. fish schooling

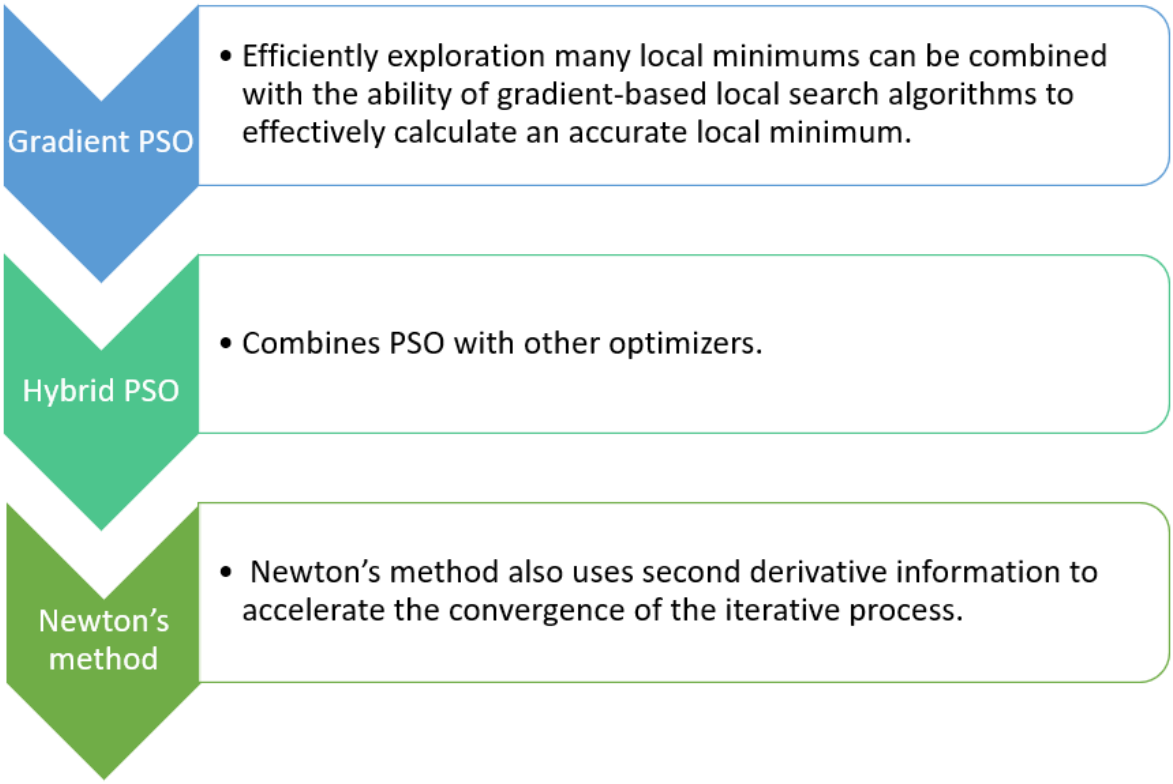
fig. bird flocking

Example

- Suppose there is a swarm (a group of birds). Now, all the birds are hungry and are searching for food. Now, in the locality of these birds, there is only one food particle. This food particle can be correlated with a resource. As we know, tasks are many, resources are limited. So this has become a similar condition as in a certain computation environment.
- Now, the birds don't know where the food particle is hidden or located. If every bird will try to find the food on its own, it may cause havoc and may consume a large amount of time.
- Thus on careful observation of this swarm, it was realized that though the birds don't know where the food particle is located, they do know their distance from it. Thus the best approach to finding that food particle is to follow the birds which are nearest to the food particle.
- This behavior of birds is simulated in the computation environment and the algorithm so designed is termed as **Particle Swarm Optimization Algorithm.**



Types of Particle Swarm Optimization



Procedure for implementation of PSO Optimization

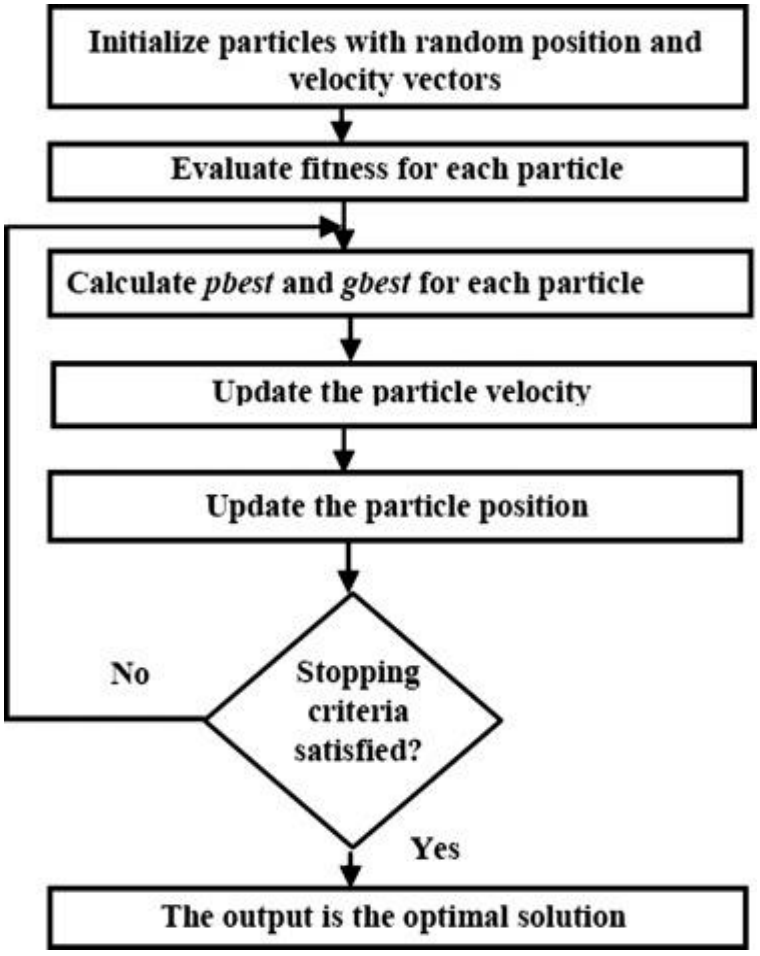


Fig. Flow Chart of PSO

Working

1. Initialize each particle with random velocity and position.
2. Calculate the objective value of all the particles.
3. Set the position and objective of each particle as p_i and p_{best} respectively.
4. Set the position and objective of the particle with the best fitness (least objective) as p_g and g_{best}, respectively.
5. Update Particles V_i^{k+1} and X_i^{k+1}
6. Update each particles p_{best} and p_i, that is, if the current p_{best} of the particle is better(less) than its p_{best}, then p_{best} and p_i are replaced with current objective value and position vector, respectively.
7. Update p_g and g_{best}, that is,if the current g_{best} of the whole swarm is fitter than g_{best}, then g_{best} and p_g are replaced with the current best objective and its corresponding position vector, respectively.
8. Steps 5-7 are repeated until stopping criterion (usually a prespecified number of iterations or a quality threshold for objective value) is reached.

Each particle updates its position based upon its own best position, global best position among particles and its previous velocity vector according to the following equations:

$$V_i^{k+1} = w.v_i^k + c_1.r_1(p_{best} - X_i^k) + c_2.r_2(g_{best} - X_i^k)$$

$X_i^{k+1} = X_i^k + a \,.\, V_i^{k+1}$

where,

V_i^{k+1}=Velocity of i^{th} particle at $(k+1)^{th}$ iteration

w = Inertia Weight of the particle

V_i^k=Velocity of i^{th} particle at k^{th} iteration

c_1, c_2=Constants having values between 0 and 2.5

r_1, r_2=Randomly numbers between 0 and 1

p_{best}=The best position of the i^{th} particle obtained based upon its own experience

g_{best} = Global best position of the particle in the populatio

X_i^{k+1}=The position of i^{th}particle at $(k+1)^{th}$ iteration

X_i^k =The position of i^{th}particle at k^{th}iteration

a=Constriction factor which will help to insure convergence

Optimally selected inertia weight w provides good balance between global and local explorations. So the equation for inertia weight is as follows:

w= w_{max} - $\frac{w_{max}-w_{min}}{iter_{max}}$. (iter)

where,

w_{max}=Inertia weight at the beginning of iterations

w_{min}= Inertia weight at the end of iterations

iter=Current iteration number

$iter_{max}$=Maximum number of iterations

Parameter selection in PSO

➢ **Swarm size**

- Too few particles---------the algorithm will trap in local optima.
- too many particles-------slow down the algorithm.
- So no exact rule for selection of swarm size, But when the dimension of problem increases, the swarm size should also be increased.

➢ **Maximum velocity**

- If it is too high, the particles move erratically and are swiftly attracted to global best (g_{best}) without enough exploration of search space and may exceed search space.
- if it is too small, the computational overhead increases and the algorithm may be unable to converge.
- Value for Vmax

$$V_{max} = \frac{X_{max} - X_{min}}{N_i}$$

N_i is the number of intervals in the dth dimension.

X_{max} and X_{min} are the maximum and minimum values of particles.

➢ **Inertia weight**

-There are different procedures for setting inertia weight:

- fixed inertia weight
- Linearly decreasing
- linearly increasing.
- non-linear, fuzzy adaptive, random....etc.

- Taking into account both simplicity and efficiency, linearly decreasing inertia Weight is the most appropriate method for setting inertia weight.

w_{max} = 0.9 and w_{min} = 0.4

Accepted values of w

➢ **Acceleration coefficients(c_1, c_2)**

- if too high — the particles move abruptly and fall in false optima.
- if too low — the particles move too slowly and the algorithm couldn't converge.
- If c_1 increases — it could increase attraction to p_{best} and decrease attraction to g_{best}.

- If c_2 increases — it could increase attraction to $g_{best}$and decrease attraction to p_{best}.

- So, setting c_1, c_2=2 for most of the problems.

Advantage

- Easy and simple to implement compare to other algorithm(i.e. ACO,GA etc....)
- Have few parameters to tune/adjust.
- Less programming complexity.
- Fast Convergence speed.
- Free Derivative algorithm and can be used to solve any type of optimization problems.

Disadvantage

- Can be difficult to define the initial design parameters.
- Can convergence prematurely and be trapped into local minimums especially with complex problems . Therefore many improved/hybrid PSO methods had been introduced.