Developing the Code: Executing Particle Swarm Optimization in SAS®

Anurag Srivastava Sangita Kumbharvadiya

ABSTRACT

Particle swarm optimization is a heuristic global optimization method which was given by James Kennedy and Russell C. Eberhart in 1995. (James Kennedy and Russell C. Eberhart) The purpose of this paper is to develop a code for particle swarm optimization in SAS 9.2.

INTRODUCTION

Particle Swarm Optimization, is combination of artificial life and social psychology as well as engineering and computer science, but differs from evolutionary computation methods like Artificial Neural Networks and Artificial Intelligence. Swarm intelligence is a branch of artificial intelligence that studies the collective behavior and emergent properties of complex, self-organized, decentralized systems with social structure (Konstantinos E. Parsopoulos and Michael N. Vrahatis) and Particle Swarm Optimization (PSO) is one of the algorithms to get best solution for a particular problem. Particle Swarm Optimization is a calculation method to optimize a problem by mathematical procedure by generating a sequence of improved approximate solutions for problems with given measure of quality. Particle Swarm Optimization is one of the latest population-based optimization methods, which does not use the filtering operation (such as crossover and/or mutation) and the members of the entire population are maintained through the search procedure.

Application of Particle Swarm Optimization

The applications of PSO are huge and vary from user to user such as best optimized solution in pattern recognition using cascading classifiers, image processing, unsupervised classification and image segmentation. PSO draws advantage in cases where other techniques are too expensive to implement or are too difficult. Also, PSO is a better choice of technique over others if there is a very high dimensionality involved in the data. Experiments also suggest that PSO is accurate and precise to fine tune the parameters for optimal performance e.g. PSO generates more compact clustering results than the regular K – means clustering technique (Abraham, Ajith; Grosan, Crina; Ramos, Vitorino; Swarm Intelligence in Data Mining) K – means algorithm depends on the initial choice of the cluster centers and Euclidean norm is sensitive to noise and outliers thereby affecting the K – means algorithm. On the contrary PSO clustering algorithm performs a globalized search in the entire solution space.

Advantages of Particle Swarm Optimization

Particle Swarm Optimization technique was widely used because of its advantages. First advantage is that there are fewer variables to change to be applied on any problems. Second advantage is PSO has better memory capacity as compared to GA because, every particle remembers its own previous best value as well as the neighborhood best value. Particle Swarm Optimization is based on artificial intelligence and so it can be applied on various different areas like medical science, machine fault identification, face recognition, human performance assessment, electric/hybrid vehicle battery pack state of change, evolving neural networks to solve problem etc. PSO occupies bigger optimization ability and it can be completed easily as well as the speed of researching is fast. PSO method is faster than other evolutionary algorithms on benchmark functions.

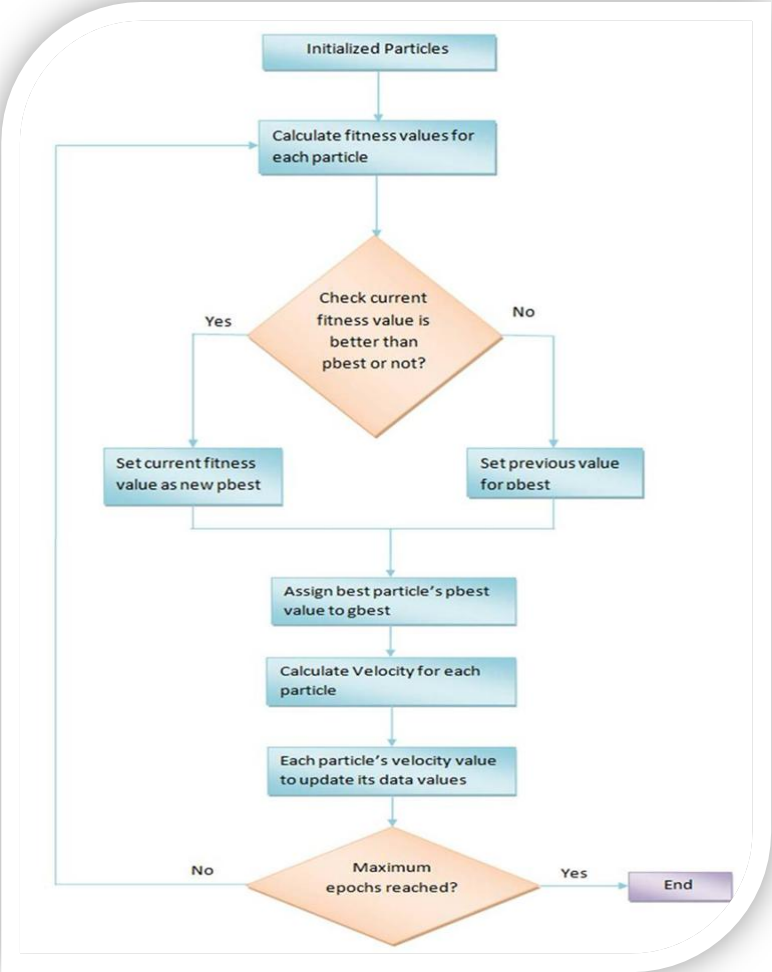
Overview of functions used

To run Particle Swarm Optimization in SAS, We need several functions and procedure as listed below.

1. Multidimensional Array: Multidimensional Array has row and column indices to identify an element. Multidimensional array is used when there is a need to retrieve a single value based on two selection criteria.

2. Iterative DO Loops: The iterative DO loop executes the statements between DO and END repetitively based on the value of an index variable.

3. GPlot Procedure: The GPLOT procedure plots the values of two or more variables on a set of coordinate axes: X and Y. The coordinates of each point on the plot correspond to two variable values in an observation of the input data set.



PSO Algorithm

The PSO Code

```
/* Set the graphics environment */
goptions reset=all border cback=white htitle=12pt;
data  pfinal;
iFUCNNO=-1;
iPOPSIZE = 40;
iDIMENSIONS = 2;
fINITWT = 0.9;
fMAXVEL = 10;
nMAXITER =2000;
fMaxPos = 100;
fERRCUTOFF =0.00001;
iLOCAL = 0;
sOutfile = "psoutfil.txt";
iXcoord1=1;
iYcoord1=1;
iZcoord1=1;
nIter=0;
array fPosi[2,40]iPOPSIZE1-iPOPSIZE40  iDIMENSIONS1
iDIMENSIONS40;
array fTempPos[2,40]iPOPSIZE1-iPOPSIZE40
   iDIMENSIONS1-iDIMENSIONS40;
array fVel[2,40]iPOPSIZE1-iPOPSIZE40
   iDIMENSIONS1-iDIMENSIONS40;
array fBestPos[2,40]iPOPSIZE1-iPOPSIZE40
   iDIMENSIONS1-iDIMENSIONS40;
array fDumVel[2]iDIMENSIONS1-iDIMENSIONS2;
array fErrVal[2]iPOPSIZE1-iPOPSIZE2;
array fPbestVal[40]iPOPSIZE1-iPOPSIZE40;
array iBetter[40]iPOPSIZE1-iPOPSIZE40;
array iNeighbor[40]iHOODINDEX1-iHOODINDEX40;
```

```
If iLOCAL > 0 Then iHOODSIZE = Int(iLOCAL / 2) * 2;
Else iHOODSIZE = iPOPSIZE;
iPopindex=1;
iDimindex=1;
Do i=1 to dim1(fPosi);
 Do j=1 to dim2(fPosi);
 fPosi(iPopindex,iDIMENSIONS)=rand('EXPO')*fMaxPos;
fBestPos(iPopindex,iDIMENSIONS)=fPosi(iPopindex,iDIM
ENSIONS);
fVel(iPopindex,iDIMENSIONS)=rand('EXPO')*fMAXVEL
;
 If rand('EXPO') > 0.5 Then
fPosi(iPopindex,iDIMENSIONS)=-
fPosi(iPopindex,iDIMENSIONS);
 If rand('EXPO')> 0.5 Then
fVel(iPopindex,iDIMENSIONS)=-
fVel(iPopindex,iDIMENSIONS);
          Output;
 end;
         output;
end ;
fInerWt = ((fINITWT - 0.4) * (nMAXITER - nIter) /
nMAXITER) + 0.4;
do i=1 to dim1(fPosi);
do j=1 to dim2(fPosi);
fDumVel(iDimindex) = fVel(iPopindex, iDimindex);
         output;
 end;
 iBetter(iPopindex) = 0;
 fErrVal(iPopindex) = 0;/*sphare function started*/
 do j = 1 To dim(fPosi);
     fErrValDim = (fPosi(iPopindex, iDimindex)) ** 2;
      fErrVal(iPopindex) = fErrVal(iPopindex) +
fErrValDim;
          output;
 end;/*end of the sphare function*/
if nIter = 1 Then fPbestVal(iPopindex) =
fErrVal(iPopindex);
 iGbest = 1;
 If fErrVal(iPopindex) < fPbestVal(iPopindex) Then
fPbestVal(iPopindex) =
fErrVal(iPopindex);
 do j=1 to dim(fPosi);
    fBestPos(iPopindex, iDimindex) = fPosi(iPopindex,
iDimindex);
                output;
 end;
 If fPbestVal(iPopindex) < fPbestVal(iGbest) Then iGbest =
iPopindex;
end;
 do i=1 to dim(fPosi); /*update velocity, position, graph
position*/ If iLOCAL > 0 Then /*Does neighborhood
calculation of iLbest*/
   do k=1 to dim(iNeighbor);
   iHOODINDEX=1;
  iNeighbor(iHOODINDEX) = iPopindex - (iHOODSIZE /
2) + iHOODINDEX;
             If iNeighbor(iHOODINDEX) < 1 Then
iNeighbor(iHOODINDEX) = iPOPSIZE +
iNeighbor(iHOODINDEX); /* Now wrap the ends of the
array*/
             If iNeighbor(iHOODINDEX) > iPOPSIZE Then
iNeighbor(iHOODINDEX) = iNeighbor(iHOODINDEX) -
iPOPSIZE;
  If iHOODINDEX = 0 Then iLbest = iNeighbor(0);/*Start
with iNeighbor(0) as iLbest and try to beat it*/
 If fPbestVal(iNeighbor(iHOODINDEX)) <
fPbestVal(iLbest) Then iLbest =
iNeighbor(iHOODINDEX);
          Output;
          end;
```

```
If iLOCAL = 0 Then iLbest = iGbest;/*Update velocity vector for
particles*/
 do j= 1 To dim(fPosi);       /*fInerWt below*/
            fVel(iPopindex, iDimindex) = (0.5 + (rand('EXPO') / 2)) *
fVel(iPopindex,  iDimindex) + 2 * rand('EXPO') *
(fBestPos(iPopindex, iDimindex) - fPosi(iPopindex,
iDimindex)) + 2 * rand('EXPO') * (fBestPos(iLbest, iDimindex) -
fPosi(iPopindex, iDimindex));
 If fVel(iPopindex, iDimindex) > fMAXVEL Then fVel(iPopindex,
iDimindex) =
fMAXVEL;
            /*Else fVel(iPopindex, iDimindex) <= (-fMAXVEL) Then
fVel(iPopindex,
iDimindex) = (-fMAXVEL);*/
            Output;
            end;
            If iBetter(iPopindex) = 1 Then
             do j = 1 To dim(fPosi);
                fVel(iPopindex, iDimindex) = fDumVel(iDimindex);
                output;
             end;
output;
end;
/*Graphics Loop: Graphically plot updated positions*/
title1 'Particle Swarm Optimization'  j=c 'and Size of Each Particle';
  /* Define the symbol shape */
symbol1 height=2.1 value=dot;
 /* Define axis characteristics */
axis1 label=('Size (in Angstroms)') minor=none;
axis2 label=('Particles') minor=none;
/* Define legend characteristics */
legend1 label=(position=(top left)'Temperature' j=l '(Celsius)');
shape=symbol(4,2);
 PROC GPLOT DATA=pfinal;
iXcoord1=fPosi(iPopndex,1)/fMaxPos;
iYcoord1=fPosi(iPopndex,2)/fMaxPos;
plot iXcoord1*iYcoord1=iZcoord1 / haxis=axis1  vaxis=axis2
                  legend=legend1;
run;
```

References

Abraham A., Grosan C., and Ramos V., Swarm Intelligence in Data Mining, Springer Publication.

Cody, Ronald P., and Raymond Pass. 1995. SAS® Programming by Example, Cary, NC: SAS Institute Inc.

John McCullock, *import Particle Swarm Optimization*, **viewed on 18th November 2012, http://www.mnemstudio.org/particle-swarm-introduction.htm.**

Kennedy J., Eberhart R. and Shi Y., Swarm Intelligence, Morgan Kaufmann Publisher.

McCaffrey J. , *Artificial Intelligent – Particle Swarm optimization*, **viewed 11 November 2012, http://msdn.microsoft.com/en-us/magazine/hh335067.aspx.**

Parsopoulos K. and Vrahatis M., Particle Swarm Optimization and Intelligence: Advances and Applications, information science reference publisher.

The Pennsylvania State University, *STAT 481 - Intermediate SAS*, **viewed on 18 November 2012, https://onlinecourses.science.psu.edu/stat481/node/41.**

Website: http://www.sas.com viewed on 30 October 2012.

Xiaohui Hu, *Particle Swarm Optimization*, **viewed 11 November 2012, http://www.swarmintelligence.org/tutorials.php.**