E.S. Gopi

Algorithm Collections for Digital Signal Processing Applications using Matlab

Springer

Algorithm Collections for Digital Signal Processing Applications Using Matlab

E.S. Gopi
National Institute of Technology, Tiruchi, India

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4020-6409-8 (HB)
ISBN 978-1-4020-6410-4 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

Algorithm Collections for Digital Signal Processing Applications Using Matlab

This book is dedicated to

my Wife G.Viji

and my Son V.G.Vasig

Preface

The Algorithms such as SVD, Eigen decomposition, Gaussian Mixture Model, PSO, Ant Colony etc. are scattered in different fields. There is the need to collect all such algorithms for quick reference. Also there is the need to view such algorithms in application point of view. This Book attempts to satisfy the above requirement. Also the algorithms are made clear using MATLAB programs. This book will be useful for the Beginners Research scholars and Students who are doing research work on practical applications of Digital Signal Processing using MATLAB.

Acknowledgments

I am extremely happy to express my thanks to the Director Dr M.Chidambaram, National Institute of Technology Trichy India for his support. I would also like to thank Dr B.Venkatramani, Head of the Electronics and Communication Engineering Department, National Institute of Technology Trichy India and Dr K.M.M. Prabhu, Professor of the Electrical Engineering Department, Indian Institute of Technology Madras India for their valuable suggestions. Last but not least I would like to thank those who directly or indirectly involved in bringing up this book sucessfully. Special thanks to my family members father Mr E.Sankara subbu, mother Mrs E.S.Meena, Sisters R.Priyaravi, M.Sathyamathi, E.S.Abinaya and Brother E.S.Anukeerthi.

Thanks

E.S.Gopi

Contents

Chapter 1

ARTIFICIAL INTELLIGENCE

Algorithm Collections

1. PARTICLE SWARM ALGORITHM

Consider the two swarms flying in the sky, trying to reach the particular destination. Swarms based on their individual experience choose the proper path to reach the particular destination. Apart from their individual decisions, decisions about the optimal path are taken based on their neighbor's decision and hence they are able to reach their destination faster. The mathematical model for the above mentioned behavior of the swarm is being used in the optimization technique as the Particle Swarm Optimization Algorithm (PSO).

For example, let us consider the two variables 'x' and 'y' as the two swarms. They are flying in the sky to reach the particular destination (i.e.) they continuously change their values to minimize the function $(x-10)^2+(y-5)^2$. Final value for 'x' and 'y' are 10.1165 and 5 respectively after 100 iterations.

The Figure 1-1 gives the closed look of how the values of x and y are changing along with the function value to be minimized. The minimization function value reached almost zero within 35 iterations. Figure 1-2 shows the zoomed version to show how the position of x and y are varying until they reach the steady state.

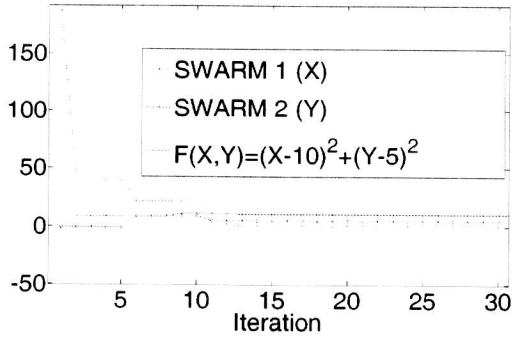

Figure 1-1. PSO Example zoomed version

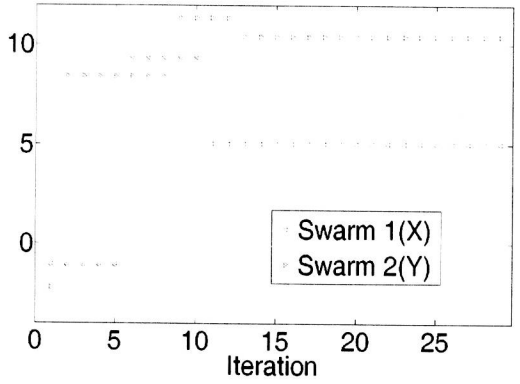

Figure 1-2. PSO Example

1.1 How are the Values of 'x and y' are Updated in Every Iteration?

The vector representation for updating the values for x and y is given in Figure 1-3. Let the position of the swarms be at 'a' and 'b' respectively as shown in the figure. Both are trying to reach the position 'e'. Let 'a' decides to move towards 'c' and 'b' decides to move towards 'd'.

The distance between the position 'c' and 'e' is greater than the distance between 'd' and 'e'. so based on the neighbor's decision position 'd' is treated as the common position decided by both 'a' and 'b'. (ie) the position 'c' is the individual decision taken by 'a', position 'd' is the individual decision taken by 'b' and the position 'd' is the common position decided by both 'a' and 'b'.

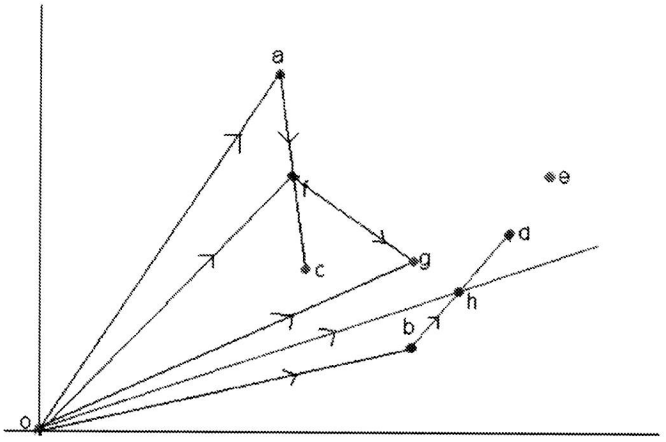

Figure 1-3. Vector Representation of PSO Algorithm

'a' based on the above knowledge, finally decides to move towards the position 'g' as the linear combination of 'oa' , 'ac' and 'ad'. [As 'd' is the common position decided].The linear combination of 'oa' and scaled 'ac' (ie) 'af' is the vector 'of'. The vector 'of' combined with vector 'fg' (ie) scaled version of 'ad' to get 'og' and hence final position decided by 'a' is 'g'.

Similarly, 'b' decides the position 'h' as the final position. It is the linear combination of 'ob' and 'bh'(ie) scaled version of 'bd'. Note as 'd' is the common position decided by 'a' and 'b', the final position is decided by linear combinations of two vectors alone.

Thus finally the swarms 'a' and 'b' moves towards the position 'g' and 'h' respectively for reaching the final destination position 'e'. The swarm 'a' and 'b' randomly select scaling value for linear combination. Note that 'oa' and 'ob' are scaled with 1 (ie) actual values are used without scaling. Thus the decision of the swarm 'a' to reach 'e' is decided by its own intuition along with its neighbor's intuition.

Now let us consider three swarms (A,B,C) are trying to reach the particular destination point 'D'. A decides A', B decides B' and C decides C' as the next position. Let the distance between the B' and D is less compared with A'D and C' and hence, B' is treated as the global decision point to reach the destination faster.

Thus the final decision taken by A is to move to the point, which is the linear combination of OA, AA' and AB'. Similarly the final decision taken

by B is to move the point which is the linear combination of OB, BB'. The final decision taken by C is to move the point which is the linear combination of OC, CC' and CB'.

1.2 PSO Algorithm to Maximize the Function F (X, Y, Z)

1. Initialize the values for initial position a, b, c, d, e
2. Initialize the next positions decided by the individual swarms as a', b', c' d' and e'
3. Global decision regarding the next position is computed as follows. Compute f (a', b, c, d, e), f (a, b', c, d, e), f (a, b, c', d, e), f (a, b, c, d', e) and f (a, b, c, d, e'). Find minimum among the computed values. If f (a', b, c, d, e) is minimum among all, the global position decided regarding the next position is a'. Similarly If f (a, b', c, d, e) is minimum among all, b' is decided as the global position regarding the next position to be shifted and so on. Let the selected global position is represented ad 'global'
4. Next value for a is computed as the linear combination of 'a' , (a'-a) and (global-a) (ie)

- nexta = a+ C1 * RAND * (a' –a) + C2 * RAND * (global –a)
- nextb = b+ C1 * RAND * (b' –b) + C2 * RAND * (global –b)
- nextc = c+ C1 * RAND * (c' –c) + C2 * RAND * (global –c)
- nextd = d+ C1 * RAND * (d' –d) + C2 * RAND * (global –d)
- nexte = e+ C1 * RAND * (e' –e) + C2 * RAND * (global –e)

5. Change the current value for a, b, c, d and e as nexta, nextb, nextc, nextd and nexte
6. If f (nexta, b, c, d, e) is less than f (a', b, c, d, e) then update the value for a' as nexta, otherwise a' is not changed.

   If f (a, nextb, c, d, e) is less than f (a, b', c, d, e) then update the value for b' as nextb, otherwise b' is not changed

   If f (a, b, nextc, d, e) is less than f (a, b, c', d, e) then update the value for c' as nextc, otherwise c' is not changed

If f (a, b, c, nextd, e) is less than f (a, b, c, d', e) then update the value for d' as nextd, otherwise d' is not changed

If f (a, b, c, d, nexte) is less than f (a, b, c, d, e') then update the value for e' as nexte, otherwise e' is not changed

7. Repeat the steps 3 to 6 for much iteration to reach the final decision.

The values for 'c1','c2' are decided based on the weightage given to individual decision and global decision respectively.

Let $\Delta a(t)$ is the change in the value for updating the value for 'a' in tth iteration, then nexta at (t+1)th iteration can be computed using the following formula. This is considered as the velocity for updating the position of the swarm in every iteration.

$$\text{nexta}\ (t+1) = a\ (t) + \Delta a(t+1)$$

where

$$\Delta a(t+1) = c1 * \text{rand} * (a' - a\ ) + c2 * \text{rand} * (\text{global} - a\ ) + w(t) * \Delta a(t)$$

'w (t)' is the weight at t^{th} iteration. The value for 'w' is adjusted at every iteration as given below, where 'iter' is total number of iteration used.

$$w(t+1)=w(t)-t*w(t)/(\text{iter}).$$

Decision taken in the previous iteration is also used for deciding the next position to be shifted by the swarm. But as iteration increases, the contribution of the previous decision is decreases and finally reaches zero in the final iteration.