Appendix A: The ANN-PSO codes (nn.pso.m)

clc

tic

close all

clear all

rng default

filename = 'datafile.xlsx';

sheetname1 = 'Sheet1';

sheetname2 = 'Sheet2';

input = xlsread(filename,sheetname1,'A1:Z10000');

target = xlsread(filename,sheetname2,'A1:Z10000');

inputs=input';

targets=target';

m=length(inputs(:,1));

o=length(targets(:,1));

n=9;

net=feedforwardnet(n);

net=configure(net,inputs,targets);

kk=m\*n+n+n+o;

for j=1:kk

LB(1,j)=-1.5;

UB(1,j)=1.5;

end

pop=200;

for i=1:pop

for j=1:kk

xx(i,j)=LB(1,j)+rand\*(UB(1,j)-LB(1,j));

end

end

maxrun=1;

for run=1:maxrun

fun=@(x) myfunc(x,n,m,o,net,inputs,targets);

x0=xx;

% pso initialization----------------------------------------------start

x=x0; % initial population

v=0.1\*x0; % initial velocity

for i=1:pop

f0(i,1)=fun(x0(i,:));

end

[fmin0,index0]=min(f0);

pbest=x0; % initial pbest

gbest=x0(index0,:); % initial gbest

% pso initialization------------------------------------------------end

% pso algorithm---------------------------------------------------start

c1=1.5; c2=2.25;

ite=1; maxite=1000; tolerance=1;

while ite<=maxite && tolerance>10^-8

w=0.1+rand\*0.4;

% pso velocity updates

for i=1:pop

for j=1:kk

v(i,j)=w\*v(i,j)+c1\*rand\*(pbest(i,j)-x(i,j))+c2\*rand\*(gbest(1,j)-x(i,j));

end

end

% pso position update

for i=1:pop

for j=1:kk

x(i,j)=x(i,j)+v(i,j);

end

end

% handling boundary violations

for i=1:pop

for j=1:kk

if x(i,j)<LB(j)

x(i,j)=LB(j);

elseif x(i,j)>UB(j)

x(i,j)=UB(j);

end

end

end

% evaluating fitness

for i=1:pop

f(i,1)=fun(x(i,:));

end

% updating pbest and fitness

for i=1:pop

if f(i,1)<f0(i,1)

pbest(i,:)=x(i,:);

f0(i,1)=f(i,1);

end

end

[fmin,index]=min(f0); % finding out the best particle

ffmin(ite,run)=fmin; % storing best fitness

ffite(run)=ite; % storing iteration count

% updating gbest and best fitness

if fmin<fmin0

gbest=pbest(index,:);

fmin0=fmin;

end

% calculating tolerance

if ite>100;

tolerance=abs(ffmin(ite-100,run)-fmin0);

end

% displaying iterative results

if ite==1

disp(sprintf('Iteration Best particle Objective fun'));

end

disp(sprintf('%8g %8g %8.4f',ite,index,fmin0));

ite=ite+1;

end

% pso algorithm-----------------------------------------------------end

xo=gbest;

fval=fun(xo);

xbest(run,:)=xo;

ybest(run,1)=fun(xo);

disp(sprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'));

disp(sprintf(' RUN fval ObFuVa'));

disp(sprintf('%6g %6g %8.4f %8.4f',run,fval,ybest(run,1)));

end

toc

% Final neural network model

disp('Final nn model is net\_f')

net\_f = feedforwardnet(n);

net\_f=configure(net\_f,inputs,targets);

[a b]=min(ybest);

xo=xbest(b,:);

k=0;

for i=1:n

for j=1:m

k=k+1;

xi(i,j)=xo(k);

end

end

for i=1:n

k=k+1;

%%

%

% for x = 1:10

% disp(x)

% end

%

xl(i)=xo(k);

xb1(i,1)=xo(k+n);

end

for i=1:o

k=k+1;

xb2(i,1)=xo(k);

end

net\_f.iw{1,1}=xi;

net\_f.lw{2,1}=xl;

net\_f.b{1,1}=xb1;

net\_f.b{2,1}=xb2;

%Calculation of MSE

err=sum((net\_f(inputs)-targets).^2)/length(net\_f(inputs))

%Regression plot

plotregression(targets,net\_f(inputs))

disp('Trained ANN net\_f is ready for the use');

%Trained ANN net\_f is ready for the use

Appendix B: The ANN-PSO codes (myfunc.m)

