BPtrain.m

%网络的训练

clc;clear;

data=xlsread('attachment1.xls')

inputDim=5;

hiddenNN=5;

outputDim=1;

nn=104;

xx=data(:,[10,4:6,9]);

yy=data(:,[11,13]);

X=xx;

Y=yy(:,1).\*yy(:,2);

ww=Y/10000;

hist(ww,10)

[output\_test,test\_simu\_1,test\_simu]=GeneticAlgorithm(X,Y,inputDim,hiddenNN,outputDim,nn)

BPeval.m

%网络的预测

load BPnet

tic

Xmax = [450;200;5;200;2.1];

Xmin = [250;10;0.5;10;0.3];

Vmax = 1\*ones(dim,1);

Vmin = -1\*ones(dim,1);

sOfP = 500;

dim = 5;

generation =100;

mm\_1 = 0.8;

mm\_2 = 0.5;

mm\_3 = 0.5;

k=0.25;

for i=1:dim

for j=1:sOfP

pop\_x(i,j) = Xmin(i)+(Xmax(i) - Xmin(i))\*rand;

pop\_v(i,j) = Vmin(i)+(Vmax(i) - Vmin(i))\*rand;

end

end

gbest = pop\_x;

for j=1:sOfP

geneticBest(j) = net(pop\_x(:,j));

end

zbest = pop\_x(:,1);

Hbest = geneticBest(1);

for j=1:sOfP

if geneticBest(j) > Hbest

zbest = pop\_x(:,j);

Hbest=geneticBest(j);

end

end

iter = 1;

record = zeros(generation, 1);

while iter <= generation

for j=1:sOfP

pop\_v(:,j)= mm\_1 \* pop\_v(:,j) + mm\_2\*rand\*(gbest(:,j)-pop\_x(:,j))+mm\_3\*rand\*(zbest-pop\_x(:,j));

for i=1:dim

if pop\_v(i,j) > Vmax(i)

pop\_v(i,j) = Vmax(i);

end

if pop\_v(i,j) < Vmin(i)

pop\_v(i,j) = Vmin(i);

end

end

pop\_x(:,j) = pop\_x(:,j) + pop\_v(:,j);

for i=1:dim

if pop\_x(i,j) > Xmax(i)

pop\_x(i,j) = Xmax(i);

end

if pop\_x(i,j) < Xmin(i)

pop\_x(i,j) = Xmin(i);

end

end

if rand > 0.87

i=ceil(dim\*rand);

pop\_x(i,j)=Xmin(i) + (Xmax(i) - Xmin(i)) \* rand;

end

pp(j) = net(pop\_x(:,j));

if pp(j) >geneticBest(j)

gbest(:,j) = pop\_x(:,j);

geneticBest(j) = pp(j);

end

if geneticBest(j) >Hbest

zbest = gbest(:,j);

Hbest=geneticBest(j);

end

end

record(iter) = Hbest;

iter = iter+1;

end

record=record.\*k;

m(generation )=k\*Hbest;

for k=1:generation -1

j=generation -k;

m(j)=m(j+1)/(0.1\*rand(1)+1)

end

GeneticAlgorithm.m

%基于遗传算法的BP神经网络模型

function [output\_test,test\_simu\_1,test\_simu]=GeneticAlgorithm(input,output,inputDim,hiddenNN,outputDim,nn)

input\_train=input(1:nn,:)';

input\_test=input(nn+1:end,:)';

output\_train=output(1:nn,:)';

output\_test=output(nn+1:end,:)';

[inputn,inputps]=mapminmax(input\_train);

[outputn,outputps]=mapminmax(output\_train);

net=newff(inputn,outputn,hiddenNN);

net.trainParam.epochs=100;

net.trainParam.lr=0.1;

net.trainParam.goal=0.00001;

[net,pp3]=train(net,inputn,outputn);

inputn\_test\_1=mapminmax('apply',input\_test,inputps);

an\_1=sim(net,inputn\_test\_1);

test\_simu\_1=mapminmax('reverse',an\_1,outputps);

error\_1=test\_simu\_1-output\_test;

maxgen=20;

sOfP=10;

pcross=0.2;

pmutation=0.1;

nOfS=inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim+outputDim;

lenchrom=ones(1,nOfS);

bound=[-3\*ones(nOfS,1) 3\*ones(nOfS,1)];

samples=struct('fitness',zeros(1,sOfP), 'chrom',[]);

avgfitness=[];

bestfitness=[];

Cbest=[];

for i=1:sOfP

samples.chrom(i,:)=Code(lenchrom,bound); %

x=samples.chrom(i,:);

samples.fitness(i)=fitnessfunc(x,inputDim,hiddenNN,outputDim,net,inputn,outputn);

end

FitRecord=[];

[bestfitness, bestindex]=min(samples.fitness);

Cbest=samples.chrom(bestindex,:);

avgfitness=sum(samples.fitness)/sOfP;

trace=[avgfitness bestfitness];

for i=1:maxgen

samples=Select(samples,sOfP);

avgfitness=sum(samples.fitness)/sOfP;

samples.chrom=Cross(pcross,lenchrom,samples.chrom,sOfP);

samples.chrom=Mutation(pmutation,lenchrom,samples.chrom,sOfP,i,maxgen,bound);

for j=1:sOfP

x=samples.chrom(j,:);

samples.fitness(j)=fitnessfunc(x,inputDim,hiddenNN,outputDim,net,inputn,outputn);

end

[newbestfitness,newbestindex]=min(samples.fitness);

[worestfitness,worestindex]=max(samples.fitness);

if bestfitness>newbestfitness

bestfitness=newbestfitness;

Cbest=samples.chrom(newbestindex,:);

end

samples.chrom(worestindex,:)=Cbest;

samples.fitness(worestindex)=bestfitness;

avgfitness=sum(samples.fitness)/sOfP;

trace=[trace;avgfitness bestfitness];

FitRecord=[FitRecord;samples.fitness];

end

w1=x(1:inputDim\*hiddenNN);

B1=x(inputDim\*hiddenNN+1:inputDim\*hiddenNN+hiddenNN);

w2=x(inputDim\*hiddenNN+hiddenNN+1:inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim);

B2=x(inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim+1:inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim+outputDim);

net.iw{1,1}=reshape(w1,hiddenNN,inputDim);

net.lw{2,1}=reshape(w2,outputDim,hiddenNN);

net.b{1}=reshape(B1,hiddenNN,1);

net.b{2}=reshape(B2,outputDim,1);

net.trainParam.epochs=100;

net.trainParam.lr=0.1;

net.trainParam.goal=0.00001;

[net,pp3]=train(net,inputn,outputn);

inputn\_test=mapminmax('apply',input\_test,inputps);

an=sim(net,inputn\_test);

test\_simu=mapminmax('reverse',an,outputps);

error=test\_simu-output\_test;

save('BPnet','net')

end

fitnessfunc.m

%适应度函数

function error =fitnessfunc(x,inputDim,hiddenNN,outputDim,net,inputn,outputn)

w1=x(1:inputDim\*hiddenNN);

B1=x(inputDim\*hiddenNN+1:inputDim\*hiddenNN+hiddenNN);

w2=x(inputDim\*hiddenNN+hiddenNN+1:inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim);

B2=x(inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim+1:inputDim\*hiddenNN+hiddenNN+hiddenNN\*outputDim+outputDim);

net.trainParam.epochs=20;

net.trainParam.lr=0.1;

net.trainParam.goal=0.00001;

net.trainParam.show=100;

net.trainParam.showWindow=0;

net.iw{1,1}=reshape(w1,hiddenNN,inputDim);

net.lw{2,1}=reshape(w2,outputDim,hiddenNN);

net.b{1}=reshape(B1,hiddenNN,1);

net.b{2}=reshape(B2,outputDim,1);

net=train(net,inputn,outputn);

an=sim(net,inputn);

error=sum(abs(an-outputn));