%小波分析的matlab文件

%clear all;

clc

%导入参数数据

%a是\*.xlsx经过拉依达准则处理后的数据集

a=xlsread('B2.xlsx');

data=a(:, 8);

figure,plot(data);

xlabel('(s)');

ylabel('变量y');

title('原始数据');

%为角度赋初始值

a=pi/8;

b=pi/8;

%为重构滤波器h0(n)冲激响应赋值

h0=cos(a)\*cos(b);

h1=sin(a)\*cos(b);

h2=-sin(a)\*sin(b);

h3=cos(a)\*sin(b);

construct=[h0,h1,h2,h3];

frequency=4;

%设定滤波器长度

decompose=construct(end:-1:1); %确定h0(-n),分解滤波器

%导入参数数据

a=xlsread('A2.xlsx');

figure,plot(data);

xlabel('时间(s)');

ylabel('变量y');

title('原始数据');

figure(1);

plot(data);

title('原数据');

%h0(n)性质校验

check1=sum(decompose);

check2=norm(decompose);

frequency=conv(data,decompose);%卷积

frequency\_down=dyaddown(frequency);%信号低频细节

figure(2);

plot(frequency\_down);

title('小波分析的低频系数');

%确定阈值并去噪的matlab代码

clc;

clear;

% 获取噪声信号

data = xlsread('E.xlsx');

[m, n] = size(data);

leleccum = data(:, 8);

indx = 1:m;

x = leleccum(indx);

lev=5;

wname='db3';

[c,l]=wavedec(x,lev,wname);

sigma=wnoisest(c,l,1);

alpha=2;

thr1=wbmpen(c,l,sigma,alpha)

[thr2,nkeep]=wdcbm(c,l,alpha)

xd1=wdencmp('gbl',c,l,wname,lev,thr1,'s',1);

[xd2,cxd,lxd,perf0,perfl2]=wdencmp('lvd',c,l,wname,lev,thr2,'h');

[thr,sorh,keepapp]=ddencmp('den','wv',x)

xd3=wdencmp('gbl',c,l,wname,lev,thr,'s',1);

subplot(411);plot(x);title('原始信号','fontsize',12);

subplot(412);plot(xd1);title('使用penalty阈值降噪后信号','fontsize',12);

subplot(413);plot(xd2);title('使用Birge-Massart阈值降噪后信号','fontsize',12);

subplot(414);plot(xd3);title('使用缺省阈值降噪后信号','fontsize',12);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%以变异系数百分比为权重建立故障判断半径的matlab代码

clear;

clc;

%A1 导致故障的样本集，A2 导致正常的样本集，两者参数项都取正常的参数,不存在则为0

data\_gz = xlsread('C1.xlsx');

[a, b] = size(data\_gz);

data\_zc = xlsread('C2.xlsx');

[c, d] = size(data\_zc);

%各特征的权重w等于其变异系数占总变异系数的概率

%变异系数百分比由SPSS求出

w = [0.090918222,0.081980853,0.096283568,0.089253191,0.098231678,...

0.006297036,0.002541642,0.001723954,0.004255227,0.015988422,...

0.098910481,0.063557656,0.008649716,0.082026046,0.082026046,...

0.061227922,0.061253648,0.003141365,0.008115261,0.003136224,...

0.040481894];

sum = 1;

%idx表示五个主要参数

idx = [3, 5 , 6, 7 , 8]; %主要参数 2, 4, 5, 6, 7

%求解故障判定距离

for j = 2 : b

%求解欧氏距离

if j > idx(length(idx))

break;

end

if j == 3 || j == 5 || j == 6 || j == 7 || j == 8

x1 = mean(data\_gz(:,j));

x2 = mean(data\_zc(:, j));

dis = sqrt(abs(x2 \* x2 - x1 \* x1));

sum = sum \* (dis ^ w(j-1));

%end

end

sum

%建立改进粒子群的BP神经网络预测模型的代码

%% 清空环境

clc

clear all;

close all;

%读取数据

%load data input\_train input\_test output\_train output\_test

%初始隐层神经元个数,需要以输入层/输出层神经元个数为基础再试错而确定

hiddennum=10;

%训练数据(3个特征，27组数据)

input1=[356 356 356 360 356 356 356 356 356 356 360 360 ...

356 360 360 356 356 356 360 360 360 360 356 360 ...

360 356 360];

input2=[356 356 360 356 356 356 356 356 356 360 360 356 ...

360 360 356 356 356 360 360 360 360 356 360 360 ...

356 360 360];

input3=[356 360 356 356 356 356 356 356 360 360 356 360 ...

360 356 356 356 360 360 360 360 356 360 360 356 ...

360 360 356];

output1=[360 356 356 356 356 356 356 360 360 356 360 360 ...

356 356 356 356 360 360 360 356 360 360 356 360 ...

360 356 356];

[inputn,inputps]=mapminmax([input1;input2;input3]); %输入数据归一化

[outputn,outputps]=mapminmax(output1); %输出数据归一化

%输入层/输出层神经元个数

inputnum=size([input1;input2;input3], 1);

outputnum=size(output1, 1);

%测试数据

inputt1=[360 356 356 356 356 360 360 356 356 356 360 360 ...

356 356 360 356 360 356 356 360 360 360 360 360 ...

356 360 360];

inputt2=[356 356 356 356 360 360 356 356 356 360 360 356 ...

356 360 356 360 356 356 360 360 360 360 360 356 ...

360 360 360];

inputt3=[356 356 356 360 360 356 356 356 360 360 356 356 ...

360 356 360 356 356 360 360 360 360 360 356 360 ...

360 360 360];

outputt1=[356 356 360 360 356 356 356 360 360 356 356 360 ...

356 360 356 356 360 360 360 360 360 356 360 360 ...

360 360 360];

inputtn=mapminmax('apply',[inputt1;inputt2;inputt3],inputps); %输入数据归一化

%预测数据

in1=[360 360 360 360 360 356];

in2=[360 360 360 360 356 356];

in3=[360 360 360 356 356 356];

%正确推断应是 [360 360 356 356 356 356]

inn=mapminmax('apply',[in1;in2;in3],inputps); %输入数据归一化

%构建网络

net=newff(inputn,outputn,hiddennum);

% 参数初始化

%粒子群算法中的两个参数

c1 = 1.49445;

c2 = 1.49445;

maxgen=65; % 进化次数<----------------------------------------

sizepop=20; %种群规模<---------------------------------------遗传算法优化

wmax=0.9;

wmin=0.4;

Vmax=1;

Vmin=-1;

popmax=8;

popmin=-8;

Dim=inputnum\*hiddennum+hiddennum+hiddennum\*outputnum+outputnum;

%% 产生初始粒子和速度

for i=1:sizepop

%随机产生一个种群

pop(i,:)=5\*rands(1,Dim); %初始种群

vov(i,:)=rands(1,Dim); %初始化速度

%计算适应度

fitness(i)=fun(pop(i,:),inputnum,hiddennum,outputnum,net,inputn,outputn,inputps,outputps); %染色体的适应度

end

% 个体极值和群体极值

[bestfitness bestindex]=min(fitness);

zbest=pop(bestindex,:); %全局最佳

gbest=pop; %个体最佳

fitnessgbest=fitness; %个体最佳适应度值

fitnesszbest=bestfitness; %全局最佳适应度值

%% 迭代寻优

for i=1:maxgen

%粒子位置和速度更新

for j=1:sizepop

w=wmax-(wmax-wmin)\*j/maxgen;

%速度更新

%length(gbest(j,:));

%length(pop(j,1:Dim))

vov(j,:) = w\*vov(j,:) + c1\*rand\*(gbest(j,:) - pop(j,1:Dim)) + c2\*rand\*(zbest - pop(j,1:Dim));

vov(j,find(vov(j,:)>Vmax))=Vmax;

vov(j,find(vov(j,:)<Vmin))=Vmin;

%种群更新

pop(j,1:Dim)=pop(j,1:Dim)+0.5\*vov(j,:);

pop(j,find(pop(j,1:Dim)>popmax))=popmax;

pop(j,find(pop(j,1:Dim)<popmin))=popmin;

%引入变异算子，重新初始化粒子

if rand>0.9

k=ceil(21\*rand);

pop(j,k)=rand;

end

%新粒子适应度值

fitness(j)=fun(pop(j,1:Dim),inputnum,hiddennum,outputnum,net,inputn,outputn,inputps,outputps);

end

%%个体极值和群体极值更新

for j=1:sizepop

%个体最优更新

if fitness(j) < fitnessgbest(j)

gbest(j,:) = pop(j,1:Dim);

fitnessgbest(j) = fitness(j);

end

%群体最优更新

if fitness(j) < fitnesszbest

zbest = pop(j,1:Dim);

fitnesszbest = fitness(j);

end

end

%%每代最优值记录到yy数组中

yy(i)=fitnesszbest;

end

%% 结果分析

plot(yy)

title(['适应度曲线 ' '终止代数＝' num2str(maxgen)],'fontsize',12);

xlabel('进化代数','fontsize',12);ylabel('适应度','fontsize',12);

x=zbest;

%% 把最优初始阀值权值赋予网络预测

% %用遗传算法优化的BP网络进行值预测

w1=x(1:inputnum\*hiddennum);

B1=x(inputnum\*hiddennum+1:inputnum\*hiddennum+hiddennum);

w2=x(inputnum\*hiddennum+hiddennum+1:inputnum\*hiddennum+hiddennum+hiddennum\*outputnum);

B2=x(inputnum\*hiddennum+hiddennum+hiddennum\*outputnum+1:inputnum\*hiddennum+hiddennum+hiddennum\*outputnum+outputnum);

net.iw{1,1}=reshape(w1,hiddennum,inputnum);

net.lw{2,1}=reshape(w2,outputnum,hiddennum);

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

net.b{2}=B2;

%% BP网络训练

%网络进化参数

net.trainParam.epochs=100;

net.trainParam.lr=0.1;

net.trainParam.goal=0.0000001;

%网络训练

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

%% BP网络预测

%数据归一化

%预测训练数据

inputn\_test=mapminmax('apply',[input1;input2;input3],inputps);

an=sim(net,inputn\_test);

%test\_simu=mapminmax('reverse',[output1],outputps);

anss=mapminmax('reverse',[an],outputps);

error=output1-anss;

figure(2)

plot(error)

title('仿真预测误差','fontsize',12);

xlabel('仿真次数','fontsize',12);ylabel('误差百分值','fontsize',12);

plot(output1,'\*','color',[29 131 8]/255);hold on

plot(anss,'-o','color',[244 208 0]/255,...

'linewidth',2,'MarkerSize',14,'MarkerEdgecolor',[138 151 123]/255);

legend('actua value','prediction')

title('预测本身数据')

xlabel('第x个样本'),ylabel('输出')

set(gca, 'Box', 'off', 'TickDir', 'out', 'TickLength', [.02 .02], ...

'XMinorTick', 'on', 'YMinorTick', 'on', 'YGrid', 'on', ...

'XColor', [.3 .3 .3], 'YColor', [.3 .3 .3],'LineWidth', 1)

%预测测试数据

an=sim(net,inputtn);

anss=mapminmax('reverse',[an],outputps);

error=outputt1-anss;

figure(3)

plot(error)

title('仿真预测误差','fontsize',12);

xlabel('仿真次数','fontsize',12);ylabel('误差百分值','fontsize',12);

plot(outputt1,'\*','color',[29 131 8]/255);hold on

plot(anss,'-o','color',[244 208 0]/255,...

'linewidth',2,'MarkerSize',14,'MarkerEdgecolor',[138 151 123]/255);

legend('actual value','prediction')

title('预测测试数据')

xlabel('第x个样本'),ylabel('输出')

set(gca, 'Box', 'off', 'TickDir', 'out', 'TickLength', [.02 .02], ...

'XMinorTick', 'on', 'YMinorTick', 'on', 'YGrid', 'on', ...

'XColor', [.3 .3 .3], 'YColor', [.3 .3 .3],'LineWidth', 1)

%预测

an=sim(net,inn);

anss=mapminmax('reverse',[an],outputps)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*