clear all

clc

%% 导入数据

[MIXtrain,MIXtest,DATAtrain,DATAtest]=online\_dataproduce();

p\_train = MIXtrain';

t\_train = DATAtrain';

p\_test = MIXtest';

t\_test = DATAtest';

%% 数据归一化

% 输入集

[pn\_train,inputps] = mapminmax(p\_train');

pn\_train = pn\_train';

pn\_test = mapminmax('apply',p\_test',inputps);

pn\_test = pn\_test';

% 输出集

[tn\_train,outputps] = mapminmax(t\_train');

tn\_train = tn\_train';

tn\_test = mapminmax('apply',t\_test',outputps);

tn\_test = tn\_test';

%% SVM模型创建/训练

[bestCVmse,bestc,bestg,pso\_option]=psoSVMcgForRegress(tn\_train,pn\_train);

% 创建/训练SVM

cmd = [' -t 2',' -c ',num2str(bestc),' -g ',num2str(bestg),' -s 3 -p 0.01'];

model = svmtrain(tn\_train,pn\_train,cmd);

%% SVM仿真预测

[Predict\_1,error\_1,tt1] = svmpredict(tn\_train,pn\_train,model);

[Predict\_2,error\_2,tt2] = svmpredict(tn\_test,pn\_test,model);

% 反归一化

predict\_1 = mapminmax('reverse',Predict\_1,outputps);

predict\_2 = mapminmax('reverse',Predict\_2,outputps);

% 结果对比

result\_1 = [t\_train predict\_1];

result\_2 = [t\_test predict\_2];

%% 绘图

figure(4)

plot(1:length(t\_train),t\_train,'r-\*',1:length(t\_train),predict\_1,'b:o')

grid on

legend('真实值','预测值')

xlabel('样本编号')

ylabel('信号')

string\_1 = {'训练集预测结果对比';

['mse = ' num2str(error\_1(2)) ' R^2 = ' num2str(error\_1(3))]};

title(string\_1)

figure(5)

plot(1:length(t\_test),predict\_2,'r-\*',1:length(t\_test),t\_test,'b:o')

grid on

legend('真实值','预测值')

xlabel('样本编号')

ylabel('信号')

string\_2 = {'测试集预测结果对比';

['mse = ' num2str(error\_2(2)) ' R^2 = ' num2str(error\_2(3))]};

title(string\_2)

k=0;

for i=1:178

if abs(t\_test(i)-predict\_2(i))<1

k=k+1;

end

end

k

function [bestCVmse,bestc,bestg,pso\_option] = psoSVMcgForRegress(train\_label,train)

%% 参数初始化

if nargin == 2

pso\_option = struct('c1',1.5,'c2',1.7,'maxgen',200,'sizepop',20, ...

'k',0.6,'wV',1,'wP',1,'v',5, ...

'popcmax',10^2,'popcmin',10^(-1),'popgmax',10^3,'popgmin',10^(-2));

end

% c1:初始为1.5,pso参数局部搜索能力

% c2:初始为1.7,pso参数全局搜索能力

% maxgen:初始为200,最大进化数量

% sizepop:初始为20,种群最大数量

% k:初始为0.6(k belongs to [0.1,1.0]),速率和x的关系(V = kX)

% wV:初始为1(wV best belongs to [0.8,1.2]),速率更新公式中速度前面的弹性系数

% wP:初始为1,种群更新公式中速度前面的弹性系数

% v:初始为3,SVM Cross Validation参数

% popcmax:初始为100,SVM 参数c的变化的最大值.

% popcmin:初始为0.1,SVM 参数c的变化的最小值.

% popgmax:初始为1000,SVM 参数g的变化的最大值.

% popgmin:初始为0.01,SVM 参数c的变化的最小值.

Vcmax = pso\_option.k\*pso\_option.popcmax;

Vcmin = -Vcmax ;

Vgmax = pso\_option.k\*pso\_option.popgmax;

Vgmin = -Vgmax ;

eps = 10^(-3);

%% 产生初始粒子和速度

for i=1:pso\_option.sizepop

% 随机产生种群和速度

pop(i,1) = (pso\_option.popcmax-pso\_option.popcmin)\*rand+pso\_option.popcmin;

pop(i,2) = (pso\_option.popgmax-pso\_option.popgmin)\*rand+pso\_option.popgmin;

V(i,1)=Vcmax\*rands(1,1);

V(i,2)=Vgmax\*rands(1,1);

% 计算初始适应度

cmd = ['-v ',num2str(pso\_option.v),' -c ',num2str( pop(i,1) ),' -g ',num2str( pop(i,2) ),' -s 3 -p 0.1'];

fitness(i) = svmtrain(train\_label, train, cmd);%=cg

end

% 找极值和极值点

[global\_fitness, bestindex]=min(fitness); % 全局极值

local\_fitness=fitness; % 个体极值初始化

global\_x=pop(bestindex,:); % 全局极值点

local\_x=pop; % 个体极值点初始化

% 每一代种群的平均适应度

avgfitness\_gen = zeros(1,pso\_option.maxgen);

%% 迭代寻优

for i=1:pso\_option.maxgen

for j=1:pso\_option.sizepop

%速度更新

V(j,:) = pso\_option.wV\*V(j,:) + pso\_option.c1\*rand\*(local\_x(j,:) - pop(j,:)) + pso\_option.c2\*rand\*(global\_x - pop(j,:));

if V(j,1) > Vcmax

V(j,1) = Vcmax;

end

if V(j,1) < Vcmin

V(j,1) = Vcmin;

end

if V(j,2) > Vgmax

V(j,2) = Vgmax;

end

if V(j,2) < Vgmin

V(j,2) = Vgmin;

end

%种群更新

pop(j,:)=pop(j,:) + pso\_option.wP\*V(j,:);

if pop(j,1) > pso\_option.popcmax

pop(j,1) = pso\_option.popcmax;

end

if pop(j,1) < pso\_option.popcmin

pop(j,1) = pso\_option.popcmin;

end

if pop(j,2) > pso\_option.popgmax

pop(j,2) = pso\_option.popgmax;

end

if pop(j,2) < pso\_option.popgmin

pop(j,2) = pso\_option.popgmin;

end

% 自适应粒子变异

if rand>0.5

k=ceil(2\*rand);

if k == 1

pop(j,k) = (20-1)\*rand+1;

end

if k == 2

pop(j,k) = (pso\_option.popgmax-pso\_option.popgmin)\*rand + pso\_option.popgmin;

end

end

%适应度值

cmd = ['-v ',num2str(pso\_option.v),' -c ',num2str( pop(j,1) ),' -g ',num2str( pop(j,2) ),' -s 3 -p 0.1'];

fitness(j) = svmtrain(train\_label, train, cmd);

%个体最优更新

if fitness(j) < local\_fitness(j)

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

local\_fitness(j) = fitness(j);

end

if fitness(j) == local\_fitness(j) && pop(j,1) < local\_x(j,1)

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

local\_fitness(j) = fitness(j);

end

%群体最优更新

if fitness(j) < global\_fitness

global\_x = pop(j,:);

global\_fitness = fitness(j);

end

if abs( fitness(j)-global\_fitness )<=eps && pop(j,1) < global\_x(1)

global\_x = pop(j,:);

global\_xx(j,:)=pop(j,:)%需要把global\_x这个量提取出来

global\_fitness = fitness(j);

end

avgfitness(i,j) = sum(global\_fitness)/pso\_option.sizepop;

end

fit\_gen(i)=global\_fitness;

avgfitness\_gen(i) = sum(fitness)/pso\_option.sizepop;

%-------新加的参量------%

% c\_plot(i)=global\_xx;

% g\_plot(i)=global\_xy;

end

pop

%% 结果分析

figure(3);

hold on;

plot(fit\_gen,'r\*-','LineWidth',1.5);

plot(avgfitness\_gen,'o-','LineWidth',1.5);

legend('最佳适应度','平均适应度');

xlabel('进化代数','FontSize',12);

ylabel('适应度','FontSize',12);

grid on;

bestc = global\_x(1);

bestg = global\_x(2);

bestCVmse = fit\_gen(pso\_option.maxgen);

line1 = '适应度曲线MSE[PSOmethod]';

line2 = ['(参数c1=',num2str(pso\_option.c1), ...

',c2=',num2str(pso\_option.c2),',终止代数=', ...

num2str(pso\_option.maxgen),',种群数量pop=', ...

num2str(pso\_option.sizepop),')'];

line3 = ['Best c=',num2str(bestc),' g=',num2str(bestg), ...

' CVmse=',num2str(bestCVmse),];

title({line1;line2;line3},'FontSize',12);

function Ig=gauss(I,ks,sigma2)

%%% 子函数: gauss()实现高斯平滑滤波

%%% 参数说明：

%%% I - 待平滑图像

%%% ks - 高斯核大小 (奇数)

%%% sigma2 - 高斯函数的方差

%%% Ig - 返回的高斯平滑后的图象

[Ny,Nx]=size(I);

hks=(ks-1)/2; % 高斯核的一半

%%- 一维卷积

if (Ny<ks)

x=(-hks:hks);

flt=exp(-(x.^2)/(2\*sigma2)); % 一维高斯函数

flt=flt/sum(sum(flt)); % 归一化

%%- 扩展

x0=mean(I(:,1:hks)); xn=mean(I(:,Nx-hks+1:Nx));

eI=[x0\*ones(Ny,ks) I xn\*ones(Ny,ks)];

Ig=conv(eI,flt);

Ig=Ig(:,ks+hks+1:Nx+ks+hks); % 截尾

else

%%- 二维卷积

x=ones(ks,1)\*(-hks:hks); y=x';

flt=exp(-(x.^2+y.^2)/(2\*sigma2)); % 二维高斯函数

flt=flt/sum(sum(flt)); % 归一化

%%- 扩展

if (hks>1)

xL=mean(I(:,1:hks)')'; xR=mean(I(:,Nx-hks+1:Nx)')';

else

xL=I(:,1); xR=I(:,Nx);

end

eI=[xL\*ones(1,hks) I xR\*ones(1,hks)];

if (hks>1)

xU=mean(eI(1:hks,:)); xD=mean(eI(Ny-hks+1:Ny,:));

else

xU=eI(1,:); xD=eI(Ny,:);

end

eI=[ones(hks,1)\*xU; eI; ones(hks,1)\*xD];

Ig=conv2(eI,flt,'valid');

end

clc;clear;

wr=[];

kmax=8;

kmin=3;

vv=kmax-kmin+1;

for k=3:8

wr(k-2)=QPSOfunciton(k);

end

minwr=min(wr);

x=find(wr==minwr);

t=max(x+2);

disp('-----------芝加哥用电模式分类数-------------：')

t

%-----------用QPSO算法--------------

load 239828497\_GCHICAGO;

xxx=Samples.LoadData(:,1);

tt=[];

[lnum,mmm]=size(xxx);

for i=1:lnum

tt(lnum+1-i)=xxx{i,1};

end

x=gauss(tt,4,0.8);

%x=[1.0000 1.0000 0.7476 0.6267 0.1696 0.0710 0.2532 0.8110];

%------给定初始化条件----------------------------------------------

k=t;

w=0.8; %惯性权重

Nmax=100; %最大迭代次数

D=2; %搜索空间维数（未知数个数）

M=20; %初始化群体个体数目

[N,N1]=size(x); %s搜素空间内点数

time=1:N1;

eps=0.1; %设置精度(在已知最小值时候用)

xmax=2;

y=x;

%------------初始化各粒子的聚类中心---------------------

C=zeros(20,k); %记录聚类中心点

for i=1:M

aa=1;

while aa==1

aa=0;

for zzz=1:k-1;

for j=zzz+1:k

if C(i,zzz)==C(i,j)

aa=1;

end

end

end

C(i,:)=randint(1,k,[1,N1]);

end

end

disp('----------样本点----------')

C

X=cell(20,k); %X{i,j}存储各个样本中各聚类中心的的位置

Y=cell(20,k); %Y{i,j}存储各个样本中各聚类中心的局部最优位置

Z=cell(1,k); %Z{i,j}存储所有样本的聚类中心的全局最优位置

%------先计算各个粒子的适应度，并初始化Pi和Pg----------------------

len=zeros(20,k);

L=zeros(20,N1);

for i=1:20 %对粒子循环

for j=1:N1 %对给定的样本点循环

for q=1:k %对聚类中心

d(j,q)=(x(1,j)-x(1,C(i,q)))^2;

end

aa=min(d(j,:));

bb=find(d(j,:)==aa);

len(i,bb)=len(i,bb)+d(j,bb);

L(i,j)=bb(1);

end

l(i)=sum(len(i,:));

fitness1(i)=l(i); %第i个粒子的局部最优

pi(i,:)=C(i,:); %聚类中心

for j=1:k

X{i,j}=x(:,C(i,j));

Y{i,j}=x(:,C(i,j));

end

end

F=zeros(50,k);

fitness2=fitness1(1); %全局最优

for i=1:20

if fitness1(i)<fitness2

fitness2=fitness1(i);

pg=C(i,:); %全局最优的聚类中心位置

good=i; %记录第几个粒子达到最好

for j=1:k

Z{j}=x(:,C(i,j));

end

end

end

F(1,:)=len(good,:);

fnum=1;

%------显示最好的聚类划分----------------------

figure

plot(time,x);

for j=1:N1

if L(good,j)==1

hold on;

plot(j,x(1,j),'.y');

elseif L(good,j)==2

hold on;

plot(j,x(1,j),'.g');

elseif L(good,j)==3

hold on;

plot(j,x(1,j),'.g');

elseif L(good,j)==4

hold on;

plot(j,x(1,j),'.g');

elseif L(good,j)==5

hold on;

plot(j,x(1,j),'.g');

elseif L(good,j)==6

hold on;

plot(j,x(1,j),'.g');

elseif L(good,j)==7

hold on;

plot(j,x(1,j),'.g');

elseif L(good,j)==8

hold on;

plot(j,x(1,j),'.g');

end

end

for i=1:3

hold on;

plot(pg(i),x(1,pg(i)),'or');

end

title('QPSO初始聚类划分结果：');

%------进入主要循环，按照公式依次迭代，直到满足精度要求------------

bianhua=0; %记录聚类中心是否变化

for t=1:Nmax

beta=(0.9-0.55)\*(Nmax-t)/Nmax+0.55;

E=zeros(1,20);

for i=1:20

x=y;

for j=1:k

x(:,C(i,j))=X{i,j};

end

if bianhua==0

%---------求样本点最小距离-------------------

MinDLength(i)=200; %样本点间的最小距离

for j=1:N1-1

for lp=2:N1

Len=(x(1,j)-x(1,lp))^2;

if Len<MinDLength(i)

MinDLength(i)=Len;

end

end

end

%----------结束-------------------------------

%----------求聚类中心之间的最小距离-------------

MinCLength(i)=200;

for j=1:k-1

for lp=2:k

Len=(x(1,C(i,j))-x(1,C(i,lp)))^2;

if Len<MinCLength(i)

MinCLength(i)=Len;

end

end

end

%------------结束-------------------------------

%--------------QPSO迭代------------------------

if MinCLength(i)>MinDLength(i)

%------------更新量子粒子位置-----------

for j=1:k

mbest=sum(X{i,j})/M;

fi=rand;

p=fi.\*X{i,j}+(1-fi).\*Z{j};

u=rand;

b=beta\*abs(mbest-X{i,j});

v=-log(u);

z=w\*p+((-1).^ceil(0.5+rand)).\*b.\*v;

X{i,j}=sign(z).\*min(abs(z),xmax);

end

else

bianhua=1; %标记聚类中心是否变化

aa=1;

while aa==1

aa=0;

for zzz=1:k-1;

for j=zzz+1:k

if C(i,zzz)==C(i,j)

aa=1;

end

end

end

C(i,:)=randint(1,k,[1,N1]);

end

end

end

%-----------对该样本重新进行聚类划分，并计算适应度-----------------

for j=1:k

x(1,C(i,j))=X{i,j};

end

len(i,:)=zeros(1,k);

for j=1:N1 %对给定的样本点循环

for q=1:k %对聚类中心

d(j,q)=(x(1,j)-x(1,C(i,q)))^2;

end

aa=min(d(j,:));

bb=find(d(j,:)==aa);

len(i,bb)=len(i,bb)+d(j,bb);

L(i,j)=bb(1);

E(i)=sqrt(d(j,k))+E(i);

end

add(i)=sum(len(i,k));

if add(i)<fitness1(i)

fitness1(i)=add(i);

for j=1:k

Y{i,j}=X{i,j};

end

end

if add(i)<fitness2

fitness2=add(i);

fgood=E(i);

fnum=fnum+1;

F(fnum,:)=fgood;

for j=1:k

Z{j}=X{i,j};

end

good=i; %记录达到全局最优的样本为哪个

goodL=L(i,:);

end

end

Emin(t)=fitness2;

end

%--------------结束------------------------------------

%----------显示聚类结果--------------------------

disp('-----------QPSO迭代次数-------------：')

t

disp('-----------QPSO最终聚类中心位置为-------：')

C(good,:)

x=y;

for j=1:k

x(1,C(good,j))=Z{j};

end

figure

for j=1:N1

if goodL(j)==1

hold on;

plot(j,x(1,j),'.y');

elseif goodL(j)==2

hold on;

plot(j,x(1,j),'.g');

elseif goodL(j)==3

hold on;

plot(j,x(1,j),'.k');

elseif goodL(j)==4

hold on;

plot(j,x(1,j),'.c');

elseif goodL(j)==5

hold on;

plot(j,x(1,j),'.m');

elseif goodL(j)==6

hold on;

plot(j,x(1,j),'.w');

elseif goodL(j)==7

hold on;

plot(j,x(1,j),'.r');

elseif goodL(j)==8

hold on;

plot(j,x(1,j),'.b');

end

end

for j=1:k

hold on;

plot(C(good,j),x(1,C(good,j)),'or');

end

title('QPSO聚类划分最好结果：');

function vvv=QPSOfunciton(k)

load 239828497\_GCHICAGO;

xxx=Samples.LoadData(:,1);

tt=[];

[lnum,mmm]=size(xxx);

for i=1:lnum

tt(lnum+1-i)=xxx{i,1};

end

x=gauss(tt,4,0.8);

% x=[1.0000 1.0000 0.7476 0.6267 0.1696 0.0710 0.2532 0.8110];

%------给定初始化条件----------------------------------------------

w=0.8; %惯性权重

Nmax=100; %最大迭代次数

D=2; %搜索空间维数（未知数个数）

M=20; %初始化群体个体数目

[N,N1]=size(x); %s搜素空间内点数

time=1:N1;

eps=0.1; %设置精度(在已知最小值时候用)

xmax=2;

y=x;

%------------初始化各粒子的聚类中心---------------------

C=zeros(M,k); %记录聚类中心点

for i=1:M

aa=1;

while aa==1

aa=0;

for zzz=1:k-1;

for j=zzz+1:k

if C(i,zzz)==C(i,j)

aa=1;

end

end

end

C(i,:)=randint(1,k,[1,N1]);

end

end

X=cell(M,k); %X{i,j}存储各个样本中各聚类中心的的位置

Y=cell(M,k); %Y{i,j}存储各个样本中各聚类中心的局部最优位置

Z=cell(1,k); %Z{i,j}存储所有样本的聚类中心的全局最优位置

%------先计算各个粒子的适应度，并初始化Pi和Pg----------------------

len=zeros(M,k);

L=zeros(M,N1);

pi=zeros(M,k);

for i=1:M %对粒子循环

for j=1:N1 %对给定的样本点循环

for q=1:k %对聚类中心

d(j,q)=(x(1,j)-x(1,C(i,q)))^2;

end

aa=min(d(j,:));

bb=find(d(j,:)==aa);

len(i,bb)=len(i,bb)+d(j,bb);

L(i,j)=bb(1);

end

l(i)=sum(len(i,:));

fitness1(i)=l(i); %第i个粒子的局部最优

pi(i,:)=C(i,:); %聚类中心

for j=1:k

X{i,j}=x(:,C(i,j));

Y{i,j}=x(:,C(i,j));

end

end

F=zeros(50,k);

fitness2=fitness1(1); %全局最优

for i=1:20

if fitness1(i)<fitness2

fitness2=fitness1(i);

pg=C(i,:); %全局最优的聚类中心位置

good=i; %记录第几个粒子达到最好

for j=1:k

Z{j}=x(:,C(i,j));

end

end

end

fnum=1;

%------进入主要循环，按照公式依次迭代，直到满足精度要求------------

bianhua=0; %记录聚类中心是否变化

for t=1:Nmax

beta=(0.9-0.55)\*(Nmax-t)/Nmax+0.55;

E=zeros(1,M);

for i=1:M

x=y;

for j=1:k

x(:,C(i,j))=X{i,j};

end

if bianhua==0

%---------求样本点最小距离-------------------

MinDLength(i)=200; %样本点间的最小距离

len11=[];

for j=1:N1-1

for lp=2:N1

Len=(x(1,j)-x(1,lp))^2;

len11=Len;

if Len<MinDLength(i)

MinDLength(i)=Len;

end

end

end

%----------结束-------------------------------

%----------求聚类中心之间的最小距离-------------

MinCLength(i)=200;len22=[];

for j=1:k-1

for lp=2:k

Len=(x(1,C(i,j))-x(1,C(i,lp)))^2;

len22=Len;

if Len<MinCLength(i)

MinCLength(i)=Len;

end

end

end

%------------结束-------------------------------

%--------------QPSO迭代------------------------

if MinCLength(i)>MinDLength(i)

%------------更新量子粒子位置-----------

for j=1:k

mbest=sum(X{i,j})/M;

fi=rand;

p=fi.\*X{i,j}+(1-fi).\*Z{j};

u=rand;

b=beta\*abs(mbest-X{i,j});

v=-log(u);

z=w\*p+((-1).^ceil(0.5+rand)).\*b.\*v;

X{i,j}=sign(z).\*min(abs(z),xmax);

end

else

bianhua=1; %标记聚类中心是否变化

aa=1;

while aa==1

aa=0;

for zzz=1:k-1;

for j=zzz+1:k

if C(i,zzz)==C(i,j)

aa=1;

end

end

end

C(i,:)=randint(1,k,[1,N1]);

end

end

end

end

%-----------对该样本重新进行聚类划分，并计算适应度-----------------

for j=1:k

x(1,C(i,j))=X{i,j};

end

len(i,:)=zeros(1,k);

for j=1:N1 %对给定的样本点循环

for q=1:k %对聚类中心

d(j,q)=(x(1,j)-x(1,C(i,q)))^2;

end

aa=min(d(j,:));

bb=find(d(j,:)==aa);

len(i,bb)=len(i,bb)+d(j,bb);

L(i,j)=bb(1);

E(i)=sqrt(d(j,k))+E(i);

end

add(i)=sum(len(i,k));

if add(i)<fitness1(i)

fitness1(i)=add(i);

for j=1:k

Y{i,j}=X{i,j};

end

end

if add(i)<fitness2

fitness2=add(i);

fgood=E(i);

fnum=fnum+1;

for j=1:k

Z{j}=X{i,j};

end

good=i; %记录达到全局最优的样本为哪个

goodL=L(i,:);

end

end

Emin(t)=fitness2;

vvv=sqrt(0.5\*len11^2+0.5\*len22^2);

end