```
%%A PSO-GRNN Model for Railway Freight Volume Prediction by Sun Yan
%%数据预处理
p_train=p(1:16,:);
t_train=t(1:16,:);
p_test=p(17:21,:);
t_test=t(17:21,:);
p_zz_train=p_train';%矩阵的转置
t_zz_train=t_train';
p_zz_test=p_test';
t_zz_test=t_test';
p_gy_train=mapminmax(p_zz_train,0,1);%归一化至（0,1）区间
t_gy_train=mapminmax(t_zz_train,0,1);%归一化至（0,1）区间
p_gy_test=mapminmax(p_zz_test,0,1);%归一化至（0,1）区间
t_gy_test=mapminmax(t_zz_test,0,1);%归一化至（0,1）区间

%%PSO 算法参数设置
c1initial=0.1;%c1 起始值
c1final=0.05;%c1 终止值
c2initial=0.05;%c2 起始值
c2final=0.1;%c2 终止值
wmin=0.1;%w 最小值
wmax=0.5;%w 最大值
popmax=1;%pop 最大值
popmin=0;%pop 最小值
vmax=0.01;%v 最大值
vmin=-0.01;%v 最小值
maxgen=150;%最大迭代次数
popsize=40;%种群规模

%%测试样本期望输出矩阵的构建
test_out_expect=repmat(t_gy_test,popsize,1);%期望输出矩阵的维度与输出矩阵维度
相同,repmat(复制矩阵名称,复制行数,复制列数)

%%初始化:计算每个粒子的适应度值
for i=1:popsize
    pop(i,:)=abs(rands(1,1));%确定每个粒子的位置，在(0,1)之间，abs 绝对值函数
    v(i,:)=rands(1,1)*0.01;%确定每个粒子的速度
    spread(i)=pop(i,:);%设置光滑因子取值，光滑因子必须是正值
    net=newgrnn(p_gy_train,t_gy_train,spread(i));%构建广义回归神经网络
    test_out_sim(i,:)=sim(net,p_gy_test);%输出测试结果
    error(i,:)=test_out_expect(i,:)-test_out_sim(i,:);%计算绝对误差
    fitness(i)=mse(error(i,:));%计算均方差函数值，得到适应值
end

%%寻找初始化个体最优和初始化全局最优
[bestfitness bestindex]=min(fitness);
gbest=pop(bestindex,:);%寻找全局最佳位置
pbest=repmat(pop,1,1);%寻找个体最佳位置
fitnesspbest=fitness;%个体最佳适应值
fitnessgbest=bestfitness;%全局最佳适应值
```

```
%%PSO 算法迭代寻优
for j=1:maxgen
    %个体最优更新
    %粒子速度更新
    for i=1:popsize
    w=wmax-(wmax-wmin)*j/maxgen;%获得惯性权重
    c1=(c1final-c1initial)*j/maxgen+c1initial;%获得加速因子 c1
    c2=(c2final-c2initial)*j/maxgen+c2initial;%获得加速因子 c2
    v(i,:)=           w*v(i,:)+           c1*rand*(pbest(i,:)-pop(i,:))+
    c2*rand*(gbest-pop(i,:));%更新粒子速度
    v(i,find(v(i,:)>vmax))=vmax;%最大速度约束
    v(i,find(v(i,:)<vmin))=vmin;%最小速度约束
    %粒子位置更新
    pop(i,:)=pop(i,:)+v(i,:);%更新粒子位置
    pop(i,find(pop(i,:)>popmax))=popmax;%最大位置约束
    pop(i,find(pop(i,:)<popmin))=popmin;%最小位置约束
    %粒子适应值更新
    spread(i)=pop(i,:);
    net=newgrnn(p_gy_train,t_gy_train,spread(i));
    test_out_sim(i,:)=sim(net,p_gy_test);
    error(i,:)=test_out_expect(i,:)-test_out_sim(i,:);
    fitness(i)=mse(error(i,:));
    end
    for i=1:popsize
    if fitness(i)<fitnesspbest(i)
    pbest(i,:)=pop(i,:);
    fitnesspbest(i)=fitness(i);
    end
    end
    %全局最优更新
    for i=1:popsize
    if fitness(i)<fitnessgbest
    gbest=pop(i,:);
    fitnessgbest=fitness(i);
    end
    end
    aa(j)=fitnessgbest;%每次迭代的最优适应值
    bb(j)=gbest;%每次迭代的最优粒子位置
end
%%绘图,在一个图中画若干条曲线时，加 hold on
plot(aa)
xlabel('Iteration Process', 'fontsize',12);
ylabel('Best Fitness Value of the Swarm', 'fontsize',12);
plot(bb)
xlabel('Iteration Process', 'fontsize',12);
ylabel('Best Position of the Swarm', 'fontsize',12);

%%寻找最佳光滑因子，以此构建 GRNN
[globalbestfitness globalbestindex]=min(aa);
globalbestspread=bb(globalbestindex);
net=newgrnn(p_gy_train,t_gy_train,globalbestspread);
test_out_best=sim(net,p_gy_test);
```

```
%%BPNN 预测
p_train=p(1:16,:);
t_train=t(1:16,:);
p_test=p(17:21,:);
t_test=t(17:21,:);
p_zz_train=p_train';
t_zz_train=t_train';
p_zz_test=p_test';
t_zz_test=t_test';
p_gy_train=mapminmax(p_zz_train,0,1);%归一化至（0,1）区间
t_gy_train=mapminmax(t_zz_train,0,1);%归一化至（0,1）区间
p_gy_test=mapminmax(p_zz_test,0,1);%归一化至（0,1）区间
t_gy_test=mapminmax(t_zz_test,0,1);%归一化至（0,1）区间
NodeNum=33;
TypeNum=1;
Epochs=500;
TF1='logsig';TF2='logsig';
net=newff(minmax(p_gy_train),[NodeNum TypeNum],{TF1 TF2},'trainlm');
net.trainParam.epochs=Epochs;
net.trainParam.goal=1e-7;
net=train(net,p_gy_train,t_gy_train);
t_gy_sim=sim(net,p_gy_test);


%%GRNN
p_train=p(1:16,:);
t_train=t(1:16,:);
p_test=p(17:21,:);
t_test=t(17:21,:);
p_zz_train=p_train';
t_zz_train=t_train';
p_zz_test=p_test';
t_zz_test=t_test';
p_gy_train=mapminmax(p_zz_train,0,1);
t_gy_train=mapminmax(t_zz_train,0,1);
p_gy_test=mapminmax(p_zz_test,0,1);
t_gy_test=mapminmax(t_zz_test,0,1);
spread=1;
net=newgrnn(p_gy_train,t_gy_train,spread);
test_out_sim=sim(net,p_gy_test);
>> error=test_out_sim-t_gy_test;
>> fitness=mse(error);


%%RBF
p_train=p(1:16,:);
t_train=t(1:16,:);
p_test=p(17:21,:);
t_test=t(17:21,:);
p_zz_train=p_train';
t_zz_train=t_train';
p_zz_test=p_test';
t_zz_test=t_test';
p_gy_train=mapminmax(p_zz_train,0,1);
t_gy_train=mapminmax(t_zz_train,0,1);
p_gy_test=mapminmax(p_zz_test,0,1);
```

```
t_gy_test=mapminmax(t_zz_test,0,1);
spread=1;
net=newrbe(p_gy_train,t_gy_train,spread);
test_out_sim=sim(net,p_gy_test);
>> error=test_out_sim-t_gy_test;
>> fitness=mse(error);
```